TecnoCampus
Escola Superior Politècnica

Centre adscrit a la
upf. Universitat Pompeu Fabra Barcelona

Grado en Ingeniería Informática de Gestión y Sistemas de Información

DESARROLLO FULL STACK PARA LA PLATAFORMA TENDIOS

Memoria

DAVID JIMÉNEZ BELMONTE
TUTOR: ALFREDO RUEDA UNSAÍN

CURSO 2022-2023

TecnoCampus
Mataró-Maresme

Agradecimientos

Quiero expresar mi más sincero agradecimiento a mi tutor Alfredo Rueda por su orientación, paciencia y valiosos consejos a lo largo de todo el proceso. Su apoyo constante ha sido fundamental para mi crecimiento académico y personal.

También quiero agradecer a mi familia y amigos por estar a mi lado durante esta etapa. Su apoyo incondicional ha sido un gran respaldo en los momentos difíciles y me han dado la motivación necesaria para seguir adelante.

Con gratitud,

David Jiménez Belmonte

Abstract

The objective of this project is to develop an application capable of tracking tenders from the main sources. To achieve this, Python-based scrappers are used, the Back End is adapted using Node.js, and the front end is developed with Vue.js. The entire development process follows the SCRUM methodology. As a result, the application is successfully developed.

Resum

L'objectiu d'aquest projecte és desenvolupar una aplicació capaç de rastrejar licitacions de les principals fonts. Per aconseguir-ho, s'utilitzen rastrejadors programats en Python, s'adapta el Back End amb Node.js i es desenvolupa el front end amb Vue.js. Tot el procés de desenvolupament es realitza seguint la metodologia SCRUM. Com a resultat, s'aconsegueix desenvolupar amb èxit l'aplicació.

Resumen

El objetivo de este proyecto es desarrollar una aplicación capaz de rastrear licitaciones de las principales fuentes. Para lograrlo, se utilizan rastreadores programados en Python, se adapta el Back End con Node.js y se desarrolla el Front End con Vue.js. Todo el proceso de desarrollo se lleva a cabo siguiendo la metodología SCRUM. Como resultado, se consigue desarrollar con éxito la aplicación.

Índice

Índice de figuras

Glosario de términos

ESUPT	Escuela Superior Politécnica - Tecnocampus
TFG	Trabajo de Fin de Grado
API	*Application Programming Interface* / Interfaz de Programación de Aplicaciones
CPU	*Central Processing Unit* / Unidad Central de Proceso
RAM	*Random Access Memory* / Memoria de Acceso Aleatorio
BOE	Boletín Oficial del Estado
DRE	*Diário da República*
TED	*Tenders Electronic Daily*

1. Introducción

El proyecto incluye un desarrollo de Back End y Front End para la plataforma web Tendios. Incluye dos secciones: Administrador y Superadministrador. Los administradores de la plataforma ven cuentas creadas, usuarios registrados, suscripciones activas y acceden a un dashboard con métricas de su interés. El superadministrador consulta errores en tiempo real de la plataforma y el estado de los rastreadores encargados de recopilar información.

El proyecto incluye también la creación de rastreadores para el Boletín Oficial del Estado (BOE), *Diário da República* (DRE), el Diario Oficial de la Unión Europea (DOUE) y las licitaciones y Contratos Menores publicados en la Plataforma de Contratación del Sector Público para recopilar licitaciones diarias y guardarlas en la base de datos de Tendios.

Otro aspecto del proyecto es la integración dentro de la plataforma de herramientas de terceros como Twitter, Mixpanel, Google y LinkedIn a través de sus API.

La plataforma actualmente cuenta con un Back End funcional con un modelo de licitación y un rastreador funcionando, así como un Front End con una plantilla implementada, el esquema del punto de inicio es el siguiente:

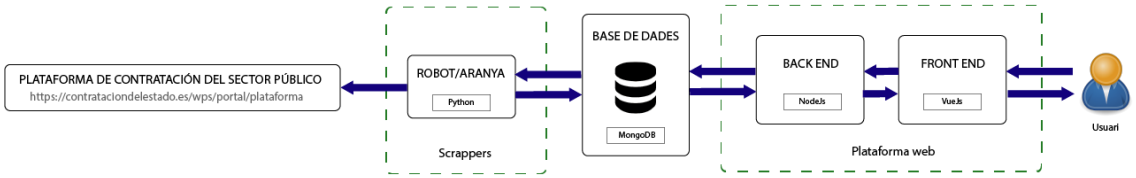

Figura 1.1 Esquema del punto de inicio

El objetivo de la realización de este proyecto es aumentar en gran medida el valor para los clientes de la plataforma Tendios respecto al punto de origen y aportar así a la plataforma una gran ventaja competitiva respecto al resto de competidores del sector.

Durante el desarrollo de este proyecto, se siguen varios pasos para alcanzar los objetivos establecidos. En primer lugar, se procede al desarrollo de los diferentes rastreadores, al mismo tiempo que se actualiza el Back End para adaptarlo a estas nuevas funcionalidades. Una vez completada esta etapa, se finaliza el desarrollo del Back End, implementando una nueva lógica de usuarios más sencilla y eficiente. Por último, se lleva a cabo la implementación del Front End, utilizando los endpoints creados en la fase anterior.

En cuanto a los procedimientos aplicados, se opta por utilizar una metodología de trabajo basada en SCRUM. Esto permite una gestión ágil del proyecto, facilitando la adaptación a los cambios y fomentando la colaboración en el equipo de desarrollo.

Durante el proceso, se encuentran diversos problemas relacionados con el *web scraping*. Esta técnica demuestra ser altamente volátil, con la posibilidad de que deje de funcionar de un día para otro. Un ejemplo de esto es el rastreador desarrollado para la página del Diário da República (DRE), que se ve afectado por un cambio en el formato de la página en marzo, lo cual imposibilita el *scraping* de dicha fuente de información. Además, en ocasiones se presentan situaciones en las que algunas páginas modifican los identificadores de sus elementos, lo que ocasiona que los rastreadores dejen de funcionar. La naturaleza dinámica de la carga de la página del DRE también genera dificultades iniciales para recopilar la información de manera precisa.

A pesar de estos desafíos, se logran obtener resultados satisfactorios. La aplicación se desarrolla con éxito, aportando la mayoría de las funcionalidades requeridas a pesar de los obstáculos encontrados durante el proceso.

En cuanto a las conclusiones más importantes, se destaca la importancia de contar con una metodología de trabajo ágil y flexible, como SCRUM, para gestionar proyectos de desarrollo de software. Asimismo, se resalta la complejidad y volatilidad del *web scraping*, siendo necesario anticipar y enfrentar cambios en las fuentes de información. A pesar de las dificultades, se logra superar los obstáculos y entregar una solución funcional.

2. Marco teórico

A continuación, se procede a analizar con detenimiento los antecedentes del proyecto y los conceptos básicos que se tienen que aclarar para conocer y entender el procedimiento que se quiere seguir durante la relación del proyecto.

2.1 Contexto

Dentro del mercado encontramos diferentes frameworks con funcionalidades similares a las planteadas en el proyecto. Un ejemplo es Django, que basa su valor en la capacidad de desarrollar aplicaciones web de manera rápida y sencilla. Django ofrece a sus clientes un dashboard desde el cual un usuario administrador puede consultar diferentes datos de la plataforma, como usuarios, el último inicio de sesión, visitas a la página, etc. Esta funcionalidad es similar a la propuesta en el proyecto para el dashboard del administrador.

Por otro lado, también encontramos frameworks como Bugsnag, Sentry, Raygun, Datadog, etc. que ofrecen una monitorización en tiempo real de las excepciones que ocurren en el código o página web. Eso es parecido al punto de la propuesta de la sección de superadministrador en el que se quiere mostrar los errores que se producen durante la ejecución de los rastreadores. Estos frameworks se pueden mejorar integrándolos dentro de la propia plataforma, con su dashboard de errores, en lugar de requerir al usuario que acceda a otra página para consultarlos.

En lo referente a la propia plataforma, en el mercado hay varias páginas web con funcionalidades similares a la de Tendios, como Infonalia, Licitaciones.es, Gestboes y Acobur. Aunque estas páginas rastrean licitaciones de las fuentes principales, Tendios se distingue de ellas al adoptar un modelo de Software como Servicio.

2.2 Necesidades de información

En este apartado, se presentan los conceptos y elementos más importantes que se abordarán en el proyecto de manera teórica. Conocer estos puntos es esencial para comprender el funcionamiento general de la aplicación y los aspectos que se desarrollarán durante su realización.

2.2.1 Licitación

Una licitación es un proceso formal y competitivo en el que una entidad pública o privada, llamada "entidad adjudicadora", busca contratar servicios o adquirir bienes para realizar un proyecto específico. La entidad adjudicadora publica un anuncio de licitación en el que especifica los requisitos y condiciones para participar. Las empresas interesadas presentan sus ofertas y la entidad adjudicadora evalúa y selecciona la oferta más conveniente de acuerdo a criterios previamente establecidos, como precio, calidad y plazo de entrega.

El proceso de licitación es un mecanismo para garantizar transparencia y equidad en la contratación pública y privada y prevenir posibles conflictos de intereses o corrupción. Es una herramienta esencial para lograr una contratación eficiente y eficaz y obtener una solución óptima para la entidad adjudicadora y los proveedores.

En el sector público, la licitación es un proceso regulado por leyes y normativas y requiere un proceso riguroso y transparente para garantizar un uso adecuado y eficiente de los recursos públicos. En el sector privado, la licitación puede ser un proceso voluntario para conseguir la mejor oferta.

2.2.2 Web Scraping

El *web scraping* consiste en una técnica de extracción de datos que permite obtener información de sitios web de manera automatizada mediante programas de software o scripts. Estos scripts navegan por la estructura de la página web y extraen información relevante, incluyendo texto, imágenes, videos, tablas, links, entre otros.

Se utiliza en una amplia gama de aplicaciones, como la investigación de mercado, la búsqueda de empleo, la comparación de precios y el monitoreo de redes sociales. Algunas empresas lo emplean para obtener información valiosa sobre sus competidores y su mercado, mientras que investigadores y periodistas lo utilizan para recopilar datos relevantes para sus investigaciones y reportes.

Es importante tener en cuenta que el *web scraping* puede ser ilegal si se extraen datos sin autorización y se violan las políticas de privacidad y los derechos de autor de un sitio web. Por ello, es necesario conocer las leyes y regulaciones aplicables antes de realizar dicha práctica.

En definitiva, el *web scraping* es una técnica valiosa para obtener datos e información en grandes cantidades de manera automatizada, siempre y cuando se utilice de manera responsable y respetando las leyes y regulaciones aplicables.

2.2.3 MongoDB

MongoDB es un sistema de base de datos de documentos NoSQL o no relacional. Está diseñado específicamente para modelos de datos específicos y tiene un esquema flexible. Estos tipos de base de datos están optimizados para aplicaciones que requieren grandes volúmenes de datos, baja latencia y modelos de datos flexibles.

MongoDB es diferente a las bases de datos SQL tradicionales, ya que almacena los datos en documentos en lugar de en registros. Estos documentos se almacenan en formato BSON, que consiste en una representación binaria del formato JSON.

Esto significa que los objetos de una misma colección pueden tener esquemas diferentes y no es necesario seguir un esquema predefinido. Por ejemplo, dos documentos en una colección llamada "Tienda" pueden almacenarse de la siguiente manera:

Figura 2.1 Ejemplos de documentos de MongoDB

Las principales características de MongoDB relevantes para el proyecto son [3]:

- Velocidad: Las consultas a la base de datos desde la aplicación son rápidas debido al equilibrio entre rendimiento y funcionalidad que ofrece su sistema de consulta de contenido.

- Consultas ad hoc: MongoDB permite realizar consultas a base de datos por campos específicos, consultas por rangos de valores o por expresiones regulares. Estas consultas pueden devolver un documento entero, un campo específico del documento o el usuario puede definir que objeto recibe mediante una función JavaScript.
- Indexación: La indexación en MongoDB es parecida a la utilizada en las bases de datos relacionales, se diferencia de estas, ya que cualquier campo del documento puede ser indexado y permite añadir múltiples índices secundarios.
- Ejecución de JavaScript del lado del servidor: MongoDB permite realizar consultas utilizando JavaScript, haciendo que estas sean enviadas directamente a la base de datos para ser ejecutadas.

2.2.4 Node.js

Node.js es un entorno de ejecución creado por los propios desarrolladores de JavaScript para permitir al lenguaje ser ejecutado en los propios ordenadores a modo de aplicaciones independientes en vez de ser solo capaz de ejecutarse en navegadores web.

Este entorno incluye todo lo necesario para ejecutar un programa escrito en JavaScript, el cual se ejecuta en el motor de tiempo de ejecución JavaScript V8, que a su vez convierte el código en un código de máquina más rápido. Esto aumenta la velocidad, ya que el código máquina es un código de nivel más bajo que se puede ejecutar sin necesidad de interpretarlo o compilarlo.

Node.js utiliza un modelo de solicitudes y respuestas sin bloqueo controlado por eventos, que funciona mediante lo que se denomina como el bucle de eventos. Cada vez que se genera una nueva solicitud, se crea un nuevo evento, el cual tiene un tipo dependiendo de ciertas características, y se añade al bucle de eventos.

En la siguiente imagen se ve un esquema del funcionamiento del bucle de eventos (Fuente: [4]).

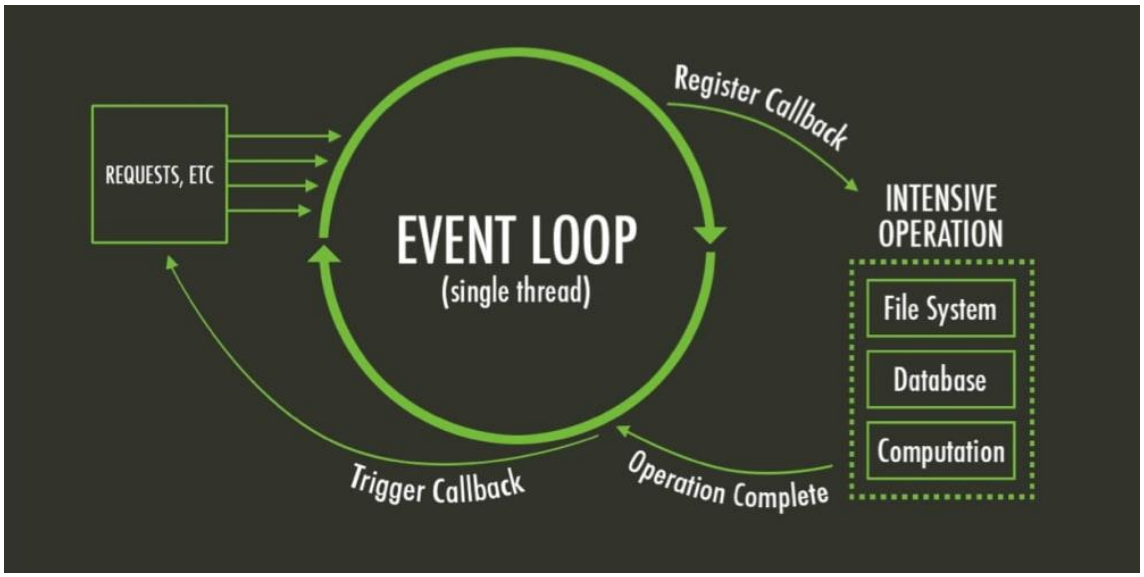

Figura 2.2 Esquema funcionamiento del Event Loop

En cada iteración, el servidor recorre el bucle de eventos y procesa cada evento. En cada iteración, el servidor procesa los eventos en el siguiente orden [5]:

- Primero los eventos de temporizador, generados por los métodos setTimout() y setInterval().
- En segundo lugar, intenta resolver los callbacks pendientes de la anterior iteración.
- Seguidamente, gestiona los eventos de tipo “Idle” e “Ignore”, estos son de uso interno únicamente.
- Después, recupera los nuevos eventos de entrada y salida generados en esta iteración y los ejecuta (a excepción de los callbacks de cierre, los programados por temporizadores y setInmediate()).
- Los próximos eventos a gestionar son los de tipo “check”, que simplemente invocan los callbacks setIntermediate().
- Por último, procesa los callbacks de cierre, como por ejemplo cerrar un socket.

En la siguiente imagen se encuentra un esquema de un bucle de ejecución (Fuente: [5]).

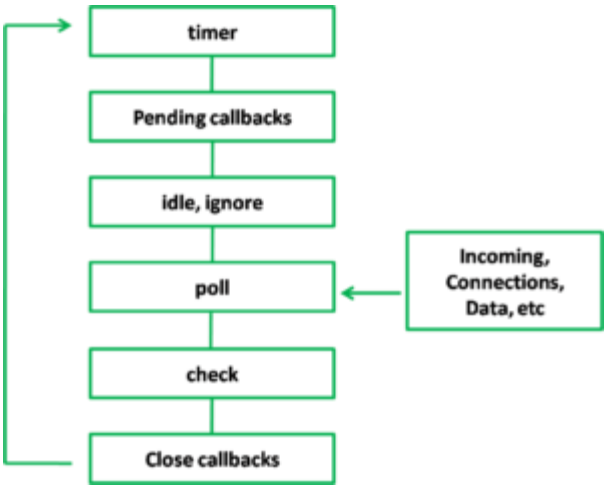

Figura 2.3 Esquema de orden de ejecución de una iteración

Esta manera de funcionar diferencia a Node.js del resto de técnicas tradicionales de servicio web. Estas funcionan generando un subproceso para cada solicitud pendiente, donde cada subproceso ocupa un espacio en la memoria RAM del dispositivo. En cambio, con el sistema que usa Node.js, este se ejecuta enteramente en un único subproceso que recorre constantemente el bucle de eventos.

Esto permite a Node.js ser un entorno de ejecución rápido y eficiente, lo que permite el desarrollo de aplicaciones de red de forma rápida, ya que puede manejar una gran cantidad de conexiones simultáneas con un gran rendimiento, lo que le da también una gran escalabilidad.

2.2.5 Mongoose

Mongoose es un marco de trabajo de Node.js para MongoDB, que proporciona un enfoque de programación orientada a objetos para interactuar con la base de datos. Algunas de las principales características de Mongoose son:

- Abstracción de esquema: Mongoose permite definir un esquema para los documentos de MongoDB, lo que significa que se puede especificar qué tipos de datos son requeridos, qué valores son permitidos y cómo deben ser validados.
- Validación de datos: Mongoose incluye un sistema de validación de datos integrado, que permite validar los datos antes de guardarlos en la base de datos.

- Middleware: Mongoose proporciona una serie de "ganchos" que se ejecutan antes y después de las operaciones de base de datos, como guardar, actualizar y eliminar, lo que permite agregar lógica adicional a estos eventos.
- Enlaces virtuales: Mongoose permite definir enlaces virtuales entre documentos, lo que significa que puedes establecer relaciones entre ellos sin tener que almacenarlos en la misma colección.
- Protección de seguridad: Mongoose proporciona una serie de características de seguridad, como la ocultación de campos y la validación de permisos, para ayudar a proteger los datos.
- Modificadores de consulta: Mongoose proporciona una serie de métodos para modificar las consultas que se realizan a la base de datos, lo que permite personalizar la forma en que se recuperan y procesan los datos.

2.2.6 Vue.js

Vue.js es un marco de JavaScript progresivo para construir interfaces de usuario. Fue desarrollado por Evan You y lanzado por primera vez en febrero de 2014. Vue.js se enfoca en ser una biblioteca ligera y fácil de usar para construir aplicaciones web complejas.

Las principales características de Vue.js son [6]:

- Enfoque en la experiencia de usuario: Vue.js está diseñado para crear aplicaciones web intuitivas y de fácil uso para los usuarios finales.
- Reactivo: Vue.js utiliza un enfoque reactivo para las actualizaciones de datos en tiempo real, lo que significa que los cambios en los datos se reflejan automáticamente en la vista.
- Componentes: Vue.js utiliza un sistema de componentes para organizar y reutilizar el código de la aplicación. Los componentes son bloques reutilizables de HTML, CSS y JavaScript que se pueden combinar para crear aplicaciones complejas.
- Directivas: Vue.js proporciona una serie de directivas para manipular la lógica de la aplicación en el lado del cliente. Estas directivas permiten la manipulación dinámica del HTML y la interacción con el usuario.
- Renderizado de servidor: Vue.js ofrece una opción para el renderizado de servidor, lo que permite mejorar la velocidad de carga y la optimización de SEO.

- Instancias de Vue: Vue.js utiliza instancias de Vue como el núcleo de la aplicación, lo que permite un control completo sobre el estado y la lógica de la aplicación.
- Comunidad: Vue.js cuenta con una comunidad activa y en crecimiento, lo que significa que hay una gran cantidad de recursos y documentación disponibles.

3. Objetivos y alcance

3.1 Objetivos

A continuación, se describen punto por punto cada uno de los objetivos incluidos dentro del proyecto. Dichos objetivos pueden variar ligeramente durante la realización del trabajo debido a necesidades de la propia empresa colaboradora.

- Se crea mediante Python los rastreadores de licitaciones para las plataformas de:
    - Boletín Oficial del Estado (BOE).
    - *Diário da República* (DRE).
    - Diario Oficial de la Unión Europea (DOUE).
    - Sección de Contratos Menores de la Plataforma de Contratación del Sector Público.
- Se crea con JavaScript y Vue.js la sección de Superadministrador, en la cual se muestra:
    - Errores en tiempo real que ocurran en la plataforma (Bug Tracking).
    - Estado de los rastreadores.
    - Posibles errores en los rastreadores.
- Se crea a través de JavaScript y Vue.js la sección de Administración, en la cual se muestra:
    - Cuentas creadas.
    - Usuarios registrados.
    - Suscripciones activas.
    - Dashboard con métricas de interés para los administradores
- Se integran dentro de la plataforma herramientas de terceros a través de sus respectivas API:
    - Twitter.
    - Mixpanel.
    - Google.
    - LinkedIn.

3.2 Alcance

El alcance del proyecto comprende la correcta implementación y funcionamiento de los objetivos mencionados con anterioridad, así como adaptar la base desde la que se parte a las nuevas funcionalidades que se introducen.

Para los rastreadores se crea uno para cada una de las plataformas mencionadas en los objetivos, pero solo se rastrean las licitaciones de la sección "Contratación del Sector Público" debido a los intereses de la empresa colaboradora. El resto de secciones no se tienen en cuenta para la implementación de esta parte del proyecto.

Debido a que cada plataforma publica información de manera diferente, cada rastreador debe estar adaptado para recabar la máxima información de su respectiva página web y se debe modificar el modelo actual de licitación en la base de datos para adaptarlo a la información proveniente de las nuevas fuentes. El objetivo es unificar la información de todas las licitaciones en un único modelo en la base de datos.

Así mismo, se debe adaptar el actual Back End de la plataforma para que, en caso de que una misma licitación aparezca en dos fuentes distintas, este debe ser capaz de reconocer que son la misma licitación y, por tanto, debe actualizar en la base de datos la licitación ya existente con la nueva información en vez de crear otro documento.

En referencia a la sección de superadministrador, en esta sección se muestra al usuario con capacidad para acceder a ella el estado de los rastreadores y los posibles errores ocurridos en estos mismos o en la plataforma. Dichas notificaciones aportan el máximo de información posible para encontrar rápidamente el error y arreglarlo, pero en ningún caso proporcionan una solución al problema.

Para el apartado de administración, el objetivo es mostrar en pantalla información relevante de la plataforma a los administradores de la plataforma, permitiéndoles así tomar decisiones basándose en esta información y hacer las gestiones que crean necesarias. La información en esta pantalla no será personalizable para cada usuario, todos recibirán en pantalla las mismas métricas y la misma información de la plataforma.

4. Análisis de referentes

Para este proyecto, se analizan diversos referentes que ofrecen características y funcionalidades de interés. Los referentes proporcionan información valiosa sobre las mejores prácticas y enfoques utilizados en el campo de la recopilación de licitaciones y la gestión de datos. A continuación, se presentan algunos de los referentes analizados:

- TED (Tenders Electronic Daily): Se realiza un análisis de la plataforma de licitaciones electrónicas de la Unión Europea. Se estudia cómo se publica y presenta la información de licitaciones en TED, incluyendo la estructura de datos, los formatos de presentación y las actualizaciones en tiempo real. También se realiza para el resto de plataformas, pero esta es especialmente relevante por no ser una plataforma de publicación de licitaciones, sino que es una plataforma que recopila licitaciones de toda Europa.
- Herramientas de seguimiento de errores en tiempo real: Se investigan herramientas como Jira, Bugzilla y Trello, que permiten registrar y visualizar los errores que ocurren en el sistema de forma instantánea. Se analiza cómo se gestionan los errores, cómo se notifican y cómo se realiza su seguimiento.
- Paneles de control y dashboards de métricas: Se examinan diferentes herramientas utilizadas en aplicaciones web para mostrar información relevante a los administradores. Se estudia el diseño y la presentación de paneles de control, así como las métricas más utilizadas en el contexto de aplicaciones similares.
- Integración de API de terceros: Se investiga cómo se integran las API de Twitter, Mixpanel, Google y LinkedIn en diversas aplicaciones. Se examina la documentación oficial de estas API y se exploran casos de uso relevantes y mejores prácticas para garantizar una integración adecuada.

Este análisis de referentes permite obtener información valiosa sobre el funcionamiento y las características clave de las plataformas y herramientas relacionadas con el proyecto. Estos referentes sirven como base para diseñar y desarrollar una aplicación sólida que cumpla con los objetivos.

Además de los referentes previamente mencionados, es relevante destacar otros proyectos que han utilizado tecnologías similares a las planteadas en el desarrollo de esta aplicación. Estos proyectos brindan ejemplos prácticos de implementaciones exitosas y enfoques que podrían ser relevantes para el proyecto actual.

1. En el proyecto titulado "Shipeer: Desarrollo de un servicio de paquetería colaborativo con Node.js" realizado por Juan Antonio Arenas Tomás en el año 2015 [1], se lleva a cabo el desarrollo de un servicio web utilizando Node.js. Este proyecto se enfoca en crear un sistema de soporte para facilitar un servicio de paquetería colaborativo. Aunque el ámbito y el enfoque de este proyecto difieren del presente, comparten el uso de Node.js como tecnología principal. El análisis de este proyecto proporciona ideas valiosas sobre la estructuración y desarrollo de la aplicación actual, adaptando conceptos y principios de servicio colaborativo a la recopilación de licitaciones.
2. Por otro lado, el trabajo realizado por Javier Santos Sánchez en el año 2022, titulado "Web Scraping y microservicios" [2], también es relevante para el desarrollo actual. En este proyecto se aborda el *web scraping* y la extracción de información de páginas web relacionadas con negocios electrónicos. Aunque el enfoque es diferente, este proyecto emplea técnicas similares de extracción de datos y desarrollo de rastreadores. El estudio detallado de este trabajo proporciona ideas valiosas sobre cómo implementar de manera eficiente los rastreadores de licitaciones propuestos en el proyecto actual, adaptando las técnicas de *web scraping* y los enfoques de microservicios al contexto de recopilación de licitaciones.

Estos proyectos previos demuestran la viabilidad y la aplicabilidad de las tecnologías propuestas en el proyecto de desarrollo de la aplicación. Al analizar y aprender de estos referentes, se obtienen conocimientos valiosos que ayudarn a guiar el diseño y la implementación del proyecto actual.

5. Metodología

Para este proyecto, se utiliza una metodología de trabajo de tipo SCRUM. Cada sprint tiene una duración aproximada de dos semanas en las que se intentan completar las tareas programadas. Al final de cada sprint, se realiza una reunión con el equipo de Tendios para revisar el estado del sprint actual, valorar los objetivos cumplidos y los que aún no han sido terminados, y definir los próximos objetivos para el siguiente sprint.

El editor de código escogido tanto para la programación de los diferentes scripts de Python como para adaptar el Back End y Front End a las nuevas funcionalidades es Visual Studio Code.

La estructura del software respeta la arquitectura modelo vista controlador ya implementada en la base proporcionada por la empresa colaboradora. En esta arquitectura, se divide el código según sus diferentes responsabilidades. En el apartado de modelos, se agrupa el código encargado de trabajar con los datos, en el de vistas, el código responsable de la representación visual de la aplicación, y en la sección de controladores, el código que junta las vistas y los modelos, donde también se implementan las funcionalidades de la aplicación.

El proyecto se adapta a la empresa colaboradora en cuanto al control de versiones, frameworks, entorno de ejecución y base de datos. Se utiliza el GitHub ya creado por la empresa para el control de versiones, Node.js para el desarrollo de Back End, Vue.js para Front End y MongoDB como base de datos.

El proyecto se ejecuta en tres servidores cloud, la información de estos es la siguiente:

- Servidor CX11 para la ejecución de la aplicación:
    - CPU: 1 Intel.
    - RAM: 2 GB.
    - Espacio de disco: 20 GB.
- Servidor CPX11 para la base de datos:
    - CPU: 2 AMD.
    - RAM: 2 GB.
    - Espacio de disco: 40 GB.

- Servidor CX21 para los rastreadores de licitaciones:
    - CPU: 2 Intel.
    - RAM: 4 GB.
    - Espacio de disco: 40 GB.

A continuación, se procede a explicar las principales características del método scrum que se utiliza en el proyecto.

5.1 Scrum

Scrum es una metodología ágil para la gestión de proyectos de software y desarrollo de productos que se enfoca en un enfoque iterativo e incremental.

Scrum establece un marco para la colaboración, la planificación y la revisión en equipo para lograr un desarrollo ágil y eficiente de productos de alta calidad. Esta metodología se basa en el trabajo en equipo y la retroalimentación continua, y está diseñada para acomodarse a los cambios y a los requisitos cambiantes del mercado y del usuario.

5.1.1 Roles

Dentro de la metodología scrum, se diferencian tres roles [7][8]:

1. Scrum Master: es el encargado de facilitar el proceso Scrum y asegurar que se cumplan las prácticas y valores ágiles.
2. Product Owner: es el responsable de definir y priorizar los requisitos del producto y de comunicar el objetivo y el alcance del proyecto al equipo de desarrollo.
3. Equipo de Desarrollo: es el grupo encargado de desarrollar el producto y de asegurarse de que se cumplan los requisitos y objetivos del proyecto.

5.1.2 Procesos en Scrum

1. Sprint: una iteración de trabajo que dura entre 1 y 4 semanas. Durante un Sprint, el equipo de desarrollo completa un trabajo definido y entregable.
2. Sprint Planning: una reunión en la que el equipo de desarrollo y el Product Owner planifican el trabajo para el próximo Sprint.
3. Daily Scrum: una reunión diaria de seguimiento en la que el equipo de desarrollo discute el progreso y el plan para el día.

4. Sprint Review: una reunión al final del Sprint en la que el equipo de desarrollo demuestra el trabajo completado y recibe retroalimentación del Product Owner y otros interesados.
5. Sprint Retrospective: una reunión en la que el equipo de desarrollo reflexiona sobre el Sprint anterior y discute cómo mejorar en el futuro.

A continuación, se presenta un esquema del funcionamiento de un sprint (Fuente: [9])

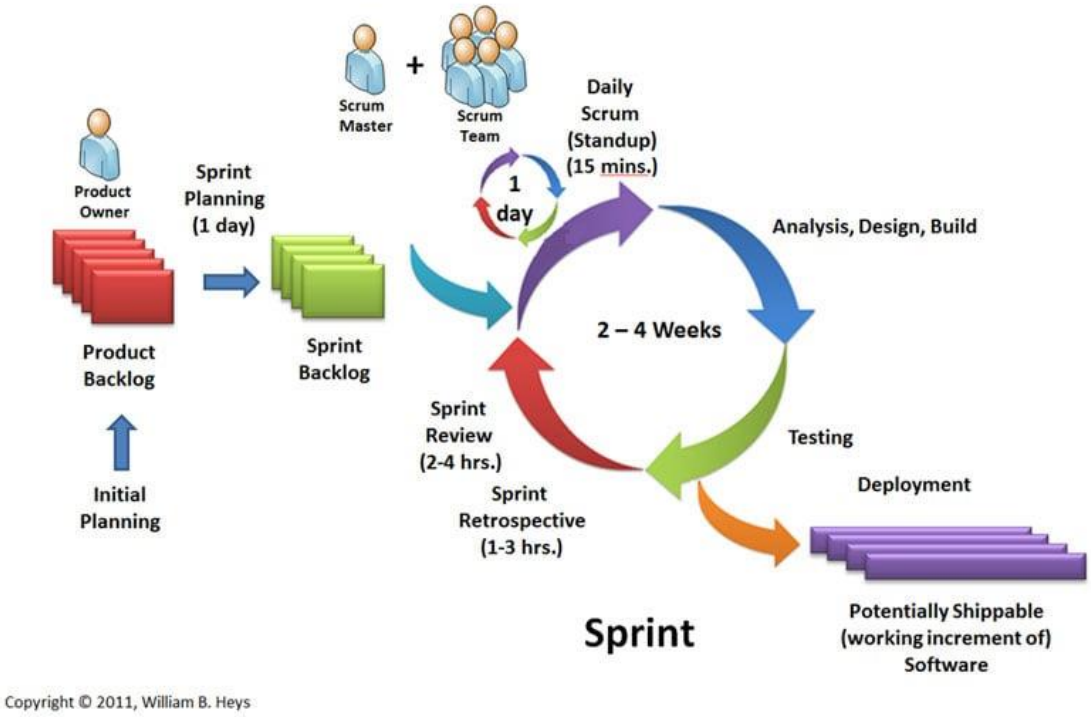

Figura 5.1 Esquema de un Sprint de SCRUM

Scrum es una metodología popular y ampliamente utilizada que ha demostrado ser efectiva para la gestión de proyectos de software y desarrollo de productos. Al enfocarse en el trabajo en equipo y la retroalimentación continua, Scrum permite a los equipos ser más ágiles y responder rápidamente a los cambios en el mercado y en los requisitos del usuario.

6. Definición de requerimientos funcionales y tecnológicos

A continuación, se detalla punto por punto los requerimientos funcionales y tecnológicos que abarcan este proyecto.

6.1 Requerimientos funcionales

- Rastrear cada día todas las licitaciones publicadas con el máximo de información posible en las plataformas de:
    - Boletín Oficial del Estado (BOE).
    - *Diário da República* (DRE).
    - Diario Oficial de la Unión Europea (DOUE).
    - Sección de Contratos Menores de la Plataforma de Contratación del Sector Público.
- Mostrar errores que ocurren en tiempo real en la plataforma en la sección de superadministrador.
- Mostrar el estado de los rastreadores en la sección de superadministrador.
- Mostrar posibles errores ocurridos en los rastreadores en la sección de superadministrador.
- Mostrar la información de las cuentas creadas en la plataforma en la sección de administrador.
- Mostrar la información relativa a los usuarios registrados en la plataforma en la sección de administrador.
- Mostrar las suscripciones activas de la plataforma en la sección de administrador.
- Mostrar un dashboard en la sección de administrador con métricas de interés para este grupo de usuarios.
- Cada mañana a las 09:00, se debe crear una publicación automática en Twitter a través de su API con el resumen de las publicaciones diarias.
- A través de la API de Mixpanel, se deben obtener métricas de la plataforma.
- Con la implementación de la API de Google, los usuarios son capaces de acceder a la plataforma mediante su cuenta de Google.

- Los usuarios son capaces de iniciar sesión a través de su perfil de LinkedIn, esto realizado a través de la API de dicha plataforma.

6.2 Requerimientos tecnológicos

- Utilizar Python para el desarrollo de los rastreadores de licitaciones.
- Utilizar BeautifulSoup4 para realizar el web scraping de las diferentes páginas web en las que se publican las licitaciones.
- Utilizar JavaScript como lenguaje de programación para Back End y Front End.
- Utilizar Visual Studio Code tanto para programar la parte de Python como modificar el Back End y el Front End.
- Utilizar Node.js para el desarrollo del Back End.
- Utilizar Vue.js para el desarrollo del Front End.
- Utilizar MongoDB como base de datos del proyecto.
- Utilizar Mongoose para la conexión entre la base de datos y Back End y trabajar con los documentos.
- Utilizar API de herramientas de terceros para su implementación dentro de la plataforma.
- Alquilar tres servidores, uno para la base de datos, otro para ejecutar la aplicación, y el último para ejecutar los scripts de los rastreadores de licitaciones.

7. Desarrollo

Para el desarrollo del proyecto en cuestión, se han identificado tres componentes.

- En primer lugar, los rastreadores de licitaciones, los cuales tienen la responsabilidad de recopilar información de diferentes sitios web y enviarla al Back End para su posterior tratamiento y almacenamiento en una base de datos.
- En segundo lugar, el Back End, que funciona como una API dentro de la aplicación y se encarga tanto de almacenar información en la base de datos como de extraerla para proporcionarla a los usuarios que la necesiten. Además, es responsable de todas las funcionalidades de la aplicación.
- En tercer lugar, se encuentra el Front End, la única parte visible para el usuario final, que consta de diferentes pantallas accesibles al usuario. La cantidad de pantallas disponibles depende de la categoría del usuario.

A continuación, se describen en detalle el desarrollo de cada uno de los componentes mencionados anteriormente, así como otros aspectos importantes del proyecto.

7.1 Herramientas y tecnologías

Durante el desarrollo del proyecto, se emplean diversas herramientas y tecnologías que desempeñan un papel fundamental en su implementación y funcionamiento. A continuación, se detallan las principales herramientas y tecnologías utilizadas:

- Visual Studio Code: Se utiliza el entorno de desarrollo integrado (IDE) Visual Studio Code como el principal editor de código para el proyecto. Gracias a su amplia gama de extensiones y su interfaz intuitiva, se facilita la codificación y la depuración del código.
- Postman: La herramienta Postman se utiliza para realizar pruebas y validar la API desarrollada. Postman permite enviar solicitudes HTTP y evaluar las respuestas recibidas, lo que facilita el proceso de desarrollo y depuración de las funciones de la API.
- MongoDB: Se utiliza MongoDB como la base de datos principal del proyecto. MongoDB, una base de datos NoSQL orientada a documentos, permite almacenar y recuperar datos de manera eficiente y flexible.

- Mongoose: Mongoose, una biblioteca de Node.js para modelado de objetos MongoDB. Se utiliza para simplificar la interacción con la base de datos MongoDB desde el backend. Mongoose proporciona una capa de abstracción para definir esquemas y realizar consultas a la base de datos.
- MongoDB Compass: Como interfaz gráfica para MongoDB, MongoDB Compass se utiliza para administrar y visualizar los datos almacenados en la base de datos. Esta herramienta proporciona una forma intuitiva de interactuar con la base de datos y realizar consultas.
- JavaScript: El lenguaje de programación JavaScript se utiliza en el proyecto para desarrollar funcionalidades interactivas en el lado del cliente. JavaScript permite agregar dinamismo y mejorar la experiencia del usuario en la interfaz de la plataforma.
- Python: Python se utiliza como lenguaje de programación principal en el proyecto. Se emplea para desarrollar los rastreadores de licitaciones y otras funcionalidades relacionadas con el procesamiento de datos y la lógica del proyecto.
- Node.js: Se utiliza Node.js, un entorno de ejecución de JavaScript del lado del servidor, para implementar el Back End del proyecto. Node.js proporciona un entorno eficiente y escalable para ejecutar el servidor y administrar las solicitudes entrantes.
- Vue.js: Vue.js es el framework de JavaScript utilizado para el desarrollo del Front End de la plataforma. Gracias a su enfoque reactivo y su capacidad para crear interfaces de usuario interactivas, Vue.js permite construir una interfaz de usuario dinámica y fácil de mantener.

Estas herramientas y tecnologías se seleccionan cuidadosamente y se utilizan de manera integrada para garantizar el desarrollo eficiente y exitoso del proyecto, cumpliendo con los objetivos establecidos

7.2 Cambios en los objetivos

El presente trabajo de fin de grado se basa en un proyecto que tiene sus inicios como una colaboración entre el autor de este mismo proyecto y la empresa ELONIAL TECHNOLOGIES, SL. Dicha empresa es la propietaria de la aplicación sobre la cual se centra el proyecto. Sin embargo, debido a la transición del autor a otra empresa, dicha colaboración se interrumpe y se decide continuar el desarrollo del proyecto de manera independiente, trabajando en el entorno local.

Esta nueva fase del proyecto implica la necesidad de replantear algunos de los objetivos originales, con el fin de adaptarlos a la nueva realidad y garantizar un enfoque efectivo. A continuación, se detallan los cambios que se llevan a cabo:

1. Se elimina la integración de herramientas de terceros: Uno de los objetivos originales consiste en integrar herramientas de terceros a través de sus respectivas API, como Twitter, Mixpanel, Google y LinkedIn. Sin embargo, considerando la nueva dirección y las limitaciones de recursos, se decide eliminar este objetivo. Esta decisión permite concentrar los esfuerzos en otras áreas del proyecto que se consideran más relevantes para su éxito.
2. Se crea una pantalla de licitaciones recientes: Como parte de la reevaluación de objetivos, se identifica la oportunidad de añadir valor al proyecto mediante la creación de una pantalla que muestre las licitaciones más recientes. Esta adición proporciona a los usuarios una visión rápida y actualizada de las oportunidades disponibles. La implementación de esta nueva funcionalidad requiere el uso de tecnologías como Python y Vue.js.
3. Se crea una pantalla de inicio de sesión (login): Otro cambio importante es la inclusión de una pantalla de inicio de sesión. Esta funcionalidad permite a los usuarios acceder de manera segura a la plataforma, garantizando la protección de la información y brindando una experiencia personalizada. La implementación de esta pantalla se lleva a cabo utilizando JavaScript y Vue.js.
4. Se crea una pantalla de registro (sign up): Además del inicio de sesión, se considera esencial proporcionar a los usuarios una forma sencilla de registrarse en la plataforma. La creación de una pantalla de registro permite capturar la información necesaria y garantizar que los usuarios tengan una cuenta activa para aprovechar

todas las funcionalidades. La implementación de esta pantalla también requiere el uso de JavaScript y Vue.js.

5. Se consolida las páginas de administrador y superadministrador: Considerando la naturaleza del proyecto y las funcionalidades previstas, se toma la decisión de combinar las páginas de administrador y superadministrador en una sola página. Esta consolidación permite optimizar el flujo de trabajo y brindar una interfaz intuitiva y coherente. Dependiendo del rol del usuario que accede, el contenido de la página se adapta dinámicamente. El desarrollo de esta funcionalidad se realiza utilizando Vue.js y sus capacidades de enrutamiento y gestión de roles.

Estos cambios en los objetivos originales del proyecto reflejan una adaptación necesaria ante la nueva situación, maximizando los recursos disponibles y priorizando las funcionalidades clave para el éxito del proyecto. La reevaluación de objetivos y la implementación de los cambios mencionados proporcionan una base sólida para avanzar y lograr los resultados deseados en esta etapa del proyecto.

7.3 Rastreadores

Los rastreadores de licitaciones, previamente mencionados, son scripts programados en Python diseñados para recolectar información sobre licitaciones de las principales fuentes donde se publican. Para lograr esto, se utilizan las librerías BeautifulSoup4 y Selenium. La información que se busca recopilar para cada licitación incluye:

- Url de la fuente: La dirección web de donde se ha extraído la información.
- Enlace a la licitación: Url en la cual se puede encontrar toda la información de la licitación.
- Expediente: Combinación de números, letras y caracteres que identifican la licitación.
- Órgano de contratación: Órgano el cual ha publicado la licitación.
- Estado de la licitación: Indica el estado actual de la licitación, pudiendo ser “publicada”, “adjudicada”, entre otros.
- Objeto del contrato: Equivalente al título de la licitación, donde se explica en breve de qué se trata.
- Presupuesto base de la licitación sin impuestos.
- Valor estimado del contrato.

- Tipo de contrato: Varía de acuerdo a la naturaleza y objeto de la contratación, siendo los más importantes los contratos de obra, suministros y servicio.
- Códigos CPV: Son códigos que permiten identificar y clasificar la actividad económica que involucra la licitación.
- Lugar de ejecución: El espacio geográfico donde se llevará a cabo la actividad descrita en la licitación.
- Procedimiento de contratación: Indica el tipo de procedimiento de adjudicación de la licitación.
- Fecha fin de presentación de oferta.
- Fecha de publicación del expediente.
- Fecha de actualización del expediente: Consiste en la fecha de la última actualización de la licitación.
- Resultado: En caso de que la licitación ya esté cerrada, indica con qué resultado se ha cerrado.
- Adjudicatario: Indica que órgano ha sido contratado.
- Número de licitadores: Indica cuantos órganos han presentado sus ofertas para la licitación.
- Importe de la adjudicación.
- Documentos.

Se utiliza la librería Selenium para que el script simule el comportamiento humano al navegar por la página web y así poder acceder a los datos necesarios. Por su parte, la librería BeautifulSoup4 recopila la información de la página que contiene los datos para enviarla posteriormente al Back End.

Los rastreadores se comunican con el Back End de la aplicación a través de los *endpoints* que éste provee para guardar las licitaciones en la base de datos. Después de recopilar toda la información de una licitación, el rastreador la agrega en un objeto JSON que se envía al Back End. Este objeto JSON sigue el siguiente esquema:

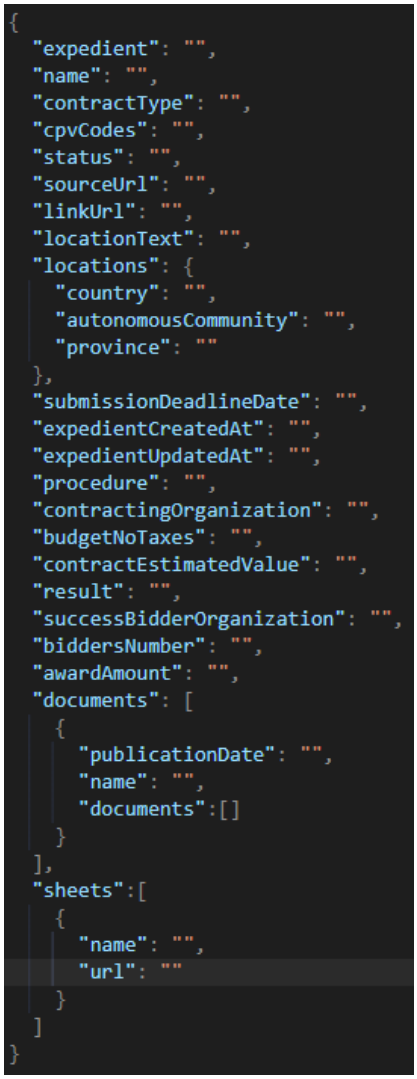

Figura 7.1 Esquema JSON enviado a Back End

En la planificación inicial del proyecto, se estableció el objetivo de crear cuatro scripts para cada plataforma de licitaciones: single, short, long y all. Sin embargo, este objetivo ha sufrido modificaciones en función de las dificultades o facilidades encontradas en cada plataforma para obtener la información de las licitaciones. A continuación, se presenta el desarrollo llevado a cabo para cada fuente.

7.3.1 Plataforma de Contratación del Sector Público

Este es el rastreador ya funcional incluido dentro de la base del proyecto. Pese a ello, se ha encontrado que la plataforma pone a disposición del público diferentes archivos comprimidos, uno por cada año, que contienen todas las licitaciones publicadas dentro de la plataforma en archivos atom.

Para aprovechar esto, se ha creado un nuevo script que lee todos los archivos dentro de la carpeta comprimida y guarda en base de datos todas las licitaciones contenidas en los archivos.

7.3.2 Contratos Menores

El rastreador de licitaciones de la sección de contratos menores en la Plataforma de Contrataciones del Sector Público cuenta con los scripts de single, short y long. Sin embargo, no se ha desarrollado el script all debido a que actualmente la fuente tiene cerca de 1.900.000 licitaciones y el programa tarda aproximadamente 4 segundos en recopilar la información de una licitación. Dicho esto, se estima que la ejecución completa de este script tardaría alrededor de 88 días, tiempo que se ha considerado excesivo para los objetivos del proyecto.

7.3.3 Boletín Oficial del Estado

El rastreador correspondiente a la fuente del BOE tiene como objetivo recopilar la información de las licitaciones publicadas en dicha plataforma. Se ha desarrollado cada uno de los scripts previstos en la planificación, pero a diferencia de los demás, se ha prescindido de la librería Selenium.

La razón es que, además de mostrar la información de las licitaciones en la misma página web como el resto de fuentes, el BOE también sube la información de cada licitación en formato XML.

Para acceder al XML de cada licitación, es necesario primero acceder a otro archivo XML que contiene todas las licitaciones publicadas en un día. La URL de este XML está vinculada a la fecha de publicación, siguiendo el siguiente formato:

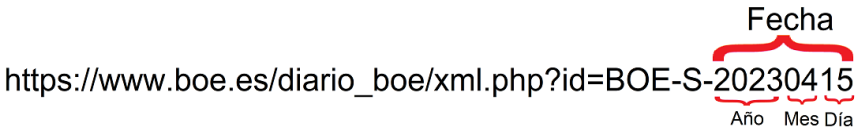

Figura 7.2 Formato de la URL

Tras acceder al documento XML de las licitaciones publicadas en un día determinado, se guarda el contenido de la página mediante la librería BeautifulSoup4. Posteriormente, el script procede a buscar la sección "Contratación del Sector Público" dentro del contenido guardado previamente. En esta sección, las licitaciones están organizadas por departamentos, siguiendo la siguiente estructura:

```
<item id="BOE-B-2023-10568">
    <titulo>Título</titulo>
    <urlPdf szBytes="170076" szKBytes="166" numPag="3">/boe/dias/2023/04/13/pdfs/BOE-B-2023-10568.pdf</urlPdf>
    <urlHtm>/diario_boe/txt.php?id=BOE-B-2023-10568</urlHtm>
    <urlXml>/diario_boe/xml.php?id=BOE-B-2023-10568</urlXml>
</item>
```

Figura 7.3 Estructura de la licitación

Para cada departamento, el script busca todas las licitaciones y, para cada una de ellas, accede al URL del archivo XML que contiene los datos de la licitación. Luego, recopila la información y envía los datos a Back End.

7.3.4 Diário da República

En relación al rastreador de la fuente portuguesa, se ha creado el script single y short, que se encarga de recopilar la información de las licitaciones publicadas en un día. No obstante, no se ha desarrollado el script de tipo long o all debido a la complejidad que presenta la página web para navegar entre las licitaciones de distintos días.

Otro inconveniente que ha surgido durante el proceso de desarrollo de estos scripts, es que esta fuente carga sus páginas de manera dinámica. Esto significa que, en primer lugar, se carga una página vacía y luego se carga la información de la licitación mediante JavaScript. Al intentar obtener información con la librería BeautifulSoup4, esta obtiene una página vacía, ya que toma la primera página cargada.

Para solucionar esta situación, se ha incorporado una función que espera a que todo el contenido dinámico de la página sea cargado antes de recolectar la información con la librería.

Desgraciadamente, durante el mes de marzo se realizó una modificación en la estructura de la página que ha hecho que este rastreador deje de funcionar y no sea posible adaptarlo al nuevo formato, por lo que la aplicación final no es capaz de recoger las licitaciones provenientes de esta fuente.

7.3.5 Diario Oficial de la Unión Europea

Este rastreador tiene la tarea de recopilar la información de las licitaciones que se publican en el Diario Oficial de la Unión Europea. Sin embargo, sólo se han desarrollado los scripts single y short debido a la gran cantidad de información que se publica en esta fuente y los tiempos de ejecución demasiado largos que genera para el alcance del proyecto.

El principal problema encontrado durante el proceso de rastreo de la información en esta plataforma ha sido la gestión de memoria. El script utiliza la librería Selenium y crea una nueva pestaña de navegador por cada licitación a la que accede. Dado que se publican muchas licitaciones en esta fuente, esto hace que la memoria del dispositivo se sature después de acceder a muchas licitaciones, debido al gran número de pestañas de navegador que se abren simultáneamente.

Para solucionar este problema, se ha tenido que adaptar el funcionamiento del script para que, una vez consultada una pestaña abierta, el programa cierre dicha pestaña y evite la apertura infinita de nuevas pestañas hasta alcanzar el límite de memoria.

7.4 Base de Datos

El proyecto utiliza MongoDB como base de datos para almacenar toda la información, según los requerimientos tecnológicos establecidos. A pesar de que esta base de datos admite documentos con esquemas diferentes en una misma colección, se ha desarrollado un esquema para todas las colecciones necesarias del proyecto, con el fin de prevenir futuros errores. Para garantizar que todos los documentos dentro de una misma colección sigan el mismo esquema, se implementan estos esquemas en Back End mediante Mongoose. El esquema diseñado para las licitaciones es el siguiente:

Tender	
_id	{ ObjectId }
slug	{ String }
expedient	{ String }
name	{ String }
contractingOrganization	{ ObjectId }
successBidderOrganization	{ ObjectId }
documents	{ Array }
sheets	{ Array }
sources	{ Array }
contractType	{ String }
cpvCodes	{ Array }
status	{ String }
procedure	{ String }
locationText	{ String }
locations	{ Map }
country	{ ObjectId }
currency	{ ObjectId }
submissionDeadlineDate	{ Date }
expedientCreatedAt	{ Date }
expedientUpdatedAt	{ Date }
budgesNoTaxes	{ Number }
contractEstimatedValue	{ Number }
result	{ String }
biddersNumber	{ Number }
awardAmount	{ Number }
isAdjudication	{ Boolean }
isMinorContract	{ Boolean }
createdAt	{ Date }
updatedAt	{ Date }
deletedAt	{ Date }

Figura 7.4 Esquema de bases de datos para Licitación

Dentro de este esquema se destacan los siguientes datos:

- Los campos obligatorios incluyen "id", "expedient", "name", "contractType", "status" y "procedure". El único campo que no se puede repetir entre licitaciones es "slug". Aunque el campo "expedient" se utiliza para identificar las licitaciones, se han dado casos en los que dos licitaciones diferentes tenían el mismo expediente. Este problema se soluciona posteriormente en Back End.
- Los campos "expedientCreatedAt" y "expedientUpdatedAt" indican la fecha de creación y la última fecha de actualización encontrada dentro de la página web,

mientras que los campos "createdAt" y "updatedAt" indican la fecha de creación y la última fecha de actualización dentro de la base de datos.

- El campo "locationText" contiene el texto extraído de la página web en relación al lugar de ejecución de la licitación, por ejemplo "España - Albacete - Albacete". Por otro lado, el campo "locations" contiene un objeto de tipo mapa con esta misma información. Un ejemplo de esto sería el siguiente:

```
country: "ES"
autonomousCommunity: "Castilla-La Mancha"
province: "Albacete"
```

Figura 7.5 Ejemplo de campo "locations"

El motivo por el cual se ha implementado el campo "locations" junto con el campo "locationText" es que, en el futuro, se puede añadir un filtrado de licitaciones por localizaciones. En caso de solo tener el campo "locationText", esta funcionalidad no sería posible.

Además, dentro del esquema de licitaciones, hay campos que hacen referencia a objetos de otros esquemas. Los campos "contractingOrganization" y "successBidderOrganization" siguen el esquema de la colección "Organization", el cual es el siguiente:

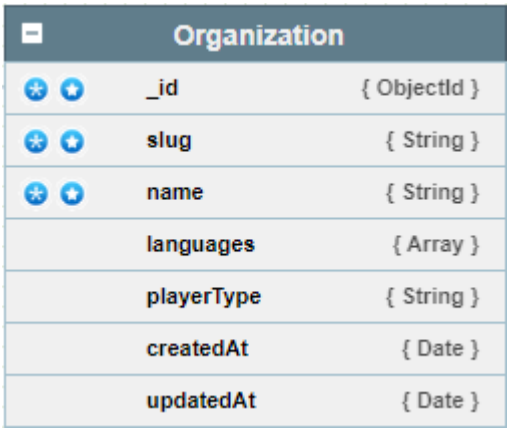

Figura 7.6 Esquema de bases de datos para Organización

Dentro de este esquema, se destaca el campo "playerType", el cual se emplea para distinguir entre las organizaciones contratantes y las licitadoras.

Además, hay otros campos que siguen el esquema de otra colección, como "country", "cpvCodes" y "currency". Los esquemas correspondientes son los siguientes:

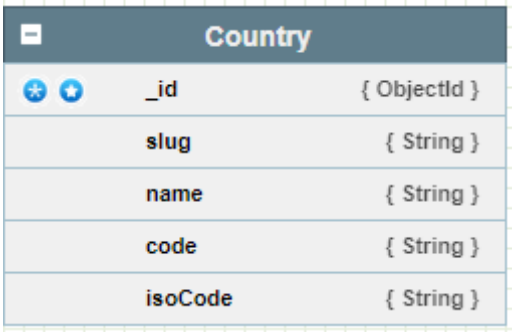

Figura 7.7 Esquema de bases de datos para País

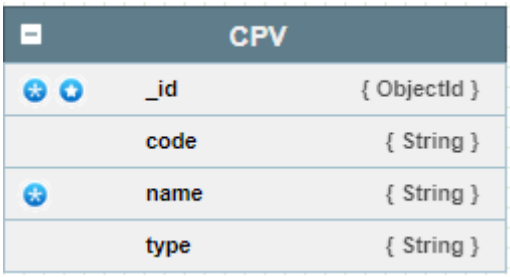

Figura 7.8 Esquema de bases de datos para Códigos CPV

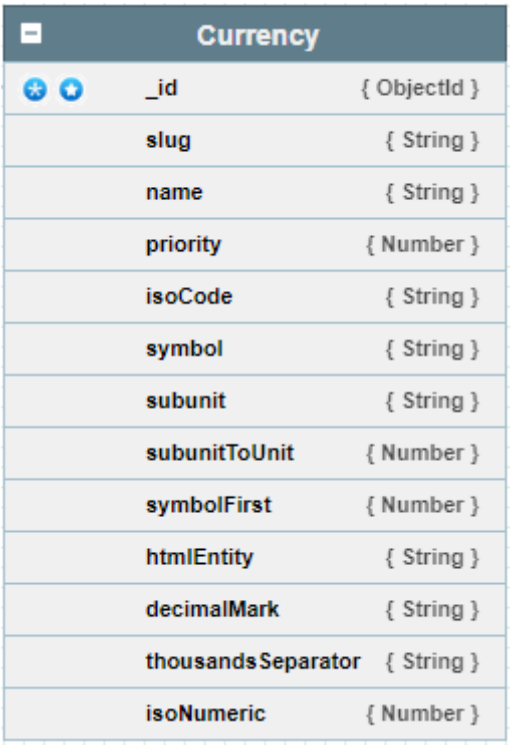

Figura 7.9 Esquema de bases de datos para Divisa

Existen otros esquemas adicionales en la base de datos que no están relacionados con el esquema de licitaciones. Entre ellos, se encuentran los esquemas de “User”, “Error” y

“Runnable”. El esquema de usuarios es sencillo y permite el inicio de sesión mediante el correo electrónico y la contraseña asociados al documento del usuario. El esquema se muestra a continuación:

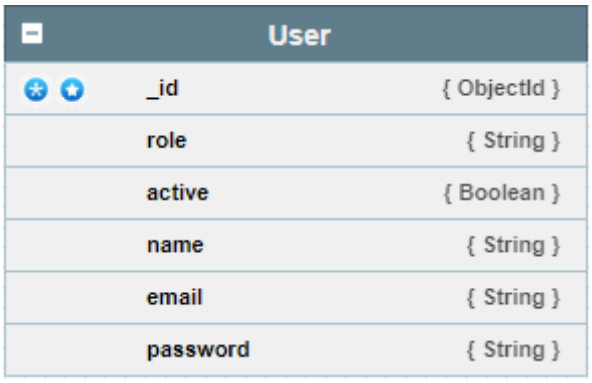

Figura 7.10 Esquema de bases de datos para Usuario

El esquema de "Error" almacena los documentos generados cuando ocurre un error y guarda la información del mismo. Este esquema se utiliza para extraer los documentos de la base de datos en un momento posterior y mostrar en la pantalla del superadministrador los últimos errores ocurridos. El esquema correspondiente es el siguiente:

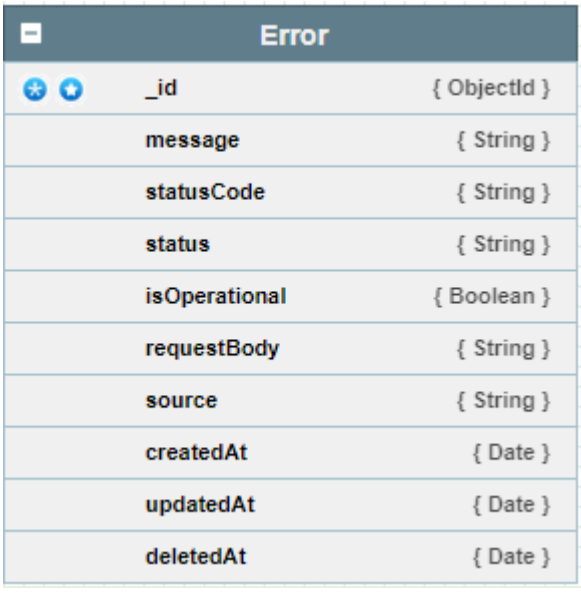

Figura 7.11 Esquema de bases de datos para Error

Se destaca el campo "isOperational", que se utiliza para indicar si el error dentro de la plataforma se origina por el código creado o por una fuente externa.

Por último, se ha creado un esquema para almacenar documentos que indiquen qué rastreadores se han ejecutado. El esquema correspondiente es el siguiente:

Figura 7.12 Esquema de bases de datos para "Runnable"

La información relativa al script ejecutado, como la duración de la ejecución, la cantidad de licitaciones recogidas, la fuente y otros datos, se almacenan en estos documentos.

7.5 Back End

Tal y como se indica en el primer punto de este proyecto, se parte de un Back End funcional que incluye un modelo de licitación y un rastreador en funcionamiento. Esta parte de la aplicación sigue el patrón Modelo-Vista-Controlador, con algunas modificaciones.

Originalmente, en este patrón, los modelos representan la capa de datos de la aplicación y contienen la lógica de negocio y las interacciones con la base de datos. Los controladores se encargan de manejar las solicitudes del cliente, interactuar con los modelos y preparar la respuesta para la vista y las vistas son responsables de mostrar la información al usuario final.

En el proyecto, se aplica una variación en la estructura y responsabilidades de cada componente en comparación con el planteamiento original del patrón. En este caso, se implementan controladores, modelos y rutas, que se asemejan a la implementación típica del patrón modelo-vista-controlador, donde los controladores actúan como el componente del

controlador, los modelos como el componente del modelo y las rutas proporcionan el enrutamiento y el mapeo de las solicitudes.

Además, en este Back End se encuentra una mayor separación entre el controlador y la vista. En lugar de generar vistas HTML directamente, el Back End está diseñado para enviar respuestas en formato JSON, que luego son consumidas por el Front End para su presentación.

Asimismo, este Back End actúa como una API con diferentes endpoints que el Front End consume para recoger la información necesaria, y otros que los rastreadores utilizan para enviar la información de las licitaciones y guardarlas en la base de datos.

Seguidamente, se detalla el desarrollo de los puntos más importantes relacionados con el Back End del proyecto.

7.5.1 Endpoints

A continuación, se detallan todos los endpoints que contiene el Back End para ser utilizados por el resto de la aplicación. Estos se dividen en endpoints relacionados con los códigos CPV, errores, organizaciones, runnables, licitaciones y usuarios. En todas las direcciones especificadas en este apartado, "URL" se refiere a "http://127.0.0.1:3000/v1" o su equivalente "http://localhost:3000/v1".

CPV:

- **Obtener todos los códigos CPV:**
  - Método: GET
  - URL: '/cpvs'
  - Descripción: Devuelve todos los códigos CPV almacenados en la base de datos.
- **Obtener un código CPV específico:**
  - Método: GET
  - URL: '/cpvs/{id}'
  - Descripción: Devuelve un único código CPV identificado por el parámetro {id}.

- **Crear un código CPV:**
  - Método: POST
  - URL: ‘/cpvs’
  - Descripción: Crea un nuevo código CPV utilizando la información proporcionada en el cuerpo de la solicitud en formato JSON.
- **Actualizar un código CPV:**
  - Método: PATCH
  - URL: ‘/cpvs/{id}’
  - Descripción: Actualiza un código CPV específico identificado por el parámetro {id}, utilizando la nueva información proporcionada en el cuerpo de la solicitud en formato JSON.
- **Eliminar un código CPV:**
  - Método: DELETE
  - URL: ‘/cpvs/{id}’
  - Descripción: Elimina de la base de datos el código CPV identificado por el parámetro {id}.

Errores:

- **Obtener todos los errores:**
  - Método: GET
  - URL: ‘/errors’
  - Descripción: Devuelve todos los errores almacenados en la base de datos.
- **Obtener un error específico:**
  - Método: GET
  - URL: ‘/errors/{id}’
  - Descripción: Devuelve un documento de error específico identificado por el parámetro {id}.

Organizaciones:

- **Obtener todas las organizaciones:**
  - Método: GET
  - URL: ‘/organizations’
  - Descripción: Devuelve todas las organizaciones almacenadas en la base de datos.
- **Obtener una organización específica:**
  - Método: GET
  - URL: ‘/organizations/{id}’
  - Descripción: Devuelve un documento de organización específico identificado por el parámetro {id}.
- **Crear una nueva organización:**
  - Método: POST
  - URL: ‘/organizations’
  - Descripción: Crea un nuevo documento de organización utilizando la información proporcionada en el cuerpo de la solicitud en formato JSON.
- **Actualizar una organización:**
  - Método: PATCH
  - URL: ‘/organizations/{id}’
  - Descripción: Actualiza un documento de organización específico identificado por el parámetro {id}, utilizando la nueva información proporcionada en el cuerpo de la solicitud en formato JSON.
- **Eliminar una organización:**
  - Método: DELETE
  - URL: ‘/organizations/{id}’
  - Descripción: Elimina de la base de datos el documento de organización identificado por el parámetro {id}.
- **Obtener organizaciones de tipo "public-contracting-institution":**
  - Método: GET
  - URL: ‘/organizations/publicContractingInstitutions’
  - Descripción: Devuelve todos los documentos de organizaciones que tengan el valor "public-contracting-institution" en el campo "playerType".

- **Obtener organizaciones de tipo "bidder":**
    - Método: GET
    - URL: ‘/organizations/bidders’
    - Descripción: Devuelve todos los documentos de organizaciones que tengan el valor "bidder" en el campo "playerType".

Runnables:

- **Obtener todos los runnables:**
    - Método: GET
    - URL: ‘/runnables’
    - Descripción: Devuelve todos los documentos de runnables almacenados en la base de datos.
- **Obtener un runnable específico:**
    - Método: GET
    - URL: ‘/runnables/{id}’
    - Descripción: Devuelve un documento de runnable específico identificado por el parámetro {id}.
- **Crear un nuevo runnable:**
    - Método: POST
    - URL: ‘/runnables’
    - Descripción: Crea un nuevo documento de runnable utilizando la información proporcionada en el cuerpo de la solicitud.
- **Actualizar un runnable:**
    - Método: PATCH
    - URL: ‘/runnables/{id}’
    - Descripción: Actualiza un documento de runnable específico identificado por el parámetro {id}, utilizando la nueva información proporcionada en el cuerpo de la solicitud.
- **Eliminar un runnable:**
    - Método: DELETE
    - URL: ‘/runnables/{id}’
    - Descripción: Elimina de la base de datos el documento de runnable identificado por el parámetro {id}.

Licitaciones:

- **Obtener licitaciones del portal "Contrataciones del Estado":**
  - Método: GET
  - URL: ‘/tenders/source/contratacionesdelestado’
  - Descripción: Devuelve todos los documentos de licitaciones que tienen el portal "Contrataciones del Estado" como fuente.
- **Crear una licitación del portal "Contrataciones del Estado":**
  - Método: POST
  - URL: ‘/tenders/sources/contratacionesdelestado/create’
  - Descripción: Crea una nueva licitación utilizando la información proporcionada en el cuerpo de la solicitud.
- **Obtener licitaciones del portal "Contratos Menores":**
  - Método: GET
  - URL: ‘/tenders/source/contratacionesmenores’
  - Descripción: Devuelve todos los documentos de licitaciones que tienen el portal "Contratos Menores" dentro de "Contrataciones del Estado" como fuente.
- **Crear una licitación del portal "Contratos Menores":**
  - Método: POST
  - URL: ‘/tenders/sources/contratacionesmenores/create’
  - Descripción: Crea una nueva licitación utilizando la información proporcionada en el cuerpo de la solicitud.
- **Obtener licitaciones del portal "BOE":**
  - Método: GET
  - URL: ‘/tenders/source/boe’
  - Descripción: Devuelve todos los documentos de licitaciones que tienen el portal "BOE" como fuente.
- **Crear una licitación del portal "BOE":**
  - Método: POST
  - URL: ‘/tenders/sources/boe/create’
  - Descripción: Crea una nueva licitación utilizando la información proporcionada en el cuerpo de la solicitud.

- **Obtener licitaciones del portal "Diário da República":**
    - Método: GET
    - URL: ‘/tenders/source/dre’
    - Descripción: Devuelve todos los documentos de licitaciones que tienen el portal "Diário da República" como fuente.
- **Crear una licitación del portal "Diário da República":**
    - Método: POST
    - URL: ‘/tenders/sources/dre/create’
    - Descripción: Crea una nueva licitación utilizando la información proporcionada en el cuerpo de la solicitud.
- **Obtener licitaciones del portal "Tenders Electronic Daily":**
    - Método: GET
    - URL: ‘/tenders/source/ted’
    - Descripción: Devuelve todos los documentos de licitaciones que tienen el portal "Tenders Electronic Daily" como fuente.
- **Crear una licitación del portal "Tenders Electronic Daily":**
    - Método: POST
    - URL: ‘/tenders/sources/ted/create’
    - Descripción: Crea una nueva licitación utilizando la información proporcionada en el cuerpo de la solicitud.
- **Obtener todas las licitaciones:**
    - Método: GET
    - URL: ‘/tenders’
    - Descripción: Devuelve todos los documentos de licitaciones almacenados en la base de datos.
- **Obtener una licitación específica:**
    - Método: GET
    - URL: ‘/tenders/{id}’
    - Descripción: Devuelve una licitación específica identificada por el parámetro {id}.

- **Actualizar una licitación:**
    - Método: PATCH
    - URL: '/tenders/{id}'
    - Descripción: Actualiza una licitación específica identificada por el parámetro {id}, utilizando la nueva información proporcionada en el cuerpo de la solicitud.
- **Eliminar una licitación:**
    - Método: DELETE
    - URL: '/tenders/{id}'
    - Descripción: Elimina de la base de datos la licitación identificada por el parámetro {id}.
- **Obtener el número total de licitaciones:**
    - Método: GET
    - URL: '/tenders/counter'
    - Descripción: Devuelve el número total de licitaciones almacenadas en la base de datos.

Usuarios:

- **Obtener todos los usuarios:**
    - Método: GET
    - URL: '/users'
    - Descripción: Devuelve todos los usuarios almacenados en la base de datos.
- **Obtener un usuario específico:**
    - Método: GET
    - URL: '/users/{id}'
    - Descripción: Devuelve un usuario específico identificado por el parámetro {id}.
- **Obtener el número de usuarios creados en las últimas 24 horas:**
    - Método: GET
    - URL: '/users/lastUsers'
    - Descripción: Devuelve el número de usuarios creados en la base de datos en las últimas 24 horas.

- **Crear un nuevo usuario:**
  - Método: POST
  - URL: '/users/signup'
  - Descripción: Crea un nuevo usuario en la base de datos utilizando la información proporcionada en el cuerpo de la solicitud y devuelve un token JWT.
- **Iniciar sesión de un usuario:**
  - Método: POST
  - URL: '/users/login'
  - Descripción: Inicia sesión de un usuario utilizando las credenciales proporcionadas en el cuerpo de la solicitud y devuelve un token JWT.
- **Actualizar la información de un usuario:**
  - Método: PATCH
  - URL: '/users/{id}'
  - Descripción: Actualiza la información de un usuario específico identificado por el parámetro {id}, utilizando la nueva información proporcionada en el cuerpo de la solicitud.
- **Eliminar un usuario:**
  - Método: DELETE
  - URL: '/users/{id}'
  - Descripción: Elimina de la base de datos el usuario identificado por el parámetro {id}.

7.5.2 Factory

Para las operaciones CRUD se ha desarrollado un controlador siguiendo el patrón factory. Esta factory sigue un enfoque modular y reutilizable al encapsular la lógica de las operaciones CRUD en funciones independientes.

La factory está compuesta por varias funciones exportadas, cada una de las cuales representa una operación CRUD específica. Estas funciones se utilizan como controladores en diferentes rutas de la aplicación.

La función "createOne" se encarga de crear un nuevo documento en la base de datos utilizando el modelo correspondiente. Recibe los datos del cuerpo de la solicitud y los guarda

en la base de datos. A continuación, devuelve una respuesta que incluye el documento creado.

La función "getOne" se utiliza para obtener un documento específico de la base de datos utilizando el modelo y el identificador proporcionados en los parámetros de la solicitud. Si se especifica una opción de población, se realiza una operación de "populate" en el documento antes de devolverlo como parte de la respuesta.

La función "getAll" se encarga de recuperar todos los documentos de la base de datos utilizando el modelo correspondiente. Puede aceptar una consulta personalizada para aplicar filtros adicionales. Utiliza la clase "APIFeatures" para filtrar, ordenar, limitar campos o paginar los resultados. La respuesta devuelta incluye la lista de documentos y metadatos adicionales.

La función "updateOne" se utiliza para actualizar un documento existente en la base de datos utilizando el modelo y el identificador proporcionados. Los nuevos datos se obtienen del cuerpo de la solicitud y se actualiza el documento correspondiente en la base de datos. La respuesta devuelta incluye el documento actualizado.

Por último, la función "deleteOne" se encarga de eliminar un documento de la base de datos utilizando el modelo y el identificador proporcionados. Utiliza el método "findByIdAndDelete" para eliminar físicamente el documento de la base de datos. La respuesta devuelta indica el éxito de la operación.

En los diferentes controladores, se importan y utilizan las funciones de la factory para manejar las diferentes operaciones CRUD en el modelo correspondiente. Esto permite una organización modular y reutilizable del código.

7.5.3 Filtros

Se ha implementado la clase "APIFeatures", que desempeña un papel fundamental en el Back End al proporcionar una capa adicional de funcionalidad para manipular y gestionar consultas a la base de datos. Esta clase resulta especialmente beneficiosa en escenarios que involucran grandes volúmenes de datos y requieren la realización de operaciones avanzadas como filtrado, ordenamiento, selección de campos o paginación.

Una de las principales ventajas de utilizar "APIFeatures" radica en su capacidad para simplificar la implementación de consultas complejas en el Back End. En lugar de tener que

escribir código adicional para gestionar cada operación individualmente, los métodos ofrecidos por "APIFeatures" permiten encadenar estas operaciones de manera fluida y legible. Esto conduce a la generación de un código más limpio, fácil de mantener y menos propenso a errores.

El primer método relevante de "APIFeatures" es "filter()", el cual se encarga de filtrar los resultados obtenidos a través de la consulta. Este método es capaz de realizar tanto filtrados básicos como avanzados. El filtrado básico consiste en excluir ciertos campos de la cadena de consulta para evitar que sean considerados en la consulta a la base de datos. Por otro lado, el filtrado avanzado permite la utilización de operadores especiales como "gte" (mayor o igual que), "gt" (mayor que), "lte" (menor o igual que) y "lt" (menor que) para realizar comparaciones más precisas dentro de la consulta.

Otro método de relevancia es "sort()", el cual se utiliza para ordenar los resultados obtenidos de la consulta. Este método puede recibir parámetros de ordenamiento a través de la cadena de consulta y aplicarlos a la consulta original. Además, es posible especificar un idioma para establecer las reglas de ordenamiento regionales, lo cual resulta útil en casos donde se trabaja con datos multilingües.

"limitFields()" es otro método significativo de "APIFeatures", el cual permite limitar los campos que se devuelven como resultado de la consulta. Esta funcionalidad resulta útil cuando se desea reducir el tamaño de la respuesta y evitar la transferencia de datos innecesarios. En caso de recibir una lista de campos a través de la cadena de consulta, este método seleccionará únicamente esos campos para ser devueltos en el resultado.

La paginación es una técnica comúnmente utilizada en el manejo de grandes volúmenes de datos, y "APIFeatures" proporciona el método "paginate()" para llevar a cabo esta tarea. Mediante este método, es posible especificar el número de página y la cantidad de elementos por página en la cadena de consulta. "APIFeatures" realiza automáticamente el cálculo de la cantidad de elementos que deben omitirse y limita la consulta para devolver solamente los elementos correspondientes a la página actual.

Finalmente, la clase "APIFeatures" también ofrece el método "getTotalItems()", el cual resulta útil para obtener el número total de documentos que coinciden con la consulta. Esta funcionalidad es particularmente relevante cuando se aplica la paginación, ya que permite mostrar información precisa sobre la cantidad total de elementos disponibles.

7.5.4 Gestión de licitaciones con el mismo expediente

Al iniciar el desarrollo de los rastreadores para recopilar las licitaciones, se optó por utilizar el expediente como campo para distinguir y diferenciar cada una de ellas. Sin embargo, durante el desarrollo del proyecto, se detectó un escenario inesperado en el que dos licitaciones completamente diferentes compartían el mismo número de expediente. Esta situación generaba errores y problemas en el proceso, ya que la lógica implementada para crear y almacenar las licitaciones en la base de datos dependía de la verificación del expediente existente antes de realizar cualquier operación.

El inconveniente radica en que, al llegar una nueva licitación y comprobarse si ya existe otra con el mismo número de expediente en la base de datos, si se encuentra una coincidencia, la nueva licitación sobrescribe la existente con la información actualizada. Esto genera una pérdida de licitaciones, ya que, si dos licitaciones diferentes comparten el mismo expediente, la segunda sobrescribe la primera y se pierden los datos de la licitación original.

Para solucionar este problema, se ha decidido implementar la librería Fuse.js en los métodos encargados de crear las licitaciones en la base de datos. Fuse.js es una librería que permite realizar búsquedas y comparaciones de texto de forma flexible y eficiente. En este caso, se utiliza para comparar el contenido del objeto del contrato cuando se detecta un expediente duplicado.

Al recibir una nueva licitación con un expediente que ya existe en la base de datos, se utiliza Fuse.js para comparar el objeto del contrato de ambas licitaciones. La biblioteca devuelve un valor que representa la similitud entre las dos cadenas de texto. En el contexto de este proyecto, se establece un umbral de similitud de 0.55. Si la puntuación devuelta por Fuse.js es superior a este umbral, se considera que las dos licitaciones son la misma y no se realiza ninguna acción adicional, simplemente se actualiza la licitación ya existente con la nueva información.

Por otro lado, si la puntuación obtenida está por debajo del umbral establecido, se interpreta que las licitaciones son diferentes y se procede a crear una nueva entrada en la base de datos en lugar de sobrescribir la licitación existente. De esta manera, se garantiza que todas las licitaciones, aunque compartan el mismo número de expediente, se registran de manera individual y se conservan todos los datos relevantes.

7.5.5 Seguridad y autentificación

Para garantizar la seguridad de la aplicación y de los datos almacenados en la base de datos, se han implementado diversos métodos que prohíben el acceso a los diferentes endpoints y páginas de la aplicación a personas no autorizadas.

Uno de los enfoques utilizados para salvaguardar la integridad de los datos en la base de datos es mediante la inclusión de una clave en la cabecera de las peticiones que generan documentos de licitaciones. Esta clave se compara con una variable de entorno almacenada en un archivo de configuración. Si las claves coinciden, la petición se considera válida y se ejecuta, permitiendo así la creación de los documentos correspondientes. En caso de que las claves no coincidan, el servidor devuelve un mensaje de error con un código de estado 401, indicando que el acceso no está autorizado.

Además de garantizar la integridad de los datos, es necesario restringir el acceso a ciertas páginas y endpoints solo a usuarios con roles específicos. Para lograr esto, se utiliza el sistema de autenticación JWT (JSON Web Tokens). JWT es un estándar abierto basado en JSON que permite la transferencia segura de información entre dos partes mediante un token compacto y autónomo.

Cuando un usuario inicia sesión o se registra en la aplicación y se autentica correctamente, se genera un token JWT único. Este token contiene información relevante para verificar la identidad y los privilegios del usuario, en este caso su id, rol y nombre. Además, el token está firmado digitalmente utilizando un algoritmo de cifrado, lo que garantiza su integridad y seguridad.

El token JWT se compone de tres partes: el encabezado, la carga útil y la firma. El encabezado especifica el tipo de token y el algoritmo de cifrado utilizado. La carga útil contiene la información adicional mencionada anteriormente, que permite validar la identidad y los roles del usuario. La firma garantiza la integridad del token y asegura que no haya sido manipulado durante la transferencia.

Cuando un usuario intenta acceder a una página o endpoint protegido, debe incluir el token JWT en la cabecera de la petición HTTP. El servidor, a su vez, verifica la validez del token utilizando la clave secreta almacenada en el servidor. Si el token es válido y no ha expirado, el servidor autoriza el acceso y permite al usuario realizar la operación solicitada. En caso

de que el token no sea válido o haya expirado, se devuelve un código de estado 401, indicando que el usuario no está autorizado para acceder a la página o endpoint solicitado.

El uso de JWT brinda una capa adicional de seguridad al sistema, ya que elimina la necesidad de almacenar y mantener sesiones en el servidor. Al contener información codificada en el propio token, se reduce la necesidad de realizar consultas adicionales a la base de datos para validar la identidad y los permisos del usuario en cada solicitud. Esto contribuye a mejorar el rendimiento y la eficiencia del sistema en general.

Además de esto, para garantizar la seguridad de la información de los usuarios, en el modelo se ha creado un middleware que, al modificar la contraseña, ya sea al crear el usuario o más adelante, esta se encripta a través de bcrypt. De esta manera, evitamos tener guardada la contraseña en base de datos. También se ha configurado para que al hacer llamadas que devuelven información de usuarios, la contraseña no se incluya dentro de la información devuelta.

7.6 Front End

El Front End es la parte de la aplicación que se encarga de la interfaz gráfica de usuario, permitiendo al usuario interactuar con la aplicación.

En el desarrollo de este proyecto, se ha utilizado Vue.js para crear las diversas vistas, como se especifica en los requisitos tecnológicos del proyecto. Sin embargo, debido a la finalización de la colaboración con la empresa colaboradora, las vistas creadas han sufrido modificaciones con respecto a los objetivos iniciales.

Dado que no se cuenta con acceso a la aplicación original y se ha trasladado el desarrollo a un entorno local, se ha considerado importante añadir las siguientes vistas:

- Página de inicio de sesión.
- Página de registro.
- Página de error.

En esta sección, se presentan las distintas vistas creadas con el objetivo de cumplir con los objetivos establecidos en el proyecto, así como los distintos mecanismos y componentes añadidos para su correcto funcionamiento.

7.6.1 Vistas

Página de licitaciones recientes

Esta página es la página principal de la aplicación, incluye una lista con las últimas veinte licitaciones recogidas por los rastreadores con su información más importante: expediente, valor del contrato, el tipo de contrato, la organización contratante y el objeto del contrato. Se presenta cada elemento en la lista de la siguiente manera:

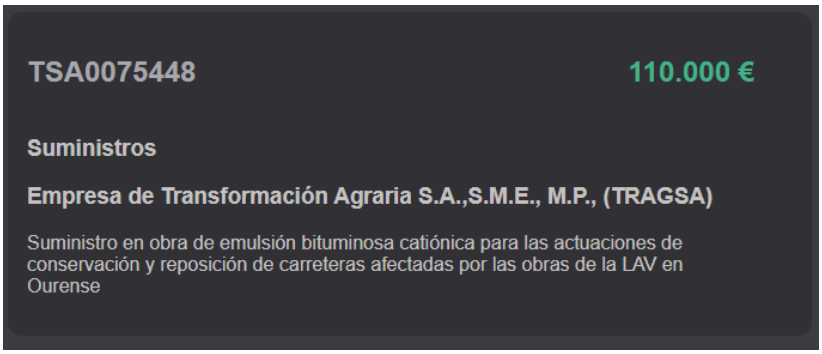

Figura 7.13 Visualización de una licitación en la página de licitaciones recientes

Para lograr esto, cuando la página entra dentro del estado “mounted”, uno de los estados dentro del ciclo de vida de una página de Vue.js, se realiza una petición al Back End para pedir dicha lista, que se obtiene a través de la combinación del endpoint que devuelve todas las licitaciones con el uso de filtros, por lo que la llamada que realiza la página es la siguiente: “http://127.0.0.1:3000/v1/tenders?limit=20&page=1”. Con esta llamada, sólo se reciben las últimas 20 licitaciones del total guardadas en base de datos.

Una vez obtenida esta lista de licitaciones del Back End, con las funcionalidades de Vue.js se crea un elemento de lista dentro de una lista desordenada por cada licitación obtenida en la llamada con la información proporcionada.

Con todo esto, se consigue mostrar por pantalla la lista con las últimas veinte licitaciones:

Figura 7.14 Página de licitaciones recientes

No hay restricción de ningún tipo para acceder a esta página, por lo que todos los usuarios de la aplicación pueden acceder.

Página de administrador/super-administrador:

Está es la página para administradores y super-administradores. Originalmente en los objetivos del proyecto se planteó realizar una página separada para cada rol, pero aprovechando las funcionalidades de Vue.js se ha decidido hacerlo en una sola página. En caso de que el usuario sea administrador, se esconden aquellos datos que sólo deban ser accesibles para los super-administradores. La información mostrada en la página es la siguiente:

- Sección de super-administrador
    - Últimos rastreadores ejecutados
    - Errores recientes en la plataforma
- Licitaciones
    - Licitaciones totales
    - Licitaciones recogidas en el último día
    - Licitaciones del BOE
    - Licitaciones de Contratación del Estado
    - Licitaciones de Contratos Menores
    - Licitaciones del DRE
    - Licitaciones del TED
- Organizaciones
    - Organizaciones registradas
    - Organizaciones contratantes
    - Organizaciones adjudicatarias
- Usuarios
    - Usuarios registrados
    - Usuarios registrados en el último día

Esta es la vista para un usuario super-administrador, en caso de ser administrador la primera sección no aparecería:

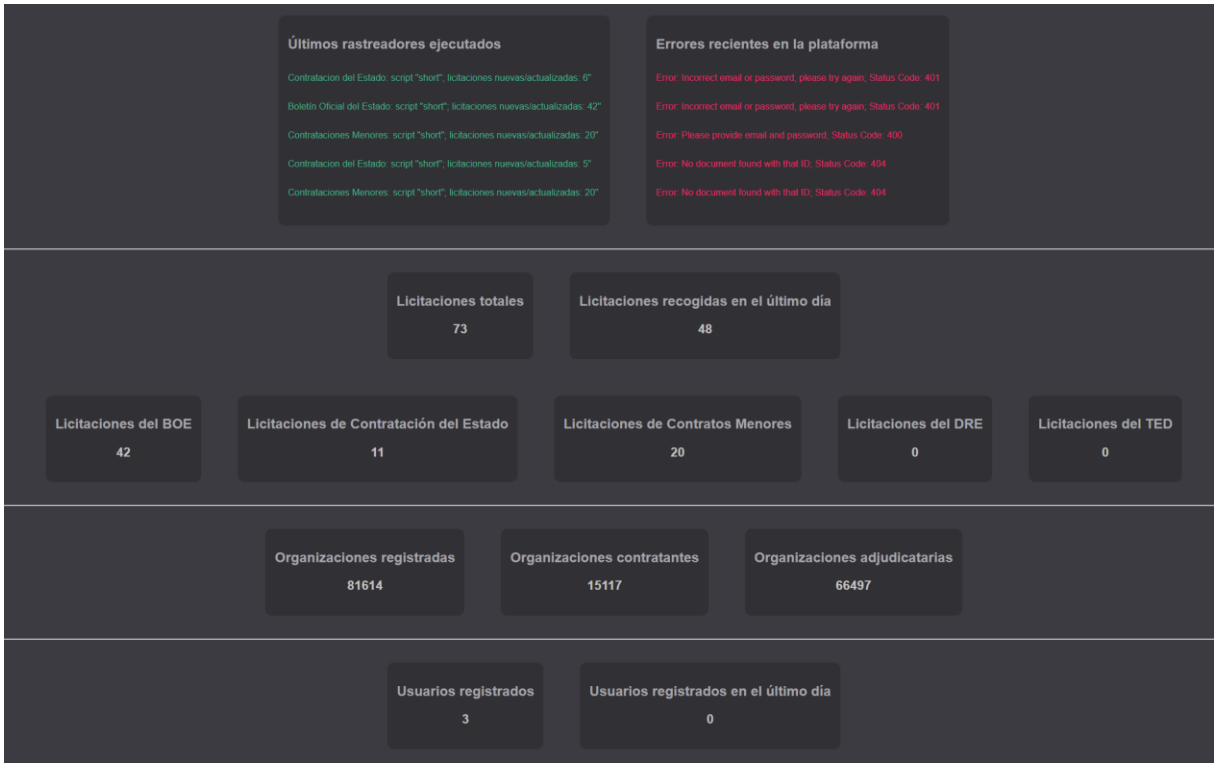

Figura 7.15 Página de super-administrador

Además, aprovechando la reactividad que ofrece Vue.js, se logra la actualización en tiempo real de toda la información. Esto implica que si hay cambios en alguno de los valores mientras el usuario está en la página, dichos valores se actualizan automáticamente sin necesidad de recargar la página.

Para lograr esto, la página carga inicialmente toda la información mediante las llamadas correspondientes a la API al llegar al estado "mounted", y luego realiza nuevas llamadas a la API cada cinco segundos. Si hay cambios en algún valor con respecto a su estado anterior, se realiza la actualización correspondiente.

Página de inicio de sesión:

Esta vista es la vista predeterminada de la aplicación y requiere autenticación para acceder. Presenta un formulario sencillo en el cual el usuario debe ingresar su correo electrónico y contraseña.

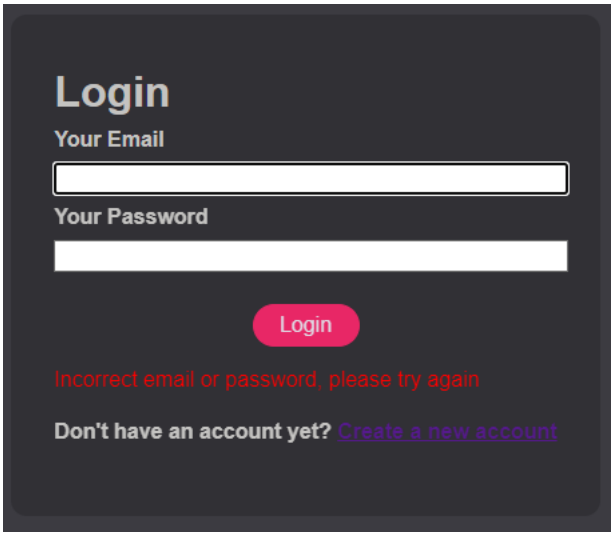

Figura 7.16 Formulario de inicio de sesión

En caso de cometer algún error, se muestra el mensaje de error devuelto por la API. Si las credenciales son correctas, el usuario es redirigido a la pantalla de licitaciones recientes. Además, se incluye un enlace para aquellos usuarios que no tengan una cuenta y deseen crear una.

Pantalla de registro

La pantalla de registro, similar a la vista anteriormente descrita, muestra un formulario donde el usuario debe proporcionar su nombre, correo electrónico y contraseña. Una vez enviado el formulario, los datos se envían al Back End para su procesamiento. Si todo es correcto, se crea un nuevo usuario y se redirige a la pantalla de licitaciones recientes.

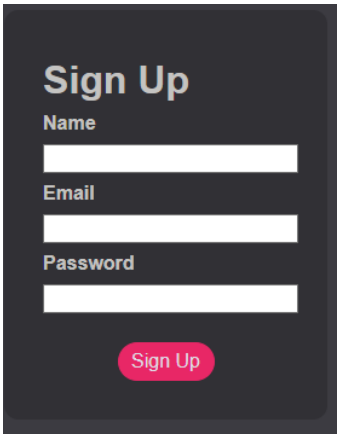

Figura 7.17 Formulario de inscripción

Pantalla de error:

La pantalla de error se muestra cuando el usuario intenta acceder a la pantalla de administrador sin tener los permisos adecuados o cuando ocurre algún error. Esta vista muestra un mensaje de error de forma simple y clara.

Figura 7.18 Página de error

7.6.2 Componentes

Aprovechando las funcionalidades proporcionadas por Vue.js, se ha desarrollado un componente llamado "Header". Este componente se utiliza en varias vistas de la aplicación y se encuentra ubicado en la parte superior de la pantalla. La siguiente imagen muestra cómo se visualiza este componente:

Figura 7.19 Componente "Header"

En la imagen se puede observar que el componente presenta un mensaje de bienvenida y un botón para cerrar sesión en el lado derecho. En el lado opuesto, se encuentra un menú que permite la navegación entre la página de administradores y la lista de las últimas licitaciones.

Gracias a las características de Vue.js, si el usuario que ha iniciado sesión no tiene el rol de administrador o super-administrador, el componente no muestra la opción de navegar hacia la página destinada a estos usuarios.

7.6.3 Enrutamiento

Para habilitar la navegación entre páginas en la aplicación, se ha utilizado el enrutador de Vue.js. Esto permite definir las rutas dentro de la aplicación y especificar qué página se debe mostrar en cada una. A continuación, se presentan las rutas implementadas:

- **"localhost:3000/":** Esta ruta redirige automáticamente a "localhost:3000/login", es decir, lleva a la página de inicio de sesión.
- **"localhost:3000/licitaciones":** Esta ruta carga la página de licitaciones recientes.
- **"localhost:3000/login":** Esta ruta renderiza la página de inicio de sesión.
- **"localhost:3000/signup":** Esta ruta lleva a la pantalla de registro.
- **"localhost:3000/admin":** Esta ruta carga la pantalla de administradores.
- **"localhost:3000/notFound":** Esta ruta carga la pantalla de error cuando una ruta no válida es accedida.
- **"localhost:3000/:notFound(.*)":** Esta ruta se carga cuando se introduce una ruta que no se encuentra entre las especificadas. Por defecto, redirige a la página de error.

Gracias al enrutador de Vue.js, se logra una navegación fluida y controlada entre las diferentes páginas de la aplicación, mejorando la experiencia del usuario.

7.6.4 Seguridad

Con el objetivo de asegurar que los usuarios no administradores no tengan acceso a la página de administradores, se ha implementado un sistema de seguridad que combina el estándar JWT con el enrutador de Vue.js.

Una vez que el Front End ha almacenado el token, se utiliza el método "beforeEach()" proporcionado por el enrutador de Vue.js. Este método se ejecuta antes de acceder a cualquiera de las rutas especificadas. Dentro de este método se consideran dos casos:

1. La ruta a la que se intenta acceder contiene información meta que indica que no se requiere autorización. En este caso, se permite el acceso a la ruta.
2. La ruta a la que se intenta acceder contiene información meta que indica que se requiere autorización. En este caso, el token se envía al Back End, el cual verifica su validez y devuelve el rol del usuario asociado al token. Si el rol es de usuario, se redirige automáticamente a la página de error. Si el rol es de administrador o super-administrador, se permite el acceso a la página de administradores.

Este sistema de seguridad basado en JWT y el enrutador de Vue.js garantiza que solamente los usuarios con los roles adecuados puedan acceder a la página de administradores, protegiendo así la información y los privilegios de la aplicación.

8. Análisis de resultados

En este proyecto, se ha desarrollado una aplicación que cumple con varios objetivos originales. Se han creado rastreadores de licitaciones utilizando Python para diversas plataformas, como el Boletín Oficial del Estado (BOE), el Diário da República (DRE), el Diario Oficial de la Unión Europea (DOUE) y la Sección de Contratos Menores de la Plataforma de Contratación del Sector Público.

Además, se ha implementado una sección de Superadministrador utilizando JavaScript y Vue.js, la cual permite mostrar en tiempo real los errores que ocurran en la plataforma, así como el estado de los rastreadores y posibles errores asociados a ellos.

También se ha creado una sección de Administración utilizando JavaScript y Vue.js, que proporciona información sobre las cuentas creadas, los usuarios registrados, las suscripciones activas y otra información relevante para el usuario.

En cuanto a la integración de herramientas de terceros, se tenía previsto utilizar las API de Twitter, Mixpanel, Google y LinkedIn, pero desafortunadamente esta integración no se ha llevado a cabo en el desarrollo de la aplicación.

Además de los objetivos originales, se han añadido otros objetivos adicionales, como la creación de una página de licitaciones recientes, una página de inicio de sesión y una página de registro. Estos objetivos adicionales se han logrado satisfactoriamente.

Durante el proceso de desarrollo, se han encontrado algunos desafíos. En particular, los rastreadores del Diário da República se han vuelto inutilizables debido a una modificación en la página web que dificulta su adaptación al nuevo formato. A pesar de este obstáculo, se han cumplido exitosamente la mayoría de los objetivos iniciales.

9. Conclusiones

Después de completar este proyecto, se pueden obtener las siguientes conclusiones:

El desarrollo de los rastreadores de licitaciones ha sido una tarea desafiante que ha presentado obstáculos inesperados. Tanto la falta de experiencia en técnicas de rastreo como las constantes modificaciones en las diversas páginas web de las plataformas seleccionadas han influido en el tiempo necesario para completar esta parte del proyecto. Los errores y cambios en la estructura de las páginas web han dificultado la adaptación y el correcto funcionamiento de los rastreadores, lo que ha requerido más tiempo de desarrollo en esta sección.

Por otro lado, el desarrollo del Front End de la aplicación ha seguido una dinámica diferente. La integración de las pantallas de administrador y superadministrador ha permitido ahorrar tiempo significativamente en el proceso de desarrollo. Al unificar estas dos áreas en una sola interfaz, se ha logrado simplificar la estructura y las funcionalidades, lo que ha llevado a una finalización más rápida de esta parte del proyecto. Este ahorro de tiempo ha permitido agregar las pantallas de licitaciones recientes, acceso y registro sin afectar el tiempo dedicado al desarrollo de esta sección.

Estas conclusiones destacan la importancia de contar con una planificación adecuada y una evaluación realista de los obstáculos que pueden surgir durante el desarrollo de un proyecto. Además, subrayan la necesidad de adquirir experiencia y conocimientos específicos en áreas técnicas para enfrentar desafíos complejos. A pesar de las dificultades encontradas, el proyecto en su conjunto ha logrado alcanzar sus objetivos y ha proporcionado una base sólida para investigaciones futuras y mejoras en el campo del seguimiento de licitaciones.

10. Posibles ampliaciones

Dentro del ámbito de posibles ampliaciones para el proyecto, se pueden considerar varias opciones:

Una de las ampliaciones que se podrían implementar es la adición de nuevas fuentes de licitaciones. Aunque el proyecto ya incluye las fuentes principales, existe la posibilidad de incorporar más fuentes en el futuro. El backend ha sido diseñado para ser adaptable y permitir la integración de nuevas fuentes sin dificultades.

Otra mejora que se puede considerar es la unificación de los controladores para la creación de licitaciones. En la implementación actual, existe un controlador individual por cada fuente de licitaciones debido a las pequeñas variaciones en la información presentada por cada una. Sin embargo, con más tiempo y recursos, se podría trabajar en un controlador único que pueda manejar estas diferencias y garantizar una mayor eficiencia en el código.

En cuanto al Front End, se pueden añadir más páginas y funcionalidades para mejorar la experiencia del usuario. Por ejemplo, se podría desarrollar una pantalla específica que muestre toda la información detallada de una licitación concreta. Esto permitiría a los usuarios acceder a los detalles relevantes de manera más rápida y sencilla.

Otra ampliación interesante sería la inclusión de un buscador dentro del Front End. Con esta funcionalidad, los usuarios podrían realizar búsquedas específicas de licitaciones utilizando criterios como palabras clave, fechas o categorías. Esto facilitaría la navegación y el acceso a la información relevante para cada usuario.

Por último, se podría mejorar la pantalla de licitaciones recientes haciendo que al pulsar sobre un elemento de la lista, la página redirija a la pantalla de detalle de una única licitación mencionada anteriormente. Esta mejora proporcionaría una experiencia de usuario más completa y permitiría un acceso rápido a los detalles necesarios.

11. Bibliografía

[1] Arenas Tomás, J. A. (2015). *Shipeer: Desarrollo de un servicio de paquetería colaborativo mediante Node. js* (Doctoral dissertation, Universitat Politècnica de València).

[2] Santos Sánchez, J. (2022). Web Scraping y microservicios.

[3] *Qué es MongoDB y características*. (2022, 3 noviembre). OpenWebinars.net. https://openwebinars.net/blog/que-es-mongodb/

[4] Node.js Event Loop. (2016). Dev.to. https://thepracticaldev.s3.amazonaws.com/i/wrtzmt2ty03ksew7ehvx.jpg

[5] GeeksforGeeks. (2021, 11 octubre). *Node.js Event Loop*. https://www.geeksforgeeks.org/node-js-event-loop/

[6] Rakowski, F. (2020, 8 octubre). *Introduction to Vue - key concepts and why it's so awesome*. Buddy: The DevOps Automation Platform. https://buddy.works/tutorials/introduction-to-vue-key-concepts

[7] Atlassian. (s. f.). *Scrum: qué es, cómo funciona y por qué es excelente*. https://www.atlassian.com/es/agile/scrum

[8] *Home | Scrum Guides*. (s. f.). https://scrumguides.org/index.html

[9] Lara, W. (2015, 31 julio). *¿Cómo funciona la metodología Scrum? Qué es y sus 5 fases*. Platzi. https://platzi.com/blog/metodologia-scrum-fases/

[10] *Aviso Legal*. (s.f.). Contrataciondelestado.es. Recuperado 9 de febrero de 2023, de https://contrataciondelestado.es/wps/portal/!ut/p/b1/jc7LCsIwFATQL5Lc3CYhWaavJFIfEFptNtKFSKWPjfj9xuLW6uwGzsCQQNqNRMkVMkXJmYSpe_a37tHPUze8exCX1EupU6rB-B0FLYpEcARqPEbQRsCTjDXb5ii8MwDOlnlVUw4GxX97-BINv_YnEhbCikOWlRZB-iQHrPK6FjZWgx-wdnEBKx_2dh6vZAxDqdydvQBJOB-K/dl4/d5/L2dJQSEvUUt3QS80SmtFL1o2X0FWRVFBSTkzME8xS0MwMkJOMFNHTUgzMFEz/

[11] *Información*. (s.f.). Boe.es. Recuperado 9 de febrero de 2023, de https://www.boe.es/informacion/aviso_legal/index.php

[12] *Avisos Legais*. (s. f.). Dre.pt. Recuperado 9 de febrero de 2023, de https://dre.pt/dre/geral/avisos-legais

[13] *Legal notice*. (s. f.). TED Tenders Electronic Daily. Recuperado 9 de febrero de 2023, de https://ted.europa.eu/TED/misc/legalNotice.do