

1. Memoria

El 23 de octubre de 2023, el Centro ITICBCN, una institución educativa especializada en Informática y Comunicaciones, nos contactó para encargarnos del desarrollo de una aplicación integral. Esta aplicación tiene como objetivo mejorar y agilizar la gestión de su bolsa de trabajo para estudiantes. El Centro ITICBCN ha expresado la necesidad de optimizar sus procesos actuales, que en su mayoría se realizan manualmente, con el fin de brindar a sus estudiantes un acceso más eficiente a las oportunidades de empleo proporcionadas por las empresas colaboradoras.

2. Objeto del proyecto

2.1 Usuarios y contexto de uso

En la situación actual, la gestión de ofertas de trabajo y la búsqueda de oportunidades de prácticas laborales para los alumnos suelen ser tareas que requieren tiempo y recursos considerables, ya que son procesos que se realizan de forma manual. Desde la recopilación y distribución de información, la validación de los perfiles de los estudiantes, el envío de currículos a empresas, entre otras tareas administrativas. Estos procesos, además de ser laboriosos, a menudo pueden tener errores humanos.

Por dicho motivo, se realizará una aplicación que permita automatizar una serie de funcionalidades con el objetivo de mejorar la eficacia del trabajo a realizar.

Se realizará una aplicación para móviles y una página web en la que empresas puedan añadir ofertas de prácticas y los alumnos puedan postular en ellas. Se trabajarán dos perfiles: alumnos y administradores, el primero con el objetivo de que puedan observar las ofertas y escoger empresas con las que se sientan cómodos y más afines a su perfil y el segundo perfil con el objetivo de que administre el tema de ofertas de empresa.

Principalmente, tanto alumnos como administradores tienen que tener un login, una vez dentro de la aplicación, los alumnos tendrán un apartado para insertar sus datos

personales y su CV. Los alumnos, tendrán un buscador donde prácticas donde podrán apuntarse y se enviará un correo automáticamente a la empresa en cuestión, donde habrá la información del alumno y su CV.

Los administradores, gestionan tanto las ofertas como las empresas, pudiendo modificar, insertar, como eliminar. También tendrán que validar los CV de los alumnos y encargarse de que la información es correcta y podrán validarlos masivamente mediante un documento CSV.

2.2 Funcionalidad

La plataforma proporciona a los alumnos la capacidad de buscar empleo, cargar sus CV y postularse a ofertas una vez que sus CV sean validados. A los administradores se les da control sobre las ofertas de empleo, la gestión de usuarios y la validación de CV, entre otras funciones. Ambos tipos de usuarios deben autenticarse a través de un inicio de sesión para acceder a estas funcionalidades.

Funcionalidades para Alumnos:

- Registro: Los alumnos deberán registrarse en la plataforma introduciendo su nombre, apellidos, correo electrónico y una contraseña segura.
- Login: Tendrán que iniciar sesión en la plataforma con su correo electrónico y contraseña para acceder a la plataforma digital.
- Subir CV: Los estudiantes podrán cargar su currículum vitae en formato digital para su revisión y validación.
- Buscar Ofertas de Empleo: Podrán encontrar diferentes ofertas de empleo utilizando filtros como ubicación, industria, y tipo de empleo.

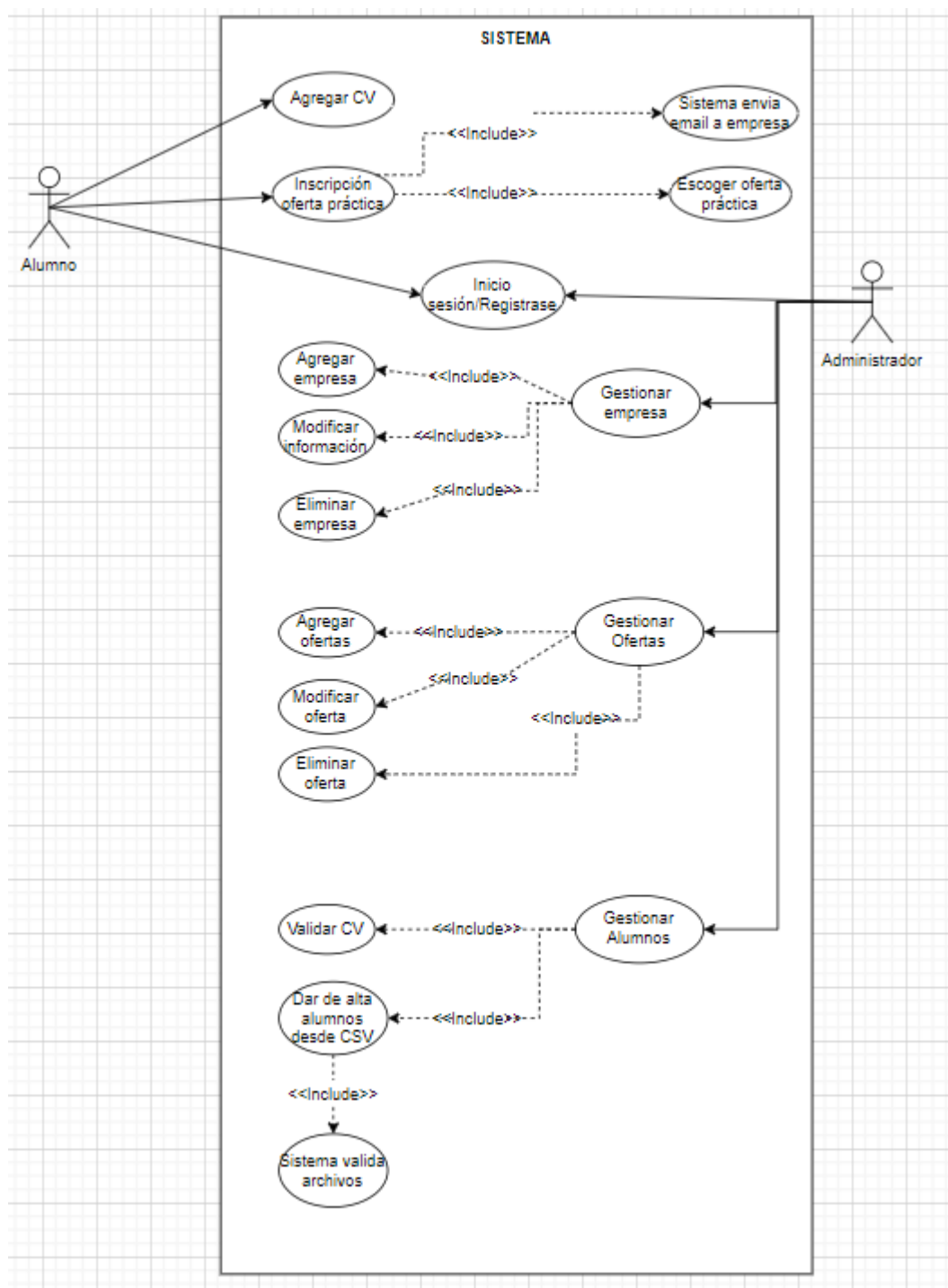
- Visualizar Detalles de Ofertas: Al encontrar una oferta que les interese podrán ver detalles de estas, que incluyen información sobre la empresa, requisitos, beneficios y cómo postular.
- Aplicar a Ofertas: Una vez que el CV del alumno sea validado, podrán postularse a las ofertas de empleo que les interesen.
- Notificaciones: Los alumnos recibirán notificaciones por correo electrónico, sobre el estado de sus aplicaciones.

Funcionalidades para Administradores:

- Inicio de Sesión: Los administradores también deberán hacer un login con credenciales específicas para acceder a las funcionalidades de administrador.
- Gestión de Ofertas de Empleo: Podrán agregar nuevas ofertas, incluyendo información detallada sobre la oferta, requisitos y fecha.
- Editar y Eliminar Ofertas: Tendrán la capacidad de editar o eliminar ofertas de empleo existentes en caso de cambios en los detalles de la oferta o cuando expiren.
- Validación de CV: Los administradores revisarán y validarán los curriculum de los alumnos para asegurarse de que cumplan con los requisitos antes de permitirles postularse a ofertas.
- Registro Masivo: Los administradores podrán realizar un registro masivo de alumnos en la plataforma a través de la carga de un archivo CSV que contenga información de los estudiantes.
- Gestión de Usuarios: Los administradores podrán crear y gestionar cuentas de usuario para alumnos, también tendrán habilitado restablecer contraseñas en caso de que la olviden.
- Historial de Actividades: Se mantendrá un registro de las actividades realizadas por los administradores para garantizar la transparencia y la seguridad de la plataforma.

3. Funcionamiento

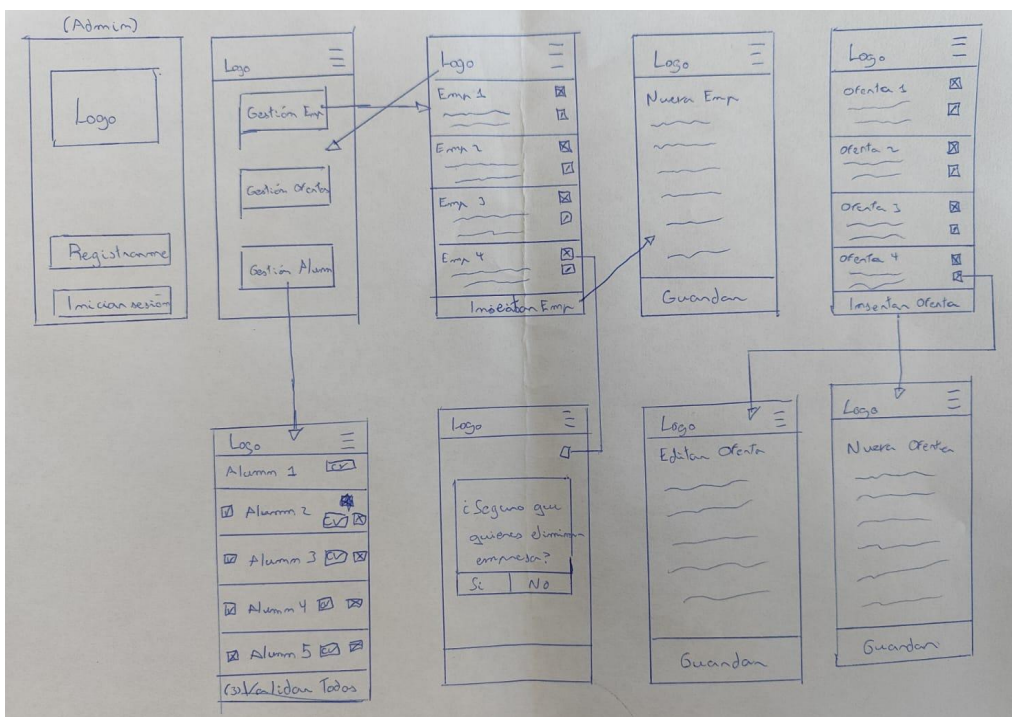
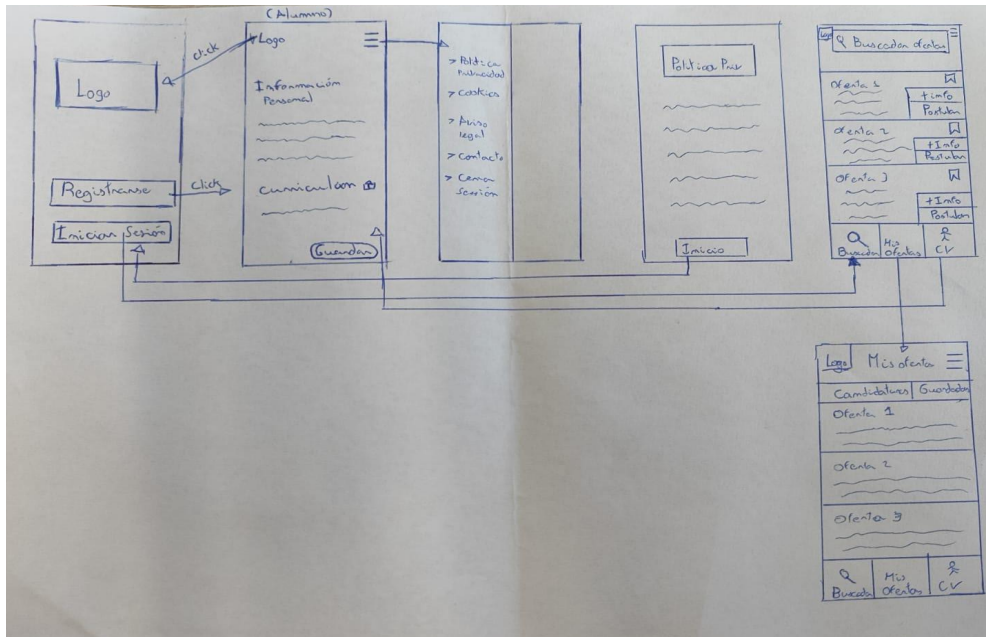
3.1 Diagrama de casos de uso



Enlace Diagrama:

https://drive.google.com/file/d/1Q3wmaqDjd9YFx1DBmwBL7I-vS2_AsUEm/view?usp=sharing

3.2 Wireframe de baja fidelidad



4. Antecedentes

Requisitos previos para el inicio del API REST:

- Ir a *Spring Initializr* y agregar las dependencias de: WEB, H2 y JPA.
- Descargar el archivo .ZIP y descomprimirlo.
- Abrir el proyecto con un IDE a elección, en nuestro caso, Eclipse.

Con esos pasos ya hechos, comenzamos con el desarrollo de las funcionalidades del API REST.

Comenzamos creando la clase objeto, en este caso es **Empresa**, ya que en ella recaen las funcionalidades que vamos a ejecutar. Funciona como una entidad específica, que contiene una serie de atributos y métodos.

Utilizaremos métodos JPA como `@Entity`, para marcar la clase Empresa como una entidad JPA y pueda ser mapeable a una tabla en una base de datos.

`@Table` para especificar el nombre de la tabla en la base de datos a la que se mapeara la entidad.

`@Id` para marcar un campo como clave primaria de la entidad.

`@GeneratedValue` que lo utilizaremos junto al método mencionado anteriormente para especificar cómo se generará el valor de la clave primaria.

A continuación, configuramos el archivo **application.properties** con:

- `spring.datasource.url=jdbc:h2:mem:testdb` para establecer la URL de la base de datos y utilizarla como prueba y desarrollo del proyecto, hay que tener en cuenta que cada vez que iniciamos la aplicación, la base de datos se elimina y se comienza de 0.
- `spring.datasource.driverClassName=org.h2.Driver` especificamos el nombre del controlador, en nuestro caso H2.
- `spring.datasource.username=` definimos el nombre de usuario para acceder a la base de datos.
- `spring.datasource.password=` establecer la contraseña de nuestro usuario de la base de datos.
- `spring.h2.console.enabled=true` Habilita la consola web de H2 para acceder y administrar la base de datos.
- `spring.jpa.show-sql=true` permite que Spring Boot imprima las sentencias SQL generada por JPA.
- `spring.jpa.hibernate.ddl-auto=update` para controlar la creación y actualización de la estructura de la base de datos.

El siguiente paso fue configurar una interfaz que extiende *JpaRepository* para que interactúe con la entidad **Empresa**. Se utilizaron dos parámetros: *Empresa* y *Long*.

Utilizamos la anotación *@Repository* para marcar la interfaz como un componente de Spring. Gracias a *JpaRepository* que proporciona una amplia variedad de métodos para realizar operaciones de acceso a datos en la entidad **Empresa** pudimos desarrollar las siguientes funcionalidades creando un controlador para manejar las solicitudes HTTP entrantes:

US: Como usuario puedo consultar una empresa en la API.

Creamos un controlador de Spring que se encargará de las solicitudes GET para poder consultar la información de la empresa que queramos. El controlador tiene la anotación *@RestController*, que responderá las solicitudes HTTP y devolverá los datos de la empresa en formato JSON.

Se puede consultar por ID.

US: Como usuario puedo agregar una empresa en la API.

Los usuarios pueden enviar una solicitud POST con los datos de la empresa en formato JSON y la empresa se guardará en la base de datos.

US: Como usuario, puedo modificar una empresa en la API.

Los usuarios deben proporcionar el ID de la empresa que desean modificar y enviar una solicitud PUT con los datos actualizados en formato JSON. El controlador llamará al servicio para actualizar la empresa correspondiente en la base de datos.

US: Como usuario, puedo eliminar una empresa de la API.

Los usuarios deben proporcionar el ID de la empresa que desean eliminar y enviar una solicitud DELETE. La empresa correspondiente se eliminará de la base de datos.

Para la siguiente parte del proyecto, nos informarnos sobre cómo funcionan las relaciones 1-N mediante anotaciones JPA

Inicialmente creamos la entidad **Oferta** para representarla en el sistema y asociar cada oferta creada a la empresa que nos interese. Esta entidad continúa el mismo proceso que la entidad **Empresa**, donde el esqueleto de la clase se encuentra formado por los atributos, un constructor y los getters y setters correspondientes.

@JoinColumn(name = "empresa_id"): Especifica el nombre de la columna en la tabla de ofertas que se utilizará como clave externa para la relación.

También, agregamos la anotación `@ManyToOne` para crear la relación con la entidad **Empresa**, lo que significa que varias ofertas pueden pertenecer a una empresa. En la clase **Empresa**, se utiliza la anotación `@OneToMany` para representar la relación con la clase **Oferta**. Esta anotación especifica que una empresa puede tener una colección de ofertas.

`@OneToMany(mappedBy = "empresa", cascade = CascadeType.ALL, orphanRemoval = true)`
`mappedBy = "empresa"`: Indica que la relación está mapeada por el atributo `empresa` en la clase **Oferta**.

`cascade = CascadeType.ALL`: Especifica que las operaciones de cascada, como eliminar, se aplicarán a las ofertas asociadas cuando se realice una operación en la empresa.

`orphanRemoval = true`: Indica que las ofertas se eliminarán si se eliminan de la colección en **Empresa**.

US. Como usuario, puedo agregar una oferta de una empresa a la API.

Los usuarios pueden agregar una oferta enviando una solicitud POST con los datos de la oferta en formato JSON. Las ofertas agregadas se almacenarán en la base de datos.

US. Como usuario, puedo consultar ofertas en la API.

Los usuarios pueden obtener una lista de todas las ofertas disponibles enviando una solicitud GET al endpoint que corresponda. Esto devolverá una lista de ofertas en formato JSON.

US. Como usuario, puedo consultar ofertas de una empresa específica en la API.

Los usuarios pueden consultar las ofertas de una empresa específica enviando una solicitud GET al endpoint que incluye el ID de la empresa. Esto devolverá una lista de ofertas asociadas a esa empresa en formato JSON.

US. Como usuario, puedo actualizar una oferta de una empresa dada en la API.

Los usuarios deben proporcionar el ID de la oferta que desean actualizar y enviar una solicitud PUT con los datos actualizados en formato JSON. El controlador llamará al servicio para actualizar la oferta correspondiente en la base de datos.

US. Como usuario, puedo eliminar una oferta de una empresa dada en la API.

Los usuarios deben proporcionar el ID de la oferta que desean eliminar y enviar una solicitud DELETE. La oferta correspondiente se eliminará de la base de datos.

US. Como usuario, si borro una empresa, automáticamente se eliminarán todas las ofertas asociadas a esa empresa.

Esta opción implica que hay una relación entre las empresas y sus ofertas, y cuando se elimina una empresa, las ofertas vinculadas también se eliminan para mantener la coherencia de los datos.

5. KANBAN

2.1 Primer Sprint

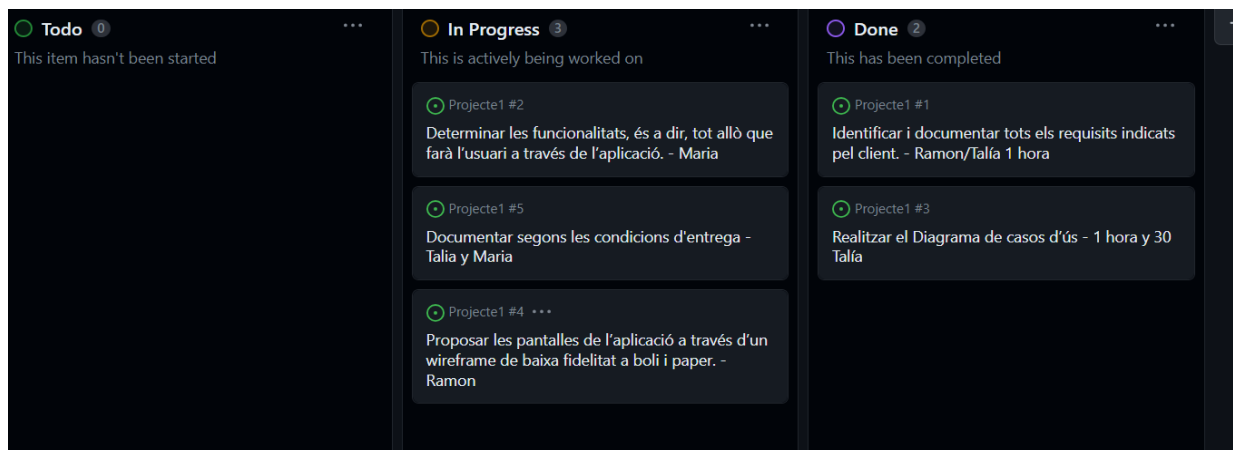
Primer día / Backlog organizativo:

- Comenzamos a definir las tareas que serán el esqueleto organizativo del proyecto.



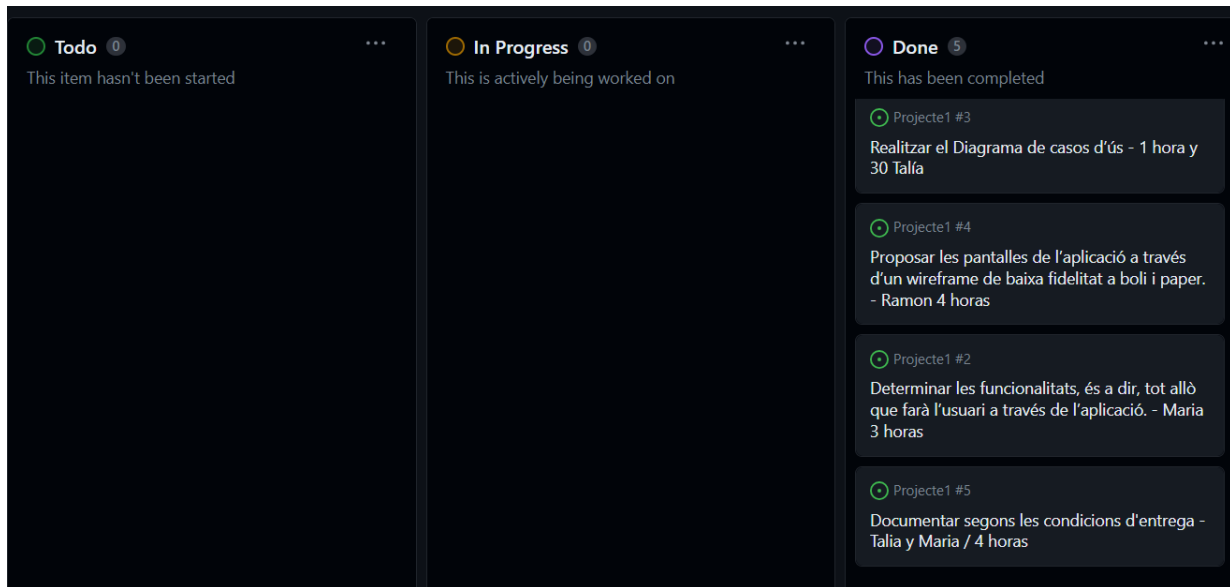
Segundo día:

- Se dan como hechas las dos primeras asignadas y se cronometra el tiempo. Documentar según las condiciones de entrega se queda en progreso porque es un trabajo que se va haciendo junto al desarrollo de la misma memoria y determinar las funcionalidades no se terminó porque se busco la forma de documentar y se buscaron ejemplos de cómo enfocar estas funcionalidades.



Tercer día:

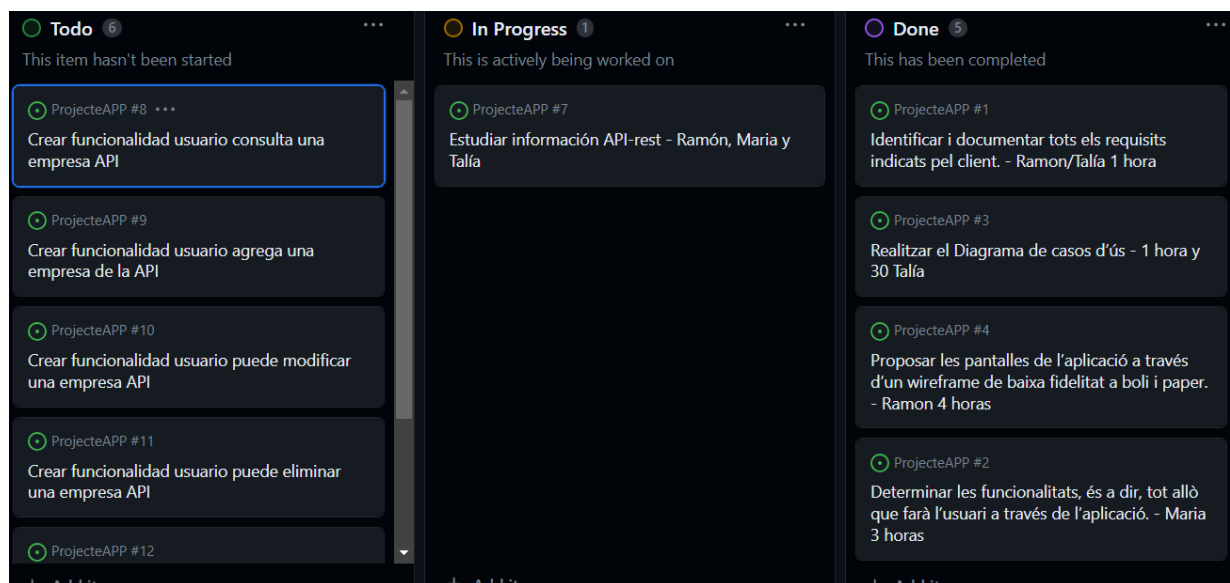
- Se finalizan las tareas y se revisa que todo esté bien documentado.



2.2 Segundo Sprint

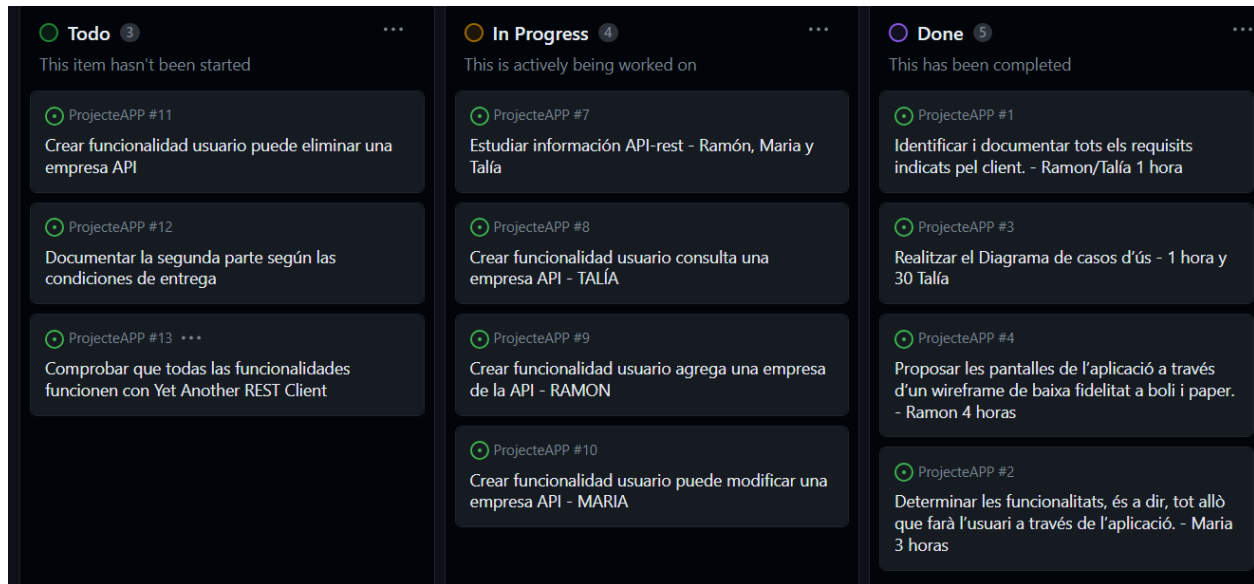
Primer día / backlog organizativo:

- Comenzamos a definir las nuevas tareas a realizar y asignamos las que se ejecutarán el día de hoy.



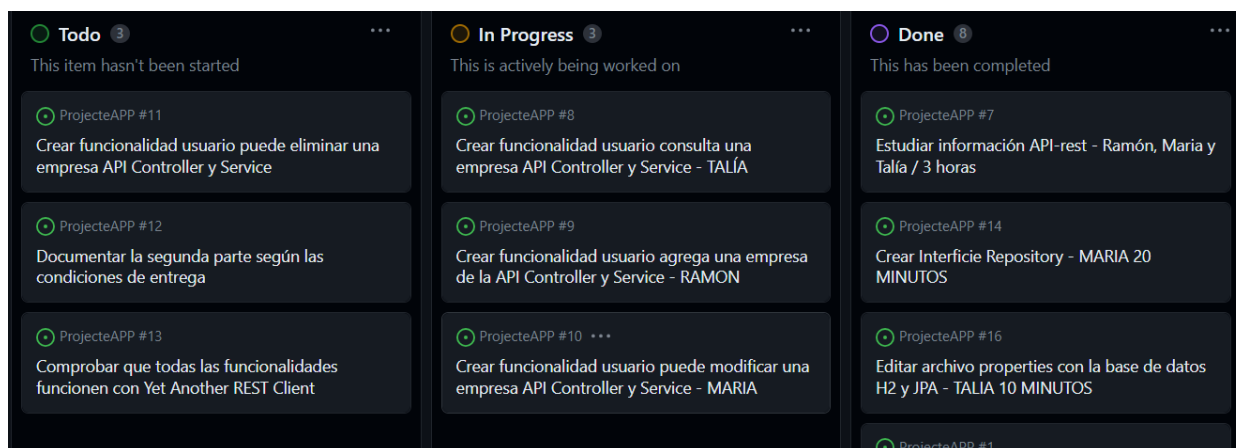
Segundo día:

- Seguimos recopilando información y comenzamos a programar las funcionalidades descritas.



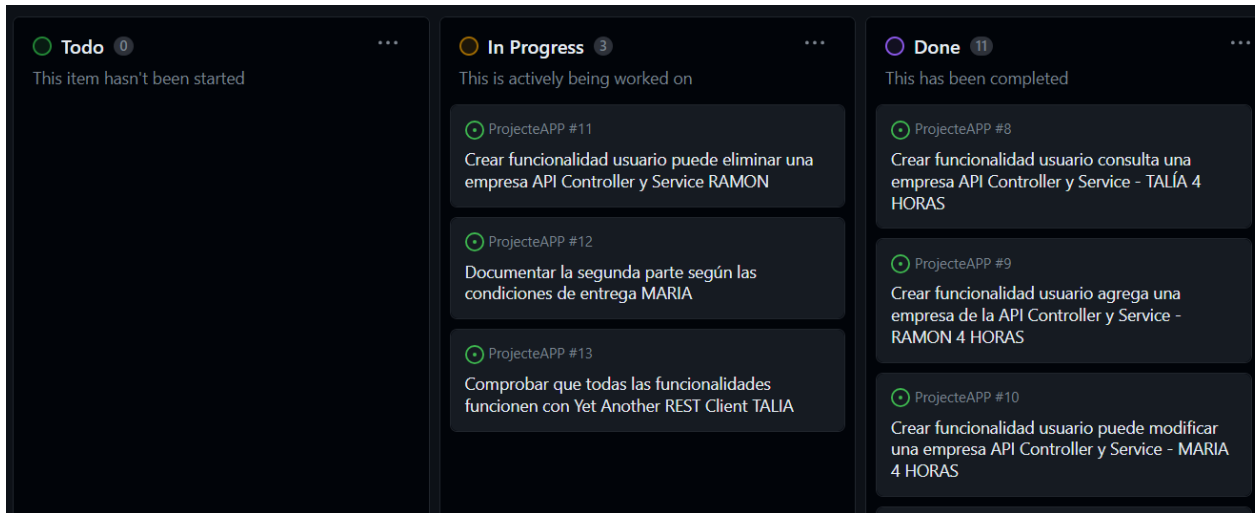
Tercer día:

- Continuamos con las antiguas funcionalidades y agregamos detalles en KANBAN que acabamos el mismo día como crear la clase REPOSITORY y editar el archivo Properties. Lo del día anterior no se acabó porque agregamos el SERVICE al proyecto, que era lo que nos faltaba.



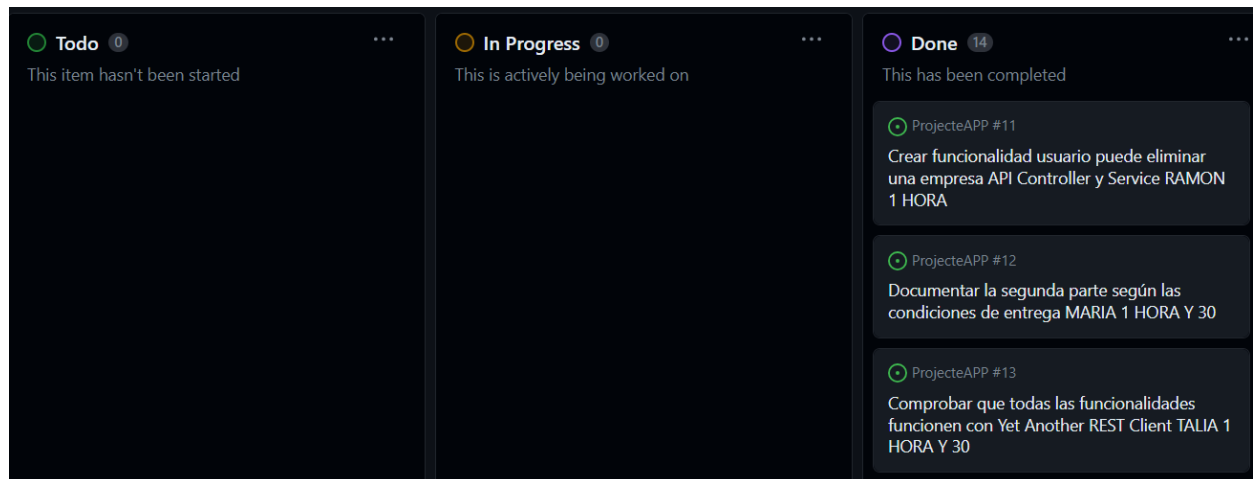
Cuarto día:

- Se finalizan las tareas asignadas y se asignan las últimas, teniendo en cuenta las pautas de documentación, además se comienzan a hacer las pruebas con Yet Another REST Client.



Quinto día:

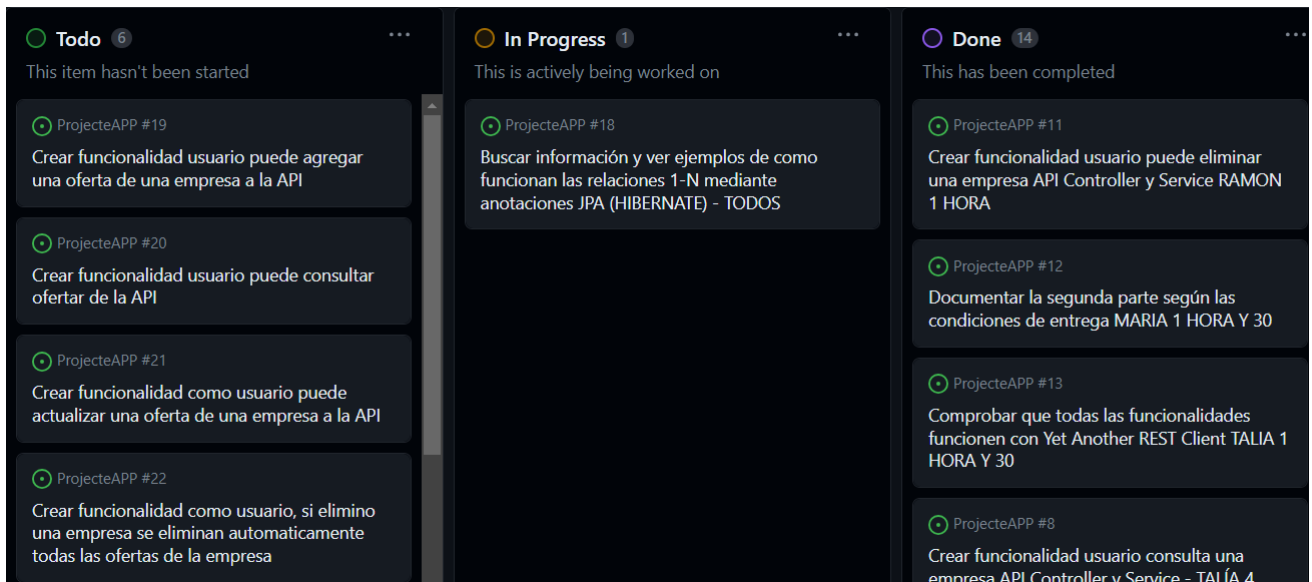
- Se termina lo pendiente y finalmente se realizaron las pruebas finales con POSTMAN, las cuales tuvieron una larga duración por un problema en la ejecución con la jerarquía de los packages del proyecto.



2.3 Tercer Sprint

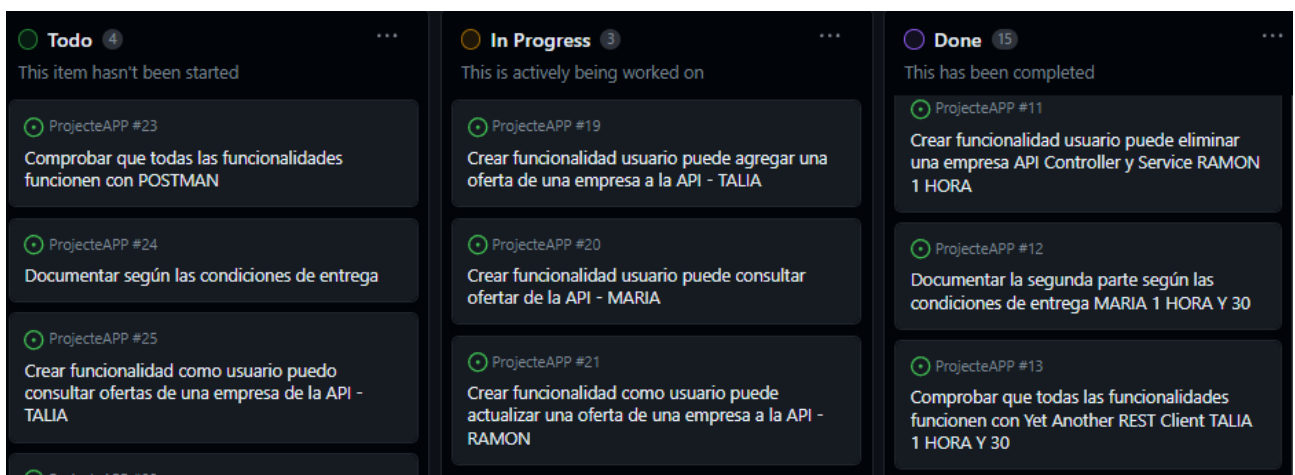
Primer y segundo día / Backlog organizativo:

- Comenzamos a asignar las tareas que vamos a desarrollar a lo largo de la semana y comenzamos con la principal para desarrollar el resto, que se basa en el autoaprendizaje, en este proceso empleamos los dos primeros días.



Tercer día:

- Nos dedicamos a desarrollar el funcionamiento del programa, centrándonos en la manera en que los usuarios podrían tanto agregar como consultar ofertas a través de la API.



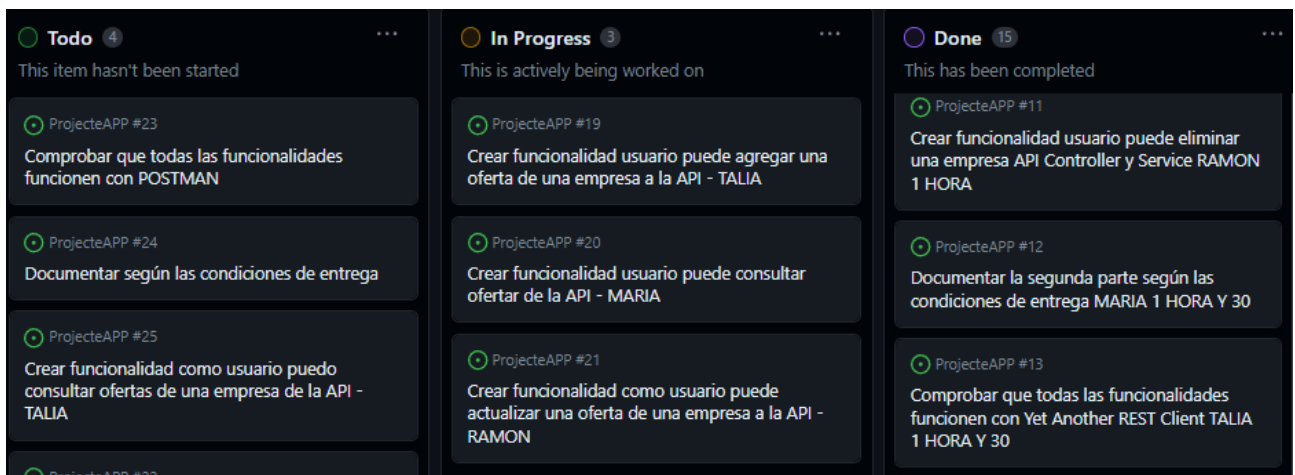
Cuarto día:

- Continuamos con la implementación de funciones, siguiendo una variedad de ejemplos y tutoriales que nos sirvieron de guía para iniciar el proceso de creación y definición de los distintos métodos.



Quinto día:

- Concluimos la implementación de las funciones para añadir y consultar ofertas, asegurándonos de su correcto funcionamiento mediante la herramienta Postman.



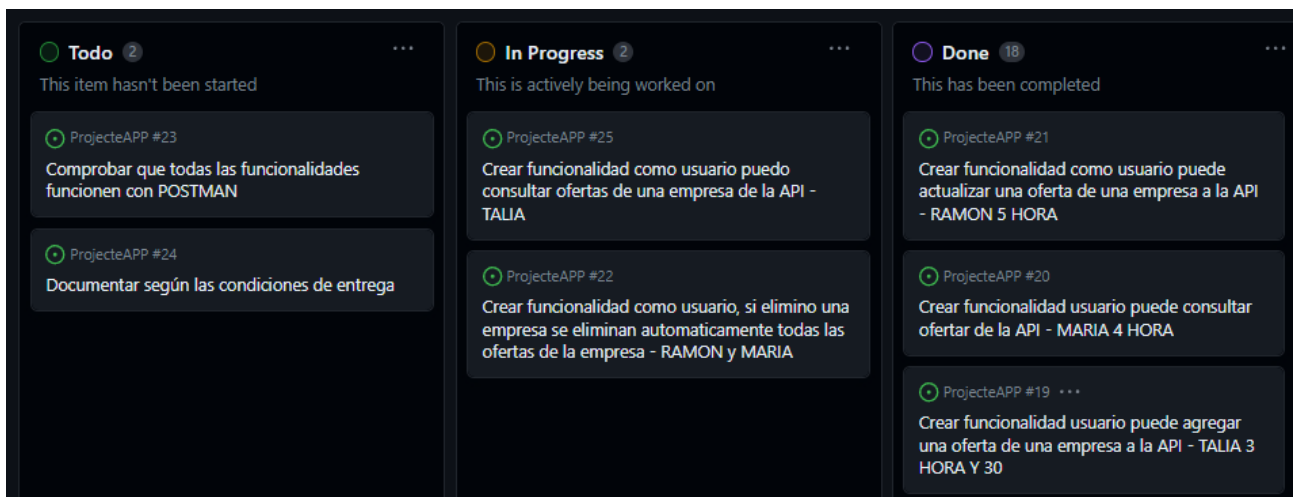
Sexto día:

- Hemos completado la programación de la función que permite a los usuarios actualizar distintas ofertas, nos adentramos en la fase inicial de la implementación de nuevas funciones, como la búsqueda de ofertas específicas de empresas y la capacidad de los usuarios para eliminarlas.



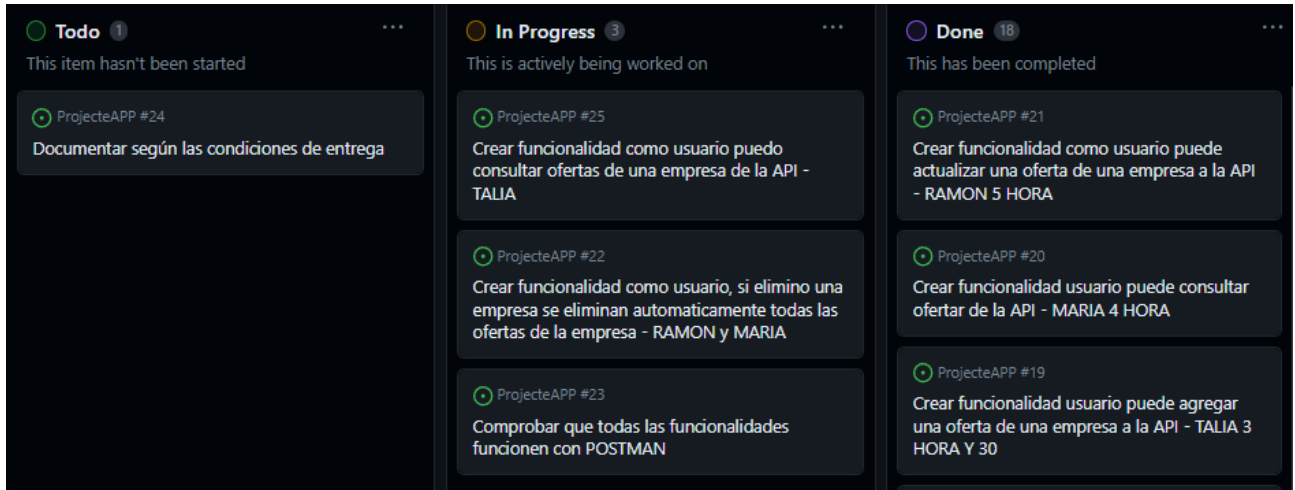
Septimo día:

- Persistimos en el desarrollo del código iniciado el día anterior, para mejorar y perfeccionar su funcionalidad.



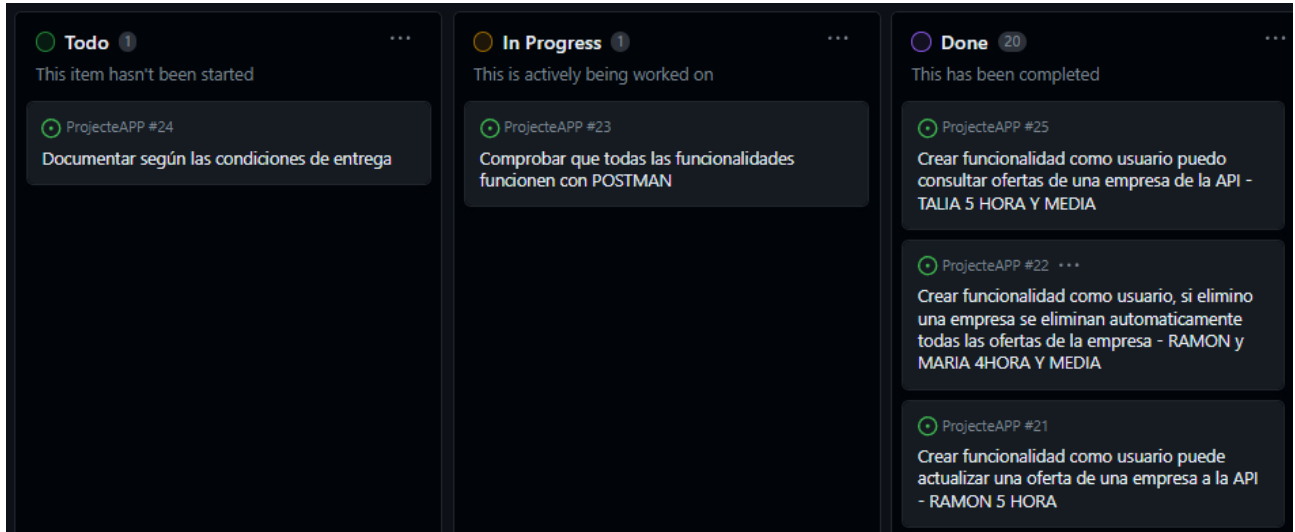
Octavo día :

- Hemos finalizado el desarrollo del código iniciado previamente y hemos dado inicio a las pruebas para verificar y asegurar su correcto funcionamiento.



Noveno día:

- Hemos concluido las pruebas con Postman y verificado que cada una de las funciones operan correctamente.



Decimo día:

- En el último día, repasamos todo el proyecto y nos aseguramos de tener una documentación completa. Así nos aseguramos de que todo esté en orden y cumpla las condiciones de entrega.

