Metodología Ágil Scrum

Alumnos: David Trujillo Carrero y Raúl Gutiérrez Merino 2ºDAW

1. Historia de usuario

1.1. Sprint I

Nombre de la historia: Registro del usuario en la aplicación	
Número: 1	**Usuario:** Todos los usuarios
Prioridad: Alta	**Riesgo en el desarrollo:** Alto
Estimación: 2 horas	**Sprint:** I
Programador responsable: David	

Descripción:
Pantalla para el registro de jugadores en nuestra aplicación. Esta pantalla se compone de un formulario con los siguientes campos: Nombre , primer apellido , segundo apellido , email , contraseña, confirmación de la contraseña y foto de perfil del usuario. Además se dispondrá de un botón para crear el usuario y un enlace a la página de inicio de sesión en caso de que el usuario ya esté registrado.

Observaciones:

- **Validaciones**
  - **Nombre:** La cadena debe comenzar con una letra mayúscula o minúscula, y puede contener entre 2 y 20 caracteres en total.
  - **Apellidos:** La cadena debe comenzar con una letra mayúscula o minúscula, y puede contener entre 2 y 30 caracteres en total.
  - **Email:** La cadena debe tener al menos 2 caracteres antes de un símbolo "@" seguido por al menos 4 letras minúsculas. Luego, debe haber un punto seguido por 2 o 3 letras minúsculas.
  - **Contraseña:** La cadena puede contener letras mayúsculas, minúsculas, dígitos y los caracteres especiales "*", "#", y "$". La longitud de la contraseña debe estar entre 6 y 12 caracteres.

- **Creación del usuario**
  Para crear el usuario se emplea una función asíncrona que llama a nuestra API a través del siguiente endpoint: POST http://127.0.0.1:8000/api/registrar.

Nombre de la historia: Login del usuario en la aplicación	
Número: 2	**Usuario:** Todos los usuarios
Prioridad: Alta	**Riesgo en el desarrollo:** Alto
Estimación: 2 horas	**Sprint:** I
Programador responsable: David	
Descripción: Pantalla para el inicio de sesión de usuario mediante sus credenciales. Esta pantalla se compone de un campo de email y contraseña. Esta es la primera página que verá el usuario nada más abrir la aplicación. Si el usuario aún no está registrado en la aplicación, dispone de un enlace para acceder al formulario de registro.	
Observaciones: - Para poder iniciar sesión se emplea una función asíncrona que llama a nuestra API a través del siguiente endpoint: POST http://127.0.0.1:8000/api/login	

Nombre de la historia: Pantalla de colaborador (Ver lotes)	
Número: 3	**Usuario:** Colaborador
Prioridad: Media	**Riesgo en el desarrollo:** Medio
Estimación: 4 horas	**Sprint:** I
Programador responsable: David	
Descripción: Pantalla de inicio que les aparecerá a todos los usuarios registrados una vez hayan iniciado sesión. Se compone de una tabla dinámica la cuál se va rellenando con todas las entregas realizadas por un usuario. Adicionalmente cuenta con una columna de acciones en la cuál el usuario podrá cancelar el envío de un lote (en este caso eliminando dicha entrega). A la izquierda de la página contamos con una barra lateral donde el usuario podrá agregar nuevos lotes.	
Observaciones: - Para rellenar la tabla dinámica con los lotes se emplea una función asíncrona que llama a nuestra API a través del siguiente endpoint: GET http://127.0.0.1:8000/api/lotes. - Para cancelar un lote se emplea otra función asíncrona que llama a la API usando el siguiente endpoint: DELETE http://127.0.0.1:8000/api/lotes{id} - Para la realización de las operaciones CRUD se emplean modales con formularios los cuáles se accionan cuando se pulsa el botón que los desencadena.	

Nombre de la historia: Pantalla de clasificador	
Número: 4	**Usuario:** Clasificador
Prioridad: Alta	**Riesgo en el desarrollo:** Alto
Estimación: 5 horas	**Sprint:** I
Programador responsable: David	

Descripción:

Cuando el usuario dispone de este rol, lo primero que verá será una tabla con la lista de componentes añadidos al sistema. Cada registro contará con una columna de acciones en la cuál el usuario podrá eliminar un componente o modificar datos de dicho componente. A la izquierda contamos con varios apartados:

- **Lotes:** Cuando desplegamos este menú, el usuario podrá acceder a la opción de clasificar lotes. Clasificar lotes se trata de otra página en la cuál el usuario podrá visualizar en una tabla los lotes que se han recibido por parte del colaborador y mediante la columna de acciones, el colaborador podrá empezar a despiezar el lote añadiendo el tipo de componente que ha obtenido con su cantidad. Cuando el clasificador comienza a despiezar un lote, el estado del lote cambia a recibido, de esta forma los colaboradores sabrán que su lote ya ha llegado. Una vez que el clasificador ha terminado de despiezar el lote, le cambiará el estado a clasificado.
- **Componentes:** Cuando desplegamos este menú, nos encontraremos con dos opciones:
  - **Ver lista de componentes:** Este apartado se trata de un enlace hacia la página donde se muestra la lista de componentes y es útil si el usuario por ejemplo se encuentra clasificando lotes y quiere volver a ver la lista de componentes.
  - **Agregar componente:** Agregar componente se trata de un modal con un formulario que permitirá al usuario agregar nuevos componentes al sistema.

Observaciones:

- Para añadir un componente se emplea una función asíncrona que llama a la API usando el siguiente endpoint: POST http://127.0.0.1:8000/api/componente{id}
- Para rellenar la tabla dinámica con los componentes se emplea una función asíncrona que llama a nuestra API a través del siguiente endpoint: GET http://127.0.0.1:8000/api/componente.
- Para editar un componente se emplea una función asíncrona que llama a la API usando el siguiente endpoint: PUT http://127.0.0.1:8000/api/componente{id}
- Para eliminar un componente se emplea una función asíncrona que llama a la API usando el siguiente endpoint: DELETE http://127.0.0.1:8000/api/componente{id}

Nombre de la historia: Pantalla de usuario del administrador	
Número: 5	**Usuario:** Administrador
Prioridad: Media	**Riesgo en el desarrollo:** Alto
Estimación: 8 horas	**Sprint:** I
Programador responsable: Raúl	
Descripción: Pantalla en la que tenemos una sidebar con navegación para ir hacia componentes y otro botón para añadir más usuarios. La zona central tiene una tabla con los usuarios que tengamos en la base de datos, con los datos de id, nombre, apellidos, correo y foto referente a los usuarios, además, tendremos en cada uno un botón para editar el usuario en específico y otro para borrarlo.	
Observaciones: - Para rellenar la tabla dinámica con los usuarios se crea una función asíncrona que llama a nuestra API a través del siguiente endpoint: GET http://127.0.0.1:8000/api/usuarios. - Para eliminar un usuario se creará otra función asíncrona que llama a la API usando el siguiente endpoint: DELETE http://127.0.0.1:8000/api/usuarios/{id} - Para editar un usuario se creará otra función asíncrona que llama a la API usando el siguiente endpoint: PUT http://127.0.0.1:8000/api/usuarios/{id} - Para la realización de las operaciones CRUD se emplean modales con formularios los cuáles se accionan cuando se pulsa el botón que los desencadena.	

Nombre de la historia: Pantalla de componentes	
Número: 6	**Usuario:** Administrador
Prioridad: Baja	**Riesgo en el desarrollo:** Alto
Estimación: 2 horas	**Sprint:** I
Programador responsable: Raúl	
Descripción: Pantalla en la que tenemos una sidebar con navegación para ir hacia usuarios y otro botón para añadir más componentes. La zona central tiene una tabla con los componentes que tengamos en la base de datos, con los datos de id, nombre y si es hardware o no referente a los usuarios, además, tendremos en cada uno un botón para editar el componentes en específico y otro para borrarlo	
Observaciones: - Para rellenar la tabla dinámica con los componentes se crea una función asíncrona	

que llama a nuestra API a través del siguiente endpoint: GET http://127.0.0.1:8000/api/componente.
- Para eliminar un componente se creará otra función asíncrona que llama a la API usando el siguiente endpoint: DELETE http://127.0.0.1:8000/api/componente/{id}
- Para editar un componente se creará otra función asíncrona que llama a la API usando el siguiente endpoint: PUT http://127.0.0.1:8000/api/componente/{id}
- Para la realización de las operaciones CRUD se emplean modales con formularios los cuáles se accionan cuando se pulsa el botón que los desencadena.
- Tenemos que detallar que: esta página es menos costosa respecto a otras, ya que es muy parecida a la de componentes de clasificador, la diferencia es que cada una tiene la navegación de los respectivos roles que tratan.

1.2. Sprint II

Nombre de la historia: Pantalla de colaborador (Entregar lote)	
Número: 7	**Usuario:** Colaborador
Prioridad: Alta	**Riesgo en el desarrollo:** Alto
Estimación: 3 horas	**Sprint:** II
Programador responsable: David	
Descripción: Página en la que un usuario puede indicar a través del mapa donde va a dejar el lote para entregarlo	
Observaciones: - Para mostrar el mapa y obtener las coordenadas se emplea la API de Google Maps, luego para poder elegir la zona donde queremos que se entregue el lote, usamos el marcador de Google al que se le añade un evento draggable para poder ser desplazado por la pantalla. - Una vez que hemos obtenido las coordenadas estas se muestran en un formulario con dos campos para posteriormente enviar la información a la base de datos empleando una función asíncrona ue llama a la API usando el siguiente endpoint: POST http://127.0.0.1:8000/api/lotes	

Nombre de la historia: Pantalla para ver joyas	
Número: 8	**Usuario:** Diseñador
Prioridad: Alta	**Riesgo en el desarrollo:** Alto
Estimación: 2 horas	**Sprint:** II
Programador responsable: Raúl	

Descripción:
En esta pantalla mostramos una tabla de joyas con todas las joyas traídas en un seeder, en la misma pantalla podemos ver su número de joya, su nombre y su foto, además también tenemos 2 acciones para utilizar, que son:

- **Modificar:** Saldrá un modal el cual nos dejara editar tanto el nombre como el nombre de la imagen (lo dejamos ya que en caso de cambiar de foto o añadir algo como cambiar el color se pueda hacer de forma más fácil)
- **Eliminar:** Saldrá un modal para eliminar esa joya de forma fácil (en caso de que se deje de fabricar esa joya en específico)

Observaciones:

- Para añadir una joya se emplea una función asíncrona que llama a la API usando el siguiente endpoint: POST http://127.0.0.1:8000/api/joya.
  Esto se realizará en el sidebar global para diseñadores.
- Para rellenar la tabla dinámica con las joyas se emplea una función asíncrona que llama a nuestra API a través del siguiente endpoint: GET http://127.0.0.1:8000/api/joya.
- Para editar una joya se emplea una función asíncrona que llama a la API usando el siguiente endpoint: PUT http://127.0.0.1:8000/api/joya{id}
- Para eliminar una joya se emplea una función asíncrona que llama a la API usando el siguiente endpoint: DELETE http://127.0.0.1:8000/api/joya{id}

Nombre de la historia: Pantalla recetas	
Número: 9	**Usuario:** Diseñador
Prioridad: Media	**Riesgo en el desarrollo:** Alto
Estimación: 2 horas	**Sprint:** II
Programador responsable: Raúl	

Descripción:
En esta pantalla mostramos una tabla de recetas con todas las recetas traídas en un seeder, en la misma pantalla podemos ver su número de receta, a qué joya hace referencia y una

descripción, además también tenemos 2 opciones para utilizar, que son: - **Modificar:** Saldrá un modal el cual nos dejará editar tanto a que joya hace referencia como la descripción. - **Eliminar:** Saldrá un modal para eliminar esta receta de forma fácil (en caso de que se deje de fabricar esa joya de esa forma)
Observaciones: - Para añadir una receta se emplea una función asíncrona que llama a la API usando el siguiente endpoint: POST http://127.0.0.1:8000/api/recetas. Esto se realizará en el sidebar global para diseñadores. - Para rellenar la tabla dinámica con las recetas se emplea una función asíncrona que llama a nuestra API a través del siguiente endpoint: GET http://127.0.0.1:8000/api/recetas. - Para editar una receta se emplea una función asíncrona que llama a la API usando el siguiente endpoint: PUT http://127.0.0.1:8000/api/recetas{id} - Para eliminar una receta se emplea una función asíncrona que llama a la API usando el siguiente endpoint: DELETE http://127.0.0.1:8000/api/recetas{id}

Nombre de la historia: Pantalla ingredientes	
Número: 10	**Usuario:** Diseñador
Prioridad: Media	**Riesgo en el desarrollo:** Alto
Estimación: 2 horas	**Sprint:** II
Programador responsable: David	
Descripción: En esta pantalla mostramos una tabla de ingredientes la cuál contiene todos los componentes que conforman cada una de las joyas.	
Observaciones: - Para rellenar la tabla dinámica con las recetas se emplea una función asíncrona que llama a nuestra API a través del siguiente endpoint: GET http://127.0.0.1:8000/api/ingredientes.	

Nombre de la historia: Pantalla de fabricación de joyas	
Número: 10	**Usuario:** Diseñador
Prioridad: Media	**Riesgo en el desarrollo:** Alto
Estimación: 5 horas	**Sprint:** II

Programador responsable: David
Descripción: En esta pantalla mostramos una tabla de ingredientes la cuál contiene todos los componentes que conforman cada una de las joyas.
Observaciones: - Para rellenar la tabla dinámica con las recetas se emplea una función asíncrona que llama a nuestra API a través del siguiente endpoint: GET http://127.0.0.1:8000/api/ingredientes.

Nombre de la historia: Pantallas anexas	
Número: 11	**Usuario:** Todos
Prioridad: Muy alta	**Riesgo en el desarrollo:** Alto
Estimación: 14 horas	**Sprint:** I y II
Programador responsable: David y Raúl	
Descripción: Hemos creado: 4 sidebar(1 para cada rol), un nav y una tarjeta de bienvenida. - **Sidebar de Administración:** En este sidebar trabajamos tanto para navegar entre los usuarios de admin y los componentes del admin, además de añadir usuarios y componentes **(Creado por Raúl)** - **Sidebar de Colaborador:** En este sidebar trabajamos tanto para navegar entre ver lote, además de entregar lote **(Creado por David)** - **Sidebar de Clasificador:** En este sidebar trabajamos tanto para navegar entre los lotes para despiezar y los componentes, además de añadir componente **(Creado por David)** - **Sidebar de Diseñador:** En este sidebar trabajamos tanto para navegar entre ir a la página para crear joyas, ver las joyas, ver recetas y ver ingredientes **(Creado por Ambos)** - **Nav:** El nav es una pequeña cabecera la cual usamos para movernos entre roles y ver nuestro usuario **(Creado por David)** - **Tarjeta:** Muestra el nombre del usuario y le da estilo a las páginas **(Creado por David)**	

2. Product Backlog y Sprint Backlog

Product Backlog	Sprint Backlog
- Registro del usuario en la aplicación - Login del usuario en la aplicación - Pantalla de colaborador (Ver lotes) - Pantalla de clasificador - Pantalla de componentes - Pantallas anexas	Sprint I
- Pantalla de colaborador (Entregar lote) - Pantalla para ver joyas - Pantalla recetas - Pantalla ingredientes - Pantalla de fabricación de joyas - Pantallas anexas	Sprint II

3. Estimación total

Sprint	Tiempo estimado
Sprint I	30 horas
Sprint II	21 horas
Total	51 horas