ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE SISTEMAS

UNIDAD DE TITULACIÓN

DESARROLLO DE UNA APLICACIÓN WEB PARA LA GESTIÓN DE LOS TRABAJOS DE TITULACIÓN DE LA FACULTAD DE SISTEMAS (FIS) DENTRO DE LA ESCUELA POLITÉCNICA NACIONAL (EPN)

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

BOLÍVAR ANDRÉS ROMÁN CABRERA
bolivar.roman@epn.edu.ec

Director: VICENTE ADRIÁN EGÜEZ SARZOSA
adrian.eguez@epn.edu.ec

Codirector: JOSE FRANCISCO LUCIO NARANJO
jose.lucio@epn.edu.ec

QUITO, 2023

APROBACIÓN DEL DIRECTOR

Como director del trabajo de titulación DESARROLLO DE UNA APLICACIÓN WEB PARA LA GESTIÓN DE LOS TRABAJOS DE TITULACIÓN DE LA FACULTAD DE SISTEMAS DENTRO DE LA EPN desarrollado por BOLÍVAR ANDRÉS ROMÁN CABRERA, estudiante de la carrera INGENIERÍA EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN habiendo supervisado la realización de este trabajo y realizado las correcciones correspondientes, doy por aprobada la redacción final del documento escrito para que prosiga con los trámites correspondientes a la sustentación de la defensa oral.

VICENTE ADRIAN EGÜEZ SARZOZA

DIRECTOR

APROBACIÓN DEL CODIRECTOR

Como codirector del trabajo de titulación DESARROLLO DE UNA APLICACIÓN WEB PARA LA GESTIÓN DE LOS TRABAJOS DE TITULACIÓN DE LA FACULTAD DE SISTEMAS DENTRO DE LA EPN desarrollado por BOLÍVAR ANDRÉS ROMÁN CABRERA, estudiante de la carrera INGENIERÍA EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN habiendo supervisado la realización de este trabajo y realizado las correcciones correspondientes, doy por aprobada la redacción final del documento escrito para que prosiga con los trámites correspondientes a la sustentación de la defensa oral.

Firmado electrónicamente por:
JOSE FRANCISCO LUCIO NARANJO

JOSE FRANCISCO LUCIO NARANJO

CODIRECTOR

DECLARACIÓN DE AUTORÍA

Yo, BOLÍVAR ANDRÉS ROMÁN CABRERA declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Escuela Politécnica Nacional puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

BOLÍVAR ANDRÉS ROMÁN CABRERA

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

RESUMEN

Esta tesis describe el desarrollo de un sistema de software para el seguimiento de los pasos de un proceso utilizando la metodología SCRUM y Auth0 como proveedor de identidad. El sistema fue diseñado específicamente para ser utilizado por la facultad de Ingeniería de Sistemas de la Escuela Politécnica Nacional. El sistema de software fue construido utilizando tecnologías web modernas como AngularJS y Node.js, y su arquitectura fue diseñada para permitir un mayor desarrollo y expansión de las capacidades del sistema. La arquitectura del sistema incluye múltiples módulos, como módulos de autenticación, base de datos y seguimiento de procesos, cada uno con interfaces bien definidas para facilitar futuras modificaciones y adiciones. El proceso de desarrollo también incluyó pruebas unitarias y funcionales para garantizar la fiabilidad y escalabilidad del sistema. La tesis concluye con una evaluación de la arquitectura del sistema y su potencial para futuros desarrollos. El éxito del desarrollo de este sistema de software demuestra el potencial de las metodologías ágiles y los proveedores de identidad para crear soluciones de software innovadoras y eficaces para contextos institucionales específicos. La arquitectura del sistema ofrece una base sólida para su posterior desarrollo y personalización, permitiendo potencialmente su adaptación a las necesidades de seguimiento de procesos de otras instituciones académicas o industrias.

Palabras clave: seguimiento de pasos de un proceso, SCRUM, Auth0, arquitectura de sistema.

ABSTRACT

This thesis describes the development of a software system for tracking process steps using the SCRUM methodology and Auth0 as the identity provider. The system was specifically designed for use by the Systems Engineering faculty of the National Polytechnic School of Ecuador. The software system was built using modern web technologies such as AngularJS and Node.js, and its architecture was designed to enable further development and expansion of the system's capabilities. The system architecture includes multiple modules, such as authentication, database, and process tracking modules, each with well-defined interfaces to facilitate future modifications and additions. The development process also included unit testing and functional testing to ensure that the system was reliable and scalable. The thesis concludes with an evaluation of the system's architecture and its potential for future development. The successful development of this software system demonstrates the potential of agile methodologies and identity providers to create innovative and effective software solutions for specific institutional contexts. The architecture of the system offers a solid foundation for further development and customization of the system, potentially enabling it to be adapted to the process tracking needs of other academic institutions or industries.

Keywords: process tracking, SCRUM, Auth0, system architecture.

1 INTRODUCCIÓN

1.1 Planteamiento del problema

La Escuela Politécnica Nacional (EPN) en su reglamento de régimen académico; capítulo XII, artículo 83, establece los mecanismos de graduación y titulación, que son: trabajo de titulación o examen complexivo. En las carreras de nivel tecnológico, superior y equivalentes es considerado trabajo de titulación:

- Artículo académico
- Proyecto de investigación
- Proyecto integrador [1]

La implementación de un sistema que organice y gestione información y documentos permite que una institución pueda ser más eficiente en sus actividades y ahorrar recursos. Cuando los trámites que se llevan de forma tradicional, es decir, de forma física, tienden a ser caóticos debido a un bajo nivel de control de documentación y existe el riesgo de pérdida temporal o permanente de información. [2]

Debido a la complejidad de los trámites que se deben realizar, algunas de las veces no están completamente claras las especificaciones de los requisitos necesarios para su realización, esto ralentiza el tiempo que se requiere para completar un trámite. A nivel nacional, el Estado ecuatoriano propuso un plan de simplificación de trámites, en donde se definen estrategias que promueven la incorporación de la tecnología y estimulan los modelos de gestión existente para tener mejor control sobre los mismos. [3]

Por los motivos descritos anteriormente, el presente proyecto brinda una solución que permita tener un mayor control sobre el proceso de gestión de trabajos de titulación, permita a los entes involucrados en el proceso informarse cuando sea requerido, así como otros beneficios indirectos incluyendo la reducción en el uso de papel en la institución.

1.2 Objetivo general

Desarrollar una aplicación web para la Facultad de Sistemas de la EPN para la gestión de los trabajos de titulación.

1.3 Objetivos específicos

- Diseñar una arquitectura para el aplicativo web.
- Recopilar las características, requerimientos y alcance del sistema de gestión de trabajos de titulación.
- Realizar pruebas funcionales para el sistema.
- Definir los roles, artefactos y componentes para la implementación del sistema.

1.4 Organización general

Este documento ha sido distribuido en seis secciones, en la primera parte se encuentra la introducción y objetivos de esta tesis. Después, en la segunda parte se puede encontrar todas las consideraciones técnicas y teóricas para el desarrollo. En la tercera parte, se encuentra el desarrollo de la metodología, así como detalles sobre la implementación del sistema que desarrolla esta tesis. En la cuarta parte, se encuentra el resumen de los resultados y ejecución de pruebas. En la quinta sección están desarrolladas las conclusiones y consideraciones finales de este documento para finalmente pasar a la sexta sección en donde se encuentran las referencias bibliográficas y anexos correspondientes.

2 CONSIDERACIONES TÉCNICAS Y TEÓRICAS FUNDAMENTALES

2.1 Conceptos técnicos

Aplicación Web

Es un sistema de software que tiene un estado determinado de acuerdo al modelo de negocio y su interfaz es entregada al usuario mediante un sistema web. La arquitectura general de una aplicación web es de un sistema cliente/servidor aclarando algunas distinciones. Entre las más importantes tenemos que la configuración de componentes se realiza de lado del servidor y normalmente los clientes no requieren ninguna configuración especial. El protocolo principal de comunicación de un sistema web es HTTP, que es un protocolo sin estado diseñado para ser robusto y tolerante a fallos. [4]

Patrón de diseño modelo-vista-controlador (MVC)

El diseñar un sistema separando las responsabilidades de este permite a los desarrolladores de software centrarse en tareas específicas cuando se desarrolla un sistema. En este esquema cada parte desempeña una tarea determinada y no comparte funcionalidad entre componentes. [5]

La esencia de la aplicación, es decir, el dominio, es llamado modelo, en un sistema orientado a objetos, este consiste en el conjunto de clases que soportan el modelo de negocio o problema que resuelve el sistema. Los modelos poseen un largo ciclo de vida debido a que el problema que resuelve la mayoría de las veces se mantiene a lo largo del tiempo.

Las interfaces que están directamente relacionadas con el modelo del sistema constituyen las vistas que componen este modelo. En un sistema orientado a objetos estas son las diferentes ventanas que permiten la entrada de información y desarrollo de acciones por parte del usuario.

Los controladores son objetos que permiten manipular una vista. De una manera más simplificada se podría decir que los controladores manejan las entradas y las vistas manejan la salida del sistema. [6]

En el ámbito de una aplicación web, este patrón de diseño es aplicable de tal manera en la que las interfaces y su interacción con el usuario se encuentran en el front-end y la lógica del negocio o problema y manejo de datos se encuentra en el back-end, se puede encontrar más información sobre este modelo en la figura 1.

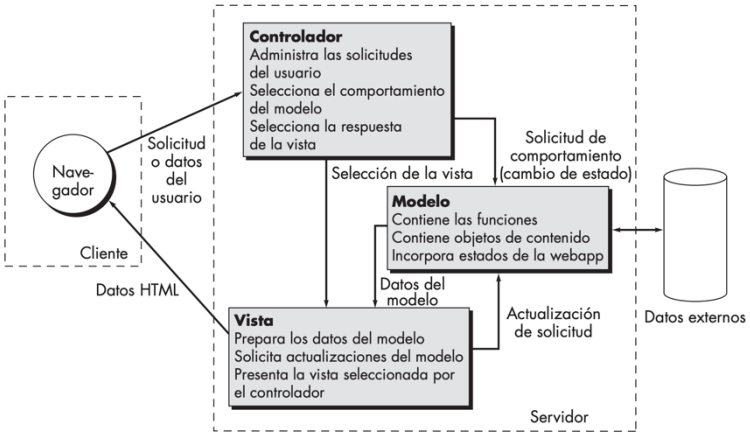

Figura 1. Arquitectura MVC

[7]

Arquitectura de API

Para realizar un diseño de API, se deben tomar en cuenta varios factores que son clave, entre los más importantes tenemos:

- Consistencia: este aspecto hace referencia a la coherencia entre estructuras de datos que se usen, representaciones, URI, mensajes de error y comportamiento de las API. Para los consumidores es más fácil trabajar de esta manera si la API se comporta de una manera intuitiva.
- Reusabilidad: el principal objetivo de este aspecto es tener en mente que un API no debe ser desarrollada para un cliente en específico, ya que esta puede ser reutilizada por cualquier cliente e incluso otra API. La funcionalidad común puede ser refactorizada en librerías y estas pueden ser reutilizadas.
- Personalización: cuando los clientes de un API no son un grupo homogéneo o tienen pocas características en común, se debe garantizar la posibilidad de personalización para que cada cliente del API pueda satisfacer sus necesidades individuales. Esta característica puede ser contradictoria con la anterior, debido a que se establece que no se debe desarrollar un API para un cliente específico, pero también se debe personalizar para cada consumidor. En realidad, se pueden tomar los dos enfoques al mismo tiempo. [8]

Aplicación de una sola página (Single Page Application) SPA

Un SPA se puede definir como una manera de hacer desarrollo web en la que toda la aplicación se aloja en una sola página. Además, no se actualiza toda la página después de que se cargó la aplicación, la lógica de presentación se carga primero y se presenta según el intercambio de vistas dentro de las regiones donde se carga

otro contenido. Es común que este tipo de páginas web se comuniquen con el servidor de manera asíncrona y comúnmente con texto en formato JSON. [9]

SCRUM

Es un marco de trabajo ligero que ayuda a las personas, equipos y organizaciones a generar valor a través de soluciones adaptativas para problemas complejos. Para la implementación de este marco de trabajo se requiere:

1. Un Product Owner que ordena el trabajo para un problema determinado en un Product Backlog.
2. Un equipo SCRUM que transforma una parte del trabajo en un incremento de valor durante un Sprint.
3. Un equipo SCRUM y sus interesados para la inspección y revisión de resultados y ajustes para un siguiente sprint.
4. Repetir todas las acciones anteriores.

En la definición de SCRUM existen varios vacíos intencionales, esto se debe a que este marco de trabajo no supone una guía completa de implementación de este para un determinado problema. En lugar de que el marco de trabajo proporcione instrucciones detalladas sobre el trabajo que se debe hacer, las reglas del equipo SCRUM deben basarse en las interacciones y relaciones de este. [10] Los equipos de trabajo dentro de SCRUM están conformados por 3 entes con sus respectivos roles:

- **Scrum Master**: conoce sobre la aplicación del marco de trabajo y guía al equipo de desarrollo en la creación de valor.
- **Product Owner**: responsable de maximizar el valor del producto resultando del trabajo del equipo de desarrollo.
- **Equipo de desarrollo**: son los responsables de crear el valor que se entrega cada sprint. Ellos determinan el cómo y el qué de los requerimientos del product owner. [10]

Scrum está compuesto de actividades y artefactos. [10] Se detalla a continuación sobre su uso:

- **Product Backlog**: es una lista ordenada de lo que se necesita hacer para sumarle valor al producto. Es la única fuente de trabajo para el equipo de desarrollo. Normalmente se forma esta lista de trabajo durante el desarrollo de un sprint para ser implementado un siguiente sprint.

- **Sprint**: se puede definir como un evento de duración fija de un mes o menos para crear valor al producto. Un nuevo sprint comienza automáticamente cuando el anterior ha terminado.
- **Sprint Planning**: es una reunión que establece objetivos y el trabajo que se realizará durante el sprint. Este plan se crea trabajando colaborativamente con todo el equipo Scrum.
- **Sprint Review**: es una reunión en donde participa el equipo scrum con las partes interesadas para revisar si se cumplieron los objetivos planteados al inicio del sprint.
- **Sprint Retrospective**: es una reunión en donde el equipo Scrum analiza los aciertos, errores y buenas prácticas que se mantuvieron durante el sprint.
- **Daily Scrum**: es una reunión diaria cuya duración máxima es de 15 minutos. El propósito principal de esta reunión es revisar si es que existen inconvenientes que impidan el desarrollo del sprint.
- **Historia de usuario**: Es una explicación general e informal de una función de software escrita desde la perspectiva del usuario final. Su propósito es articular cómo una característica del software agregará valor al producto. [11]

Para este proyecto integrador, se ha definido el formato para detallar las historias de usuario en la tabla 1. Estas historias serán parte del product backlog que se desarrollará posteriormente.

Historia de Usuario			
Descripción	"Como [persona], [quiero], [para]"		
Prioridad		**Costo**	
Criterios de aceptación			
Tareas a realizar			

Tabla 1. Formato para historias de usuario

A continuación, se brinda una breve descripción sobre los campos de la tabla 1:

- **Historia de usuario**: contiene un código único para cada historia de usuario.
- **Descripción**: consiste en una descripción desde el punto de vista del usuario, asumiendo su rol en el sistema, su requisito y su propósito.
- **Prioridad**: sirve para asignar una prioridad de desarrollo al momento de definir las historias que se desarrollarán en un sprint.
- **Costo**: es un puntaje que estima de manera aproximada el esfuerzo que tomará el desarrollo de una historia determinada. Este puntaje se asocia con un tiempo acorde al ritmo de desarrollo.
- **Criterios de aceptación**: consiste en un conjunto de criterios que se deben cumplir para que una historia de usuario cumpla el concepto de done.
- **Tareas a realizar**: conjunto de todas las actividades que se deben desarrollar para que una historia de usuario se complete.

Definición de Done

La definición de done puede establecerse como una descripción formal de las características funcionales y de calidad que debe cumplir un incremento. Una vez que un incremento ha alcanzado este estado, puede ser presentado para un Sprint Review, caso contrario el elemento que no haya alcanzado este estado permanecerá en el backlog para ser considerado en un futuro sprint. [10]

2.2 Herramientas de desarrollo

Frameworks de desarrollo

- **NestJS**: es un marco de trabajo para construir aplicaciones eficientes y escalables del lado del servidor de Node.js. Está construido con TypeScript y es totalmente compatible con este. Combina elementos de programación orientada a objetos, programación funcional y programación reactiva funcional. Este proporciona un nivel de abstracción por encima de los marcos de trabajo comunes de Node.js y también expone sus APIs directamente al desarrollador. Esto permite incorporar módulos de terceros que estén disponibles para la plataforma principal. [12]

- **Angular**: es un marco de diseño de aplicaciones y una plataforma de desarrollo para crear SPAs (single-page application) que son eficientes y sofisticadas. Está construida sobre el lenguaje TypeScript y como plataforma incluye un marco de

trabajo basado en componentes para construir aplicaciones web que son escalables. También se incluye una colección de bibliotecas que cumplen varias necesidades, incluyendo el enrutamiento, gestión de formularios, comunicación cliente-servidor, etc. [13]

- **Auth0**: es una solución que permite añadir autenticación y autorización a diferentes tipos de aplicaciones, además evita el coste, tiempo y riesgo que conlleva una implementación propia para autenticar y autorizar a los usuarios de una aplicación. [14, p. 0] Se usa principalmente un token de acceso y un token de usuario en donde se almacena información relacionada al usuario que permite verificar su identidad ante una aplicación. El token mencionado anteriormente se añade a una petición HTTPS que es verificada por el host de autenticación (Auth0) y luego redirigida a otro host que manejará la petición según los permisos configurados en el diseño de la aplicación. [15, p. 0]

Lenguajes de programación

- **TypeScript**: es un lenguaje de programación desarrollado por Microsoft en 2012. Su principal objetivo es mejorar la productividad del desarrollo de aplicaciones complejas. Fue lanzado como un superconjunto de JavaScript. Esto significa que cualquier código válido para JavaScript también es válido para TypeScript. Entre las principales características de este lenguaje, tenemos que se incluye opciones para la tipificación y programación orientada a objetos. [16]

Base de datos

- **MongoDB**: se define a sí misma como una base de datos de documento con la escalabilidad y flexibilidad necesaria para hacer consultar y crear los índices que sean necesarios. Este es un motor de base de datos no relacional, es decir, NoSQL. Además, en MongoDB se guardan los datos en documentos en formato JSON, su principal ventaja es que los nombres de estos campos JSON pueden variar entre diferentes documentos y las estructuras de los datos pueden ser cambiadas fácilmente a lo largo del tiempo. [17]

Entorno de desarrollo

- **Webstorm**: Mejor conocido como IDE por sus siglas en inglés (integrated development environment). Permite codificar en JavaScript y tecnologías

relacionadas con esta, como TypeScript, React, Vue, Angular, Node.js, etc. Este IDE hace el del desarrollo una experiencia más cómoda y ayuda a automatizar tareas complejas con facilidad. [18]

Control de versiones

- **GitHub**: se podría definir como un sitio web basado en la nube que guarda y administra código. Esta herramienta ayuda a darle seguimiento al código usando un sistema de control de versiones específico llamado Git creado inicialmente por Linus Tolvards en 2005. Este sistema de control de versiones se vale de la creación de ramas que representan bifurcaciones de un estatus del código en donde se registran cambios en una línea de tiempo. Estos cambios pueden luego ser unidos con otras ramas, revertidos, etc. Ayudándonos a mantener registro del desarrollo realizado. [19]

3 METODOLOGIA

3.1 Equipo Scrum

- Product Owner: Nancy Naranjo (Representante de secretaría)
- Scrum Master: Ing. Adrian Egüez
- Equipo de desarrollo: Bolívar Román

3.2 Requerimientos iniciales

Para definir los requisitos del sistema se realizó una entrevista al product owner de este equipo, en este caso, Nancy Naranjo. Ella fue escogida porque es la encargada de los grados dentro de la facultad de sistemas. Durante el desarrollo de la entrevista, se preguntó por los principales aspectos del proceso de graduación de los estudiantes de la facultad, además como posibles impedimentos y bloqueos que se presentan durante el desarrollo del proceso tradicional. Finalmente, se resumen los requisitos obtenidos en la tabla 2.

Módulo	Requerimientos
Administrador	Gestionar a los encargados de un proceso y/o requerimiento dentro del proceso general de graduación. Se debe brindar información sobre cómo realizar un paso del proceso.
	Gestionar a los procesos que existen dentro de la facultad por los cuales se puede graduar un estudiante, entre los que se pudo revisar tenemos: examen complexivo, tesis, proyecto integrador.
	Gestionar las diferentes carreras que existen dentro de la facultad de sistemas, actualmente existen: ingeniería en sistemas informáticos y de computación (en cierre), ingeniería en ciencias de la computación, ingeniería de software.
	Gestionar los diferentes usuarios del sistema, debe existir un usuario administrador encargado de registrar las diferentes estudiantes, carreras y procesos, así como los pasos de cada proceso que por conveniencia serán llamados subprocesos a lo largo de este documento. Los estudiantes pueden acceder al sistema por registro propio o por invitación por parte del administrador.

Módulo	Requerimientos
Estudiante	Mostrar la información de la carrera y proceso que estoy siguiendo, así como información de cada subproceso que detallará las instrucciones que se deben seguir para completar un determinado subproceso.

Tabla 2. Resumen de necesidades obtenidas de la entrevista

3.3 Prototipo del sistema

Una vez identificadas las necesidades del product owner, se procede a diseñar un prototipo en la herramienta de diseño Figma. Para lograr esta tarea, se ha analizado a detalle cada una de las características que se pudo obtener de la entrevista. El enlace del prototipo completo puede ser encontrado en la sección de anexos de este documento.

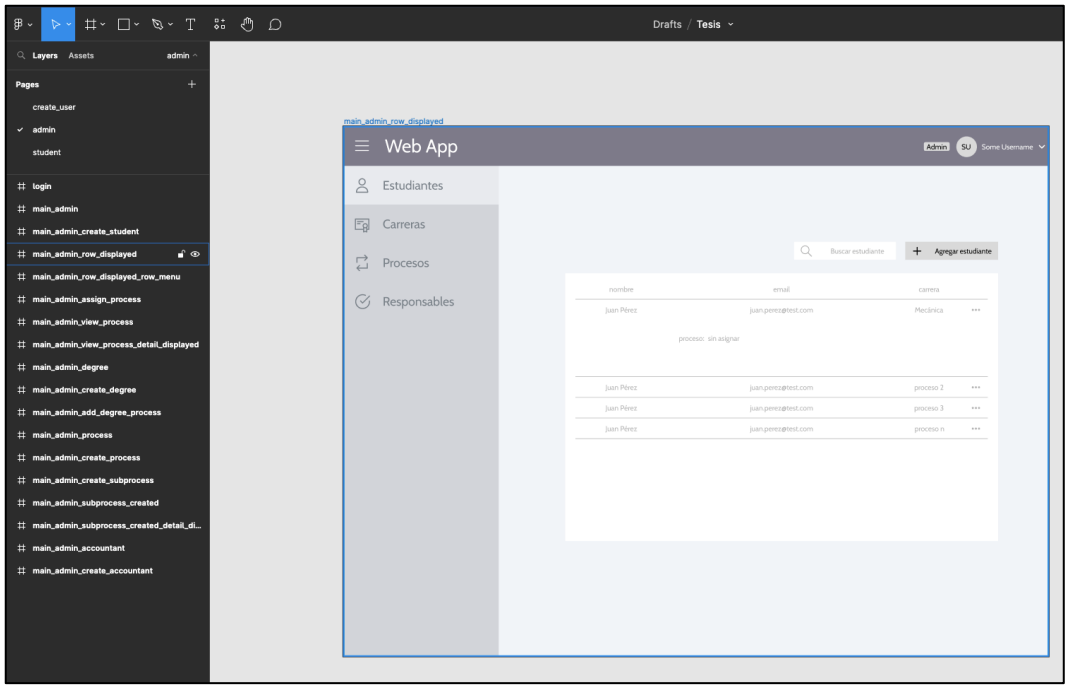

Figura 2. Vista general del prototipo del sistema

Como se puede observar en la figura 2, la herramienta permite diseñar una interfaz de manera fiel a la realidad. Después de que el prototipo del sistema fue revisado y aprobado por parte del product owner, se procede a definir aspectos y detalles técnicos sobre la implementación del sistema.

3.4 Arquitectura de la aplicación

Como se puede observar en la figura 3, para definir la arquitectura del sistema, así como las herramientas que se usarán para su desarrollo, se han tomado varias consideraciones:

- Se debe gestionar eficientemente y de manera segura a los usuarios del sistema proveyéndoles opciones de inicio de sesión, así como la facilidad de acceso al aplicativo desde diferentes plataformas.
- El sistema puede crecer en volumen de usuarios rápidamente considerando el número de estudiantes en la facultad de sistemas, por lo que la gestión de usuarios debe ser escalable sin perder ninguna de las funcionalidades mencionadas anteriormente.
- El módulo desarrollado para los estudiantes será informativo, es decir, solamente permitirá visualizar la información de carrera y procesos que actualmente se encuentra siguiendo.

Una vez establecido el alcance y aspectos principales sobre el sistema, se empleará a Auth0 como principal gestor de identidad para el sistema, debido a que integra todas las funcionalidades requeridas, además, esto permite reducir las tareas de desarrollo en la creación y gestión de todos los usuarios del sistema, permitiendo dirigir toda la atención en el desarrollo del sistema en sí.

Además, se optó por crear un API, esta decisión se ha tomado ya que la principal forma de acceso de los estudiantes sería desde un dispositivo móvil. Sin embargo, debido a que la mayor parte de tareas del sistema deben ser realizadas por el usuario administrador que, en un escenario real, sería un encargado de secretaría dentro de una facultad; se realizará una aplicación web, ya que esta permite el manejo de forma eficiente al administrador. La creación de un API permitirá en la posteridad la implementación de una aplicación móvil que cumpla las mismas funciones pero que facilite el uso y acceso a los estudiantes.

El desarrollo del aplicativo web se hará usando Angular debido a que se deben crear múltiples pantallas que permitan una buena interacción con los usuarios, así como el despliegue de formularios, recarga y actualización de datos en el background de la aplicación, así como reflejar cambios después de una operación sobre un registro en la base de datos.

Para el almacenamiento de datos, se optó por MongoDB, esto debido a que resulta más sencillo el almacenamiento de los datos de la aplicación. Se tienen múltiples relaciones entre las entidades del sistema, que en una base de datos relacional incrementarían la dificultad de su implementación. Además, MongoDB permite almacenar los datos en un clúster, permitiendo su acceso desde cualquier dispositivo conectado a internet. Esto hace que los datos no dependan de otro servidor y estén disponibles siempre que se requieran.

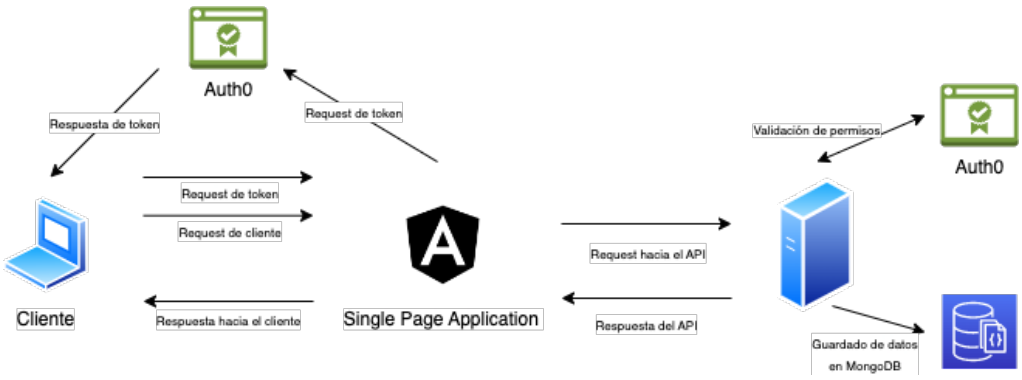

Figura 3. Arquitectura de aplicación

Se deben tomar en cuenta algunos aspectos para el desarrollo, entre los más importantes tenemos:

- Debido a que Auth0 es un proveedor de identidades verificado, todas las aplicaciones deben correr sobre HTTPS. Es decir, sobre una conexión segura. Para ello, se debe generar e instalar un certificado local en el ambiente de desarrollo.
- El SDK que Auth0 provee para realizar el desarrollo cuenta con un sinnúmero de funcionalidades, es conveniente revisar implementaciones previas antes de usarlo en el código para conocer cómo funciona.
- Al usarse Auth0 para autenticar y autorizar a los usuarios, implícitamente se está usando el esquema de control de acceso por roles RBAC por sus siglas en inglés.
- Auth0 se vale de JWT o JSON Web Token para almacenar información de inicio de sesión, permisos y meta data del usuario. Revisar la documentación de Auth0 para conocer su implementación más a detalle.

3.5 Roles del sistema

Para definir los roles, se han identificado a quienes interactuarán con el sistema, para ello, tenemos únicamente a 2 usuarios principales, un administrador y un estudiante, a continuación, se detallan las funciones que cada uno cumplirá dentro del sistema:

- **Administrador**

  El administrador del sistema tiene acceso total a todas las funcionalidades del sistema. Puede visualizar, agregar, editar y eliminar datos en todos los módulos del sistema. En la figura 4 se pueden observar algunos de los permisos que han sido otorgados al rol “admin”.

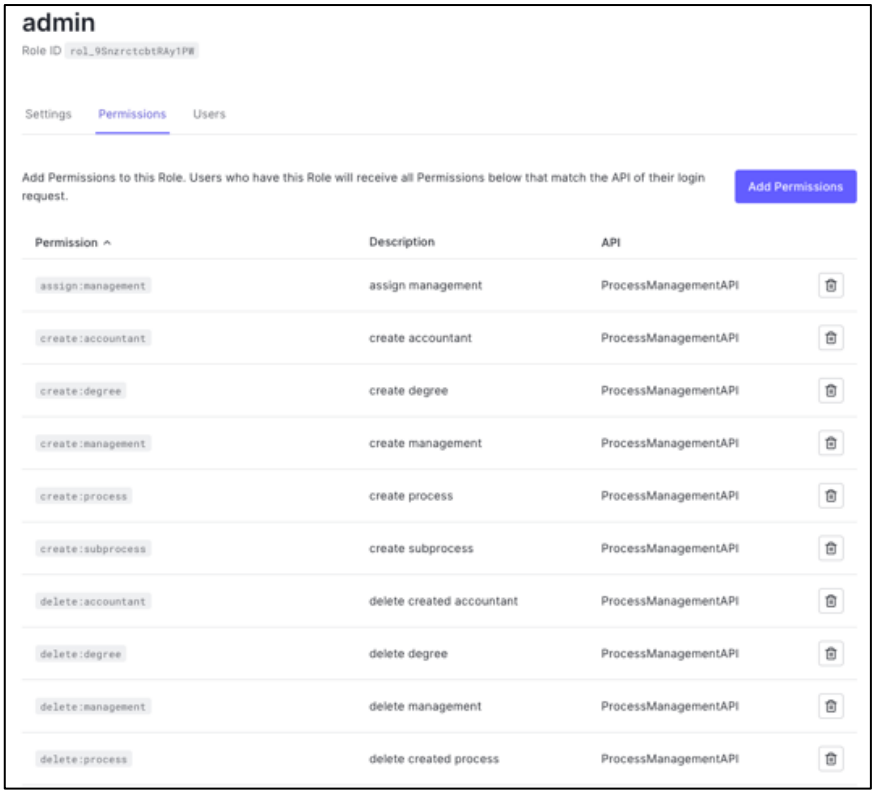

Figura 4. Permisos asignados al rol "admin"

- **Estudiante**

  El estudiante solamente tiene acceso a la visualización de la información que proporciona el API, así como la asignación de un proceso a sí mismo. En la figura 5 se pueden observar los permisos asignados al rol “student”.

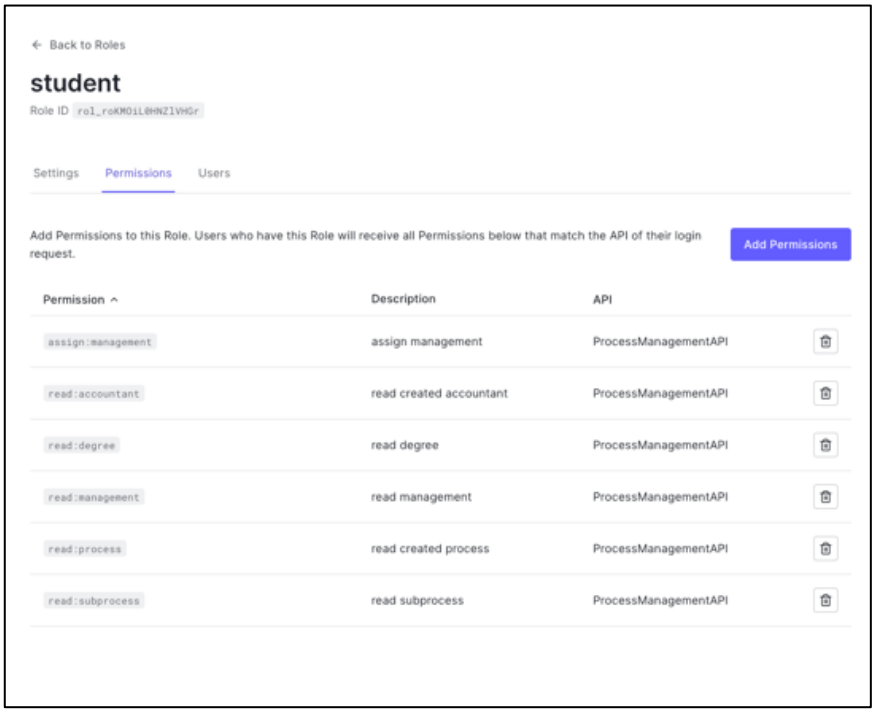

Figura 5. Permisos asignados al rol "student"

3.6 Product Backlog

El product backlog está formado por las historias que se detallan a continuación. También se especifica la nomenclatura empleada en la construcción de la tabla:

- H: historia de usuario
- F: funcional
- NF: no funcional
- 001: número de historia

Con la finalidad de estimar el esfuerzo que llevará completar las historias de usuario del backlog, el equipo debe determinar una escala para asignar estimaciones. La escala más común está basada en la serie de Fibonacci modificada propuesta por Mike Cohn [20], que tiene los elementos definidos en la tabla 3.

1	2	3	5	8	13	20	40	100

Tabla 3. Escala de estimación de esfuerzo.

Una vez establecida la escala a usarse para la estimación de esfuerzo para este proyecto, se tiene una interpretación del puntaje establecido para esta escala en la tabla 4.

Puntaje	Interpretación
1, 2, 3	Se usa para ítems pequeños
5, 8, 13	Se usa para ítem medianos, para algunos equipos de desarrollo 13 sería el límite de puntaje que entraría en un sprint. Comúnmente un ítem de 13 puntos se dividiría en tareas más pequeñas.
20, 40	Se usa para ítem grandes, por ejemplo, historias de usuario a nivel de una característica o tema en específico.
100	Puede representar una característica muy amplia o una épica entera.

Tabla 4. Interpretación de escala de estimación de esfuerzo.

Tomando en cuenta la estimación propuesta anteriormente, se estimaron las historias usuario. La información detallada sobre cada historia se puede encontrar en la tabla 5.

Código	Descripción	Prioridad	Costo	Criterios de aceptación
HF001	Como usuario, deseo ingresar al sistema con un correo electrónico y una contraseña.	Media	40	• El sistema permite ingresar con un correo electrónico y contraseña
HF002	Como usuario, deseo registrarme en el sistema con un correo electrónico y una contraseña.	Media	20	• El sistema permite registrarse con un correo electrónico y una contraseña

Código	Descripción	Prioridad	Costo	Criterios de aceptación
HF003	Como estudiante, deseo visualizar la información de carrera, proceso y subprocesos en los que estoy inscrito. En caso de no tener registro, el sistema debe permitirme escoger una carrera y un proceso para inscribirme.	Alta	40	• Una vez ingresado, el sistema permite visualizar información general de mi proceso. • Si se inicia sesión por primera vez después del registro, el sistema permitirá escoger una carrera y proceso.
HF004	Como administrador, deseo registrar y eliminar usuarios en el sistema.	Alta	40	• El administrador puede agregar y eliminar usuarios en el sistema.
HF005	Como administrador, deseo registrar, actualizar y eliminar carreras en el sistema.	Media	40	• El administrador puede registrar, actualizar y eliminar carreras en el sistema.
HF006	Como administrador, deseo asignar uno o varios procesos a una carrera.	Media	20	• El administrador puede agregar uno o varios procesos a una carrera.
HF007	Como administrador, deseo registrar, actualizar y eliminar procesos en el sistema.	Media	40	• El administrador puede registrar, actualizar y eliminar procesos en el sistema.

Código	Descripción	Prioridad	Costo	Criterios de aceptación
HF008	Como administrador, deseo agregar subprocesos a un proceso en el sistema.	Media	20	• El administrador puede agregar subprocesos a un proceso en el sistema.
HF009	Como administrador, deseo registrar, actualizar y eliminar encargados al sistema.	Media	40	• El administrador puede agregar encargados al sistema.
HN001	Como usuario, deseo utilizar el sistema en un navegador web.	Alta	Ninguna	N/A

Tabla 5. Product Backlog

Adicionalmente, en la guía de desarrollo SCRUM, contamos con una serie de criterios que las historias de usuario deben cumplir, esto se hace para las historias de usuario sean consideradas como *buenas*. Una historia de usuario que depende ampliamente de otras, o que tiene una complejidad muy alta, puede representar un problema en el normal desarrollo de un sprint. Los principios que estas historias cumplen son los principios INVEST que se resumen a continuación:

- **Independiente**
  Las historias altamente dependientes entre sí dificultan su estimación, priorización y planificación dentro de un sprint.
- **Negociable**
  Una historia de usuario puede ser susceptible de cambios, además, esta no representa una estricta descripción de lo que debe hacer, sino, debe capturar y plasmar las principales funcionalidades del negocio que se pretenden cumplir con su desarrollo.

- **Valiosa**

  Una historia de usuario debe aportar valor a un cliente o usuario, es por esto que, en esta se debe representar funcionalidad al producto que el usuario o cliente usará.
- **Estimable**

  Una historia de usuario debe ser estimable por el equipo que la diseñará, desarrollará y probará. Esto es de gran utilidad ya que el equipo de desarrollo deberá dimensionar el esfuerzo de una historia de usuario para poder completar sus tareas dentro de un sprint.
- **Sized (dimensionada)**

  Una historia de usuario debe ser debidamente dimensionada para que el equipo de desarrollo conozca el esfuerzo que debe emplear en completar dicha tarea. Además, se deben tener historias lo más pequeñas posibles, ya que de esta manera se garantiza que se cumplirán dentro del sprint y el equipo de desarrollo no corre el riesgo pasar una historia no finalizada a un sprint posterior.
- **Testable (factible de probar)**

  Este criterio hace principalmente referencia a que una historia debe tener definidos sus criterios de aceptación mediante los cuales el equipo de desarrollo podrá probar la funcionalidad de esta. Los criterios de aceptación deben de escribirse con poco detalle técnico ya que deben evaluar la funcionalidad del producto para el negocio y no otras características no funcionales que pueden ser cubiertas en otro tipo de pruebas. [20]

3.7 Release Planning

Para planificar cada sprint se deben tomar en cuenta varios aspectos importantes, entre ellos tenemos: la velocidad de desarrollo del equipo, cantidad de historias de usuario en el backlog y curva de aprendizaje con las tecnologías a emplearse. Inicialmente se ha establecido que la capacidad de desarrollo del equipo es de 60 puntos estableciendo un sprint de 5 días laborables, es decir, de lunes a viernes.
La suma de la estimación del backlog nos arroja un total de 360 puntos. Por lo tanto, tenemos:

$$\frac{\text{Tamaño del product backlog}}{\text{Capacidad de desarrollo}} = \text{Número de sprints}$$

$$\frac{360}{60} \approx 6, \text{aproximadamente 6 sprints}$$

Después de tener la estimación de número de sprints correspondiente, se debe priorizar el desarrollo de las historias a realizar y considerar si es que existe algún tipo de elemento que represente un bloqueo entre historias. En la tabla 6 se define la distribución correspondiente.

	Release Planning					
	Sprint 1	**Sprint 2**	**Sprint 3**	**Sprint 4**	**Sprint 5**	**Sprint 6**
	HF001	HF009	HF007	HF005	HF004	HF003
	HF002		HF008	HF006		
Total	60	40	60	60	40	40

Tabla 6. Release Planning

Como se puede observar, se han distribuido las historias de tal manera en la que no sobrepasamos la capacidad del equipo de desarrollo, así como las historias que dependen de otras para su desarrollo. De esta manera, se garantiza que se tiene espacio dentro de un sprint para realizar revisiones pertinentes o investigación en caso de que se requiera.

3.8 Sprints

3.8.1 Definición de ready y done

Los elementos que se encuentran en el backlog deben estar listos para ser trasladados a un sprint de tal manera que el equipo de desarrollo pueda comprometerse con confianza y completar estas tareas al finalizar un sprint. [20]

En el desarrollo de este proyecto, una historia de usuario podrá ser considerada como lista para entrar a un sprint cuando se haya definido:

- Estimación de la historia de usuario
- Tareas a desarrollarse
- Criterios de aceptación
- Cumplir con los criterios INVEST.

Adicionalmente, también deben establecerse criterios para que una historia de usuario sea marcada como finalizada. Conceptualmente, la definición de done para una historia es una lista de criterios que se espera cumplir exitosamente. [20]
Para este proyecto, la definición de done para una historia consta en el cumplimiento de los criterios de aceptación con los que la historia fue definida.

3.8.2 Sprint 0

Durante el sprint 0, se revisarán las tecnologías que serán utilizadas para el desarrollo de este proyecto, además, se crearán los repositorios y proyectos base necesarios para poder comenzar con el desarrollo.

- **Objetivo del sprint**: crear de proyectos base, implementar de librerías necesarias para el desarrollo, investigar las tecnologías a usarse en el proyecto.

- **Sprint backlog**: en este sprint no se tiene previsto desarrollar ninguna historia.

- **Daily meeting**: en la ejecución de este sprint se presentaron varios inconvenientes detallados en la tabla 7.

Inconveniente encontrado	Solución
Se deben ejecutar las aplicaciones sobre protocolo HTTPS.	Generar certificados locales usando la herramienta mkcert e instalándolos para poder ejecutar tanto backend como frontend sobre HTTPS.
Existe un error al momento de redirigir la aplicación a la página principal después del inicio de sesión.	Al ejecutarse sobre el dominio “localhost” el proveedor de identidad Auth0 lo considera como un dominio inseguro, arrojando un error de configuración. Se debe cambiar el dominio local sobre el cual se ejecuta el front-end.

Inconveniente encontrado	Solución
No se autentican las peticiones desde el front-end hacia el back-end.	Se debe verificar en la configuración del proyecto de Auth0 que la opción de RBAC se encuentre habilitada para nuestra API.

Tabla 7. Resumen de inconvenientes por Daily meeting, sprint 0.

- **Sprint Review**: se cumplió con el objetivo propuesto al inicio del sprint, no existe un incremento para revisarse por el product owner.

- **Sprint Retrospective**: se deben revisar foros e implementaciones de otros SPAs y APIs que usen Auth0 para comprender más a profundidad su implementación. Se recomienda para el siguiente sprint continuar con la revisión de documentación de Auth0.

3.8.3 Sprint 1

- **Objetivo del sprint**: implementar un inicio de sesión de usuarios usando Auth0.
- **Sprint backlog**: en este sprint se tiene previsto implementar un inicio de sesión que siga todo el flujo de autenticación de nuestro proveedor de identidades Auth0, así como obtener los roles del usuario y validar su identidad.

  Para el desarrollo de este sprint se tienen las historias HF001 y HF002 detalladas en las tablas 8 y 9 respectivamente.

Historia de Usuario	HF001		
Descripción	Como usuario, deseo ingresar al sistema con un correo electrónico y una contraseña.		
Prioridad	Media	**Costo**	40

Criterios de aceptación
• Se debe mostrar una pantalla de inicio que tenga un botón de inicio de sesión. • El botón de inicio de sesión redirigirá el inicio de sesión a Auth0. • Cuando se inicie sesión con un usuario con rol de administrador, se debe mostrar el maquetado del SPA de administrador. • Cuando se inicie sesión con un usuario con rol de estudiante, se debe mostrar el maquetado del SPA de estudiante. • No se debe permitir el acceso a una ruta no autorizada según los roles del usuario al inicio de sesión. • Se debe visualizar la solicitud de token de autenticación después del inicio de sesión. • Pruebas unitarias.
Tareas a realizar

• Maquetar botón de inicio de sesión en el SPA.
• Usando el SDK de Auth0, solicitar un inicio de sesión con redirección.
• Configurar la URL de redirección en el dashboard de Auth0.
• Habilitar el RBAC en la configuración del SPA.
• Crear y configurar un rol de administrador y otro de estudiante en el dashboard de Auth0.
• Configurar un HTTP INTERCEPTOR dentro del SPA para añadir un token de autenticación a las peticiones del SPA.
• Configurar un HTTP INTERCEPTOR dentro del SPA para mostrar mensajes de error en caso de que las peticiones fallen. En el SPA, crear un módulo de administrador y crear componentes para: navegación lateral con menú, incluyendo:
  ○ Módulo de estudiantes.
  ○ Módulo de carreras.
  ○ Módulo de procesos y subprocesos.
  ○ Módulo de encargados.
• En el SPA, crear un módulo de estudiante y crear componentes para: navegación lateral con menú, incluyendo:
  ○ Módulo de procesos y subprocesos.

Tareas a realizar
• Configurar en el módulo de ruteo del SPA las rutas correspondientes para los módulos creados anteriormente.

Tabla 8. Historia de usuario HF001

Historia de Usuario	HF002		
Descripción	Como usuario, deseo registrarme en el sistema con un correo electrónico y una contraseña.		
Prioridad	Media	**Costo**	20
Criterios de aceptación			
• Después de dar clic en el botón de inicio de sesión en el SPA, se debe mostrar el formulario de inicio de sesión y también se debe mostrar la opción de registrarse con un usuario y una contraseña. • Después de haber enviado los datos de registro, se debe redirigir al módulo de estudiante.			
Tareas a realizar			
• Habilitar en el dashboard de Auth0 el registro de usuarios en el formulario de inicio de sesión. (Public sign up) • Configurar una regla de flujo de autenticación para que los usuarios registrados mediante este flujo sean asignados con rol de estudiante automáticamente después del registro.			

Tabla 9. Historia de usuario HF002

- **Daily Scrum**

  El resumen de daily scrum se encuentra debidamente detallado en la tabla 10.

Daily Scrum – Sprint 1			
Día	**¿Qué se hizo ayer que ayudó al equipo a lograr el objetivo del sprint?**	**¿Qué se hará hoy para ayudar al equipo de desarrollo a lograr el objetivo del sprint?**	**¿Se ve algún impedimento que evite que el equipo logre el objetivo del sprint?**
1	N/A	Se crearán todos los módulos requeridos en la HF001 y se maquetará la página de inicio.	N/A
2	Creación de módulos.	Creación de guard de roles.	N/A
3	Creación de guard.	Configuración del SPA en el dashboard de Auth0.	Persiste un error de redirección de Auth0 hacia la página principal del SPA.
4	Configuración de Auth0 en el SPA.	Configuración de las reglas de autenticación en el SPA.	No se ejecuta la regla de autenticación adecuadamente.
5	Configuración de reglas del SPA.	Solucionar el error de la regla de autenticación para asignar roles y habilitar el registro de usuarios en Auth0.	N/A

Tabla 10. Daily Scrum – Sprint 1

- **Codificación**

  En las figuras 6, 7 y 8 se puede observar el resultado del maquetado y creación de formularios especificados previamente en las historias de usuario.

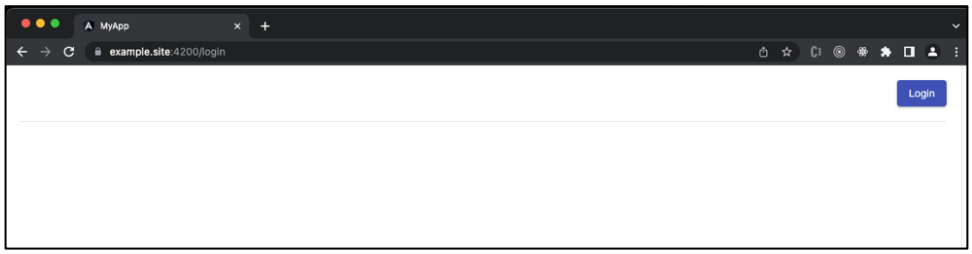

Figura 6. Página de inicio de sesión

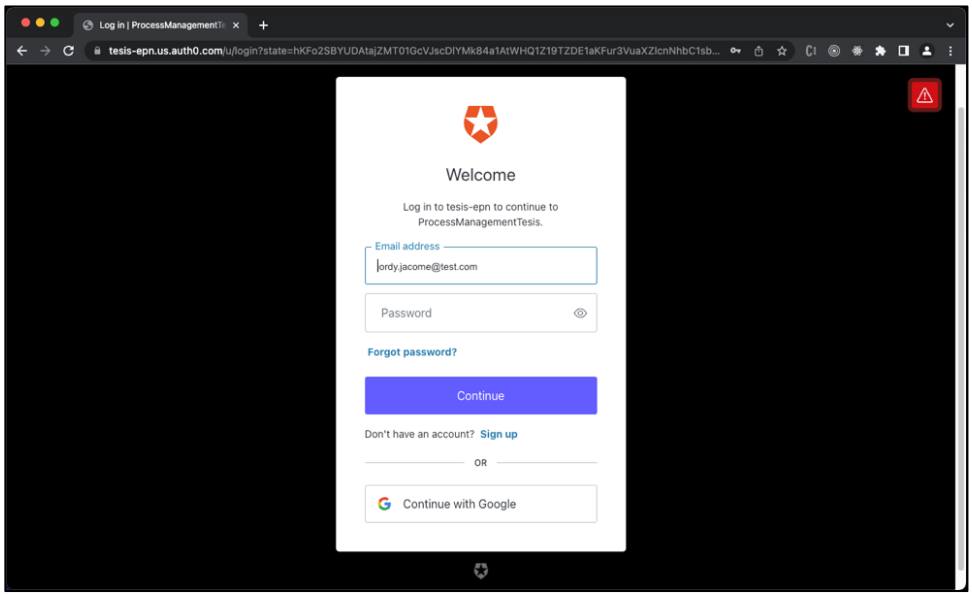

Figura 7. Formulario de inicio de sesión

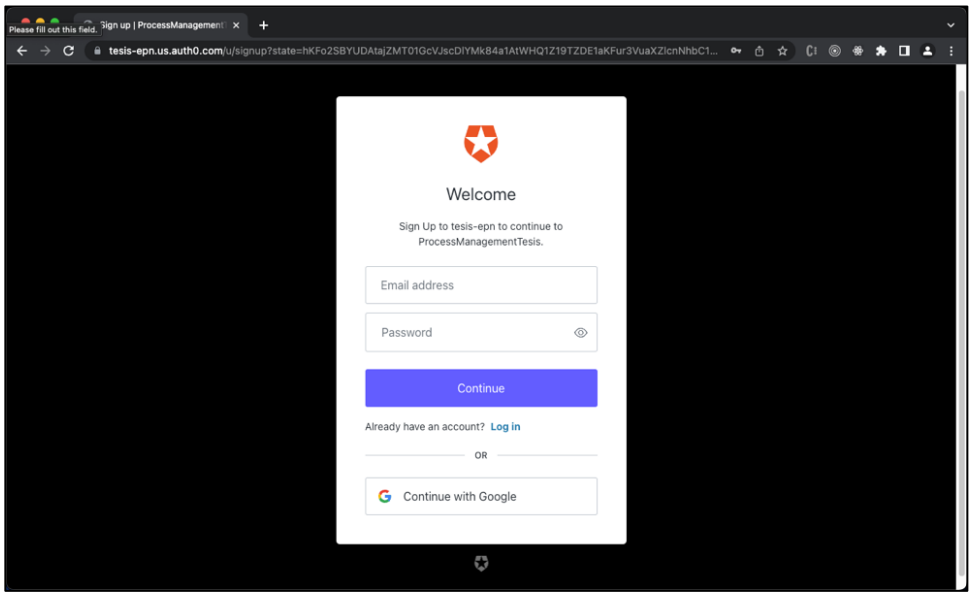

Figura 8. Formulario de registro de usuario

- **Sprint Review**

  Al finalizar el sprint, se validaron los criterios de aceptación de las historias que se desarrollaron y se definieron algunas observaciones que deberán resolverse posterior investigación. Se detallan las observaciones encontradas en la tabla 11.

Historia	Observación
HF002	La regla de autenticación debe enviar un correo para validar el correo electrónico de la cuenta. Se debe validar por qué el servidor de correo rechaza los envíos.
HF001	Ninguna.

Tabla 11. Resultados – Sprint Review 1

- **Sprint Retrospective**

  Durante el desarrollo de este sprint, se puede evidenciar que no se mantuvo el ritmo esperado, esto se debe a que existieron inconvenientes con las pruebas que validaban la regla de asignación de roles en el registro de un nuevo usuario. Se tiene provisto investigar para poder darle solución al inconveniente encontrado en la review de este sprint. Sin embargo, este inconveniente no representó un impedimento para la finalización de las

historias planificadas para el desarrollo. El resumen de avance de este sprint se encuentra detallado en la figura 9.

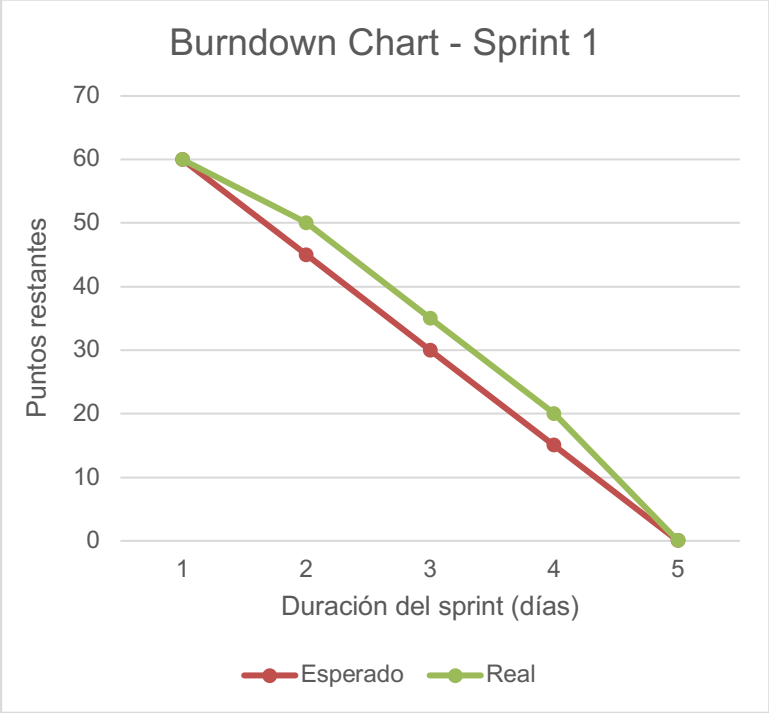

Figura 9. Burndown Chart – Sprint 1

3.8.4 Sprint 2

- **Objetivo del sprint:** implementar un CRUD para el módulo de encargados del sistema principal.

- **Sprint backlog**

  Para el desarrollo de este sprint se tiene la historia HF009 detallada en la tabla 12.

Historia de Usuario	HF009		
Descripción	Como administrador, deseo registrar, actualizar y eliminar encargados al sistema.		
Prioridad	Media	**Costo**	40
Criterios de aceptación			
• El administrador puede ver, agregar, editar y eliminar encargados al sistema.			

Criterios de aceptación
• Al momento de agregar o editar un encargado en el sistema, se deben validar los datos de entrada en el formulario: nombre, teléfono, correo electrónico y dirección, luego actualizar y mostrar el nuevo registro.
Tareas a realizar

• Crear el documento para almacenar los datos de encargados en el proyecto del back-end tomando en cuenta los siguientes campos: *name, phone, email, address.* Considerar a todos los campos como requeridos.
• Crear los puntos de entrada en el API que cumplan las siguientes características:
  o Lectura de todos los usuarios, GET
  o Lectura de un usuario determinado con ID, GET
  o Creación de un usuario con los campos especificados, POST
  o Editar un usuario con ID y con los campos especificados, PATCH
  o Eliminar un usuario con ID, DELETE
• Validar los datos para los puntos de entrada detallados anteriormente.
• Crear un GUARD que valide el token JWT provisto por el API de Auth0. (Autenticación)
• Crear un GUARD que valide los permisos del usuario en el token JWT para la lectura, escritura, edición y eliminación de datos en los puntos de entrada detallados anteriormente. (Autorización)
• Validar los permisos usando el siguiente esquema (endpoint, permiso)
  o GET (:/id), “read:accountant”
  o GET (), “read:accountant”
  o POST (), “create:accountant”
  o PATCH (:/id), “update:accountant”
  o DELETE (:id), “delete:accountant”
• Crear un proyecto de base de datos en MongoDB Cloud y en el proyecto configurar las credenciales de acceso al clúster de

MongoDB, librerías e inyección de dependencia para poder hacer uso de la base de datos.

- En el back-end, dentro del servicio de encargados, crear las funciones y lógica necesarias para crear, leer, editar y eliminar encargados según corresponda. Tomar en cuenta la inyección de dependencias del servicio de MongoDB.
- En el front-end, usar el componente de encargados y maquetar la vista según corresponda (revisar mock-ups).
- Crear un módulo compartido en la raíz del proyecto para crear las interfaces necesarias que van a mapear los datos que se reciban del back-end en cada request.
- Usar el servicio de API y crear las funciones necesarias para consumir los métodos del back-end, parametrizar la URL de raíz y colocarla en un archivo para ser leído como variable de ambiente.
- Para la creación de encargados, maquetar el formulario de creación y usar ReactiveForms de Angular. Validar que todos los campos sean requeridos y además el formato y tipo de dato según corresponda.
- Después de una operación de inserción o actualización, actualizar el DataSource para mostrar los cambios realizados por el usuario.
- Escribir pruebas unitarias tanto para el código generado en el front-end y en el back-end.

Tabla 12. Historia de usuario HF009

- **Daily Scrum**

  El resumen de daily scrum se encuentra debidamente detallado en la tabla 13.

Daily Scrum – Sprint 2			
Día	**¿Qué se hizo ayer que ayudó al equipo a lograr el objetivo del sprint?**	**¿Qué se hará hoy para ayudar al equipo de desarrollo a lograr el objetivo del sprint?**	**¿Se ve algún impedimento que evite que el equipo logre el objetivo del sprint?**
1	N/A	Se creará el documento de MongoDB y se configurará el proyecto para conectar la base de datos.	Se está rechazando la conexión hacia la base de datos. Se muestra un mensaje de DNS no encontrado.
2	Se creó el documento de MongoDB para almacenar los datos.	Se crearán las funciones y lógica necesaria para el CRUD de encargados, así como los puntos de entrada para las funciones.	N/A
3	Se creó la lógica para el CRUD y sus respectivos puntos de entrada.	Creación de GUARDS para el proyecto del back-end y pruebas unitarias. Maquetado de la interfaz principal en el front-end.	No se validan correctamente los permisos del usuario que vienen en el token JWT.
4	Creación y configuración de GUARDS y maquetado del front-end.	Implementación de lógica y funciones para consumir los datos del back-end.	N/A

Daily Scrum – Sprint 2			
Día	**¿Qué se hizo ayer que ayudó al equipo a lograr el objetivo del sprint?**	**¿Qué se hará hoy para ayudar al equipo de desarrollo a lograr el objetivo del sprint?**	**¿Se ve algún impedimento que evite que el equipo logre el objetivo del sprint?**
5	Lógica para consumir los datos del back-end.	Mostrar datos del backend y completar las acciones de actualización y eliminación en el SPA.	N/A

Tabla 13. Daily Scrum – Sprint 2

- **Codificación**

  En las figuras 10 y 11 se puede observar el resultado del maquetado y creación de formularios especificados previamente en las historias de usuario.

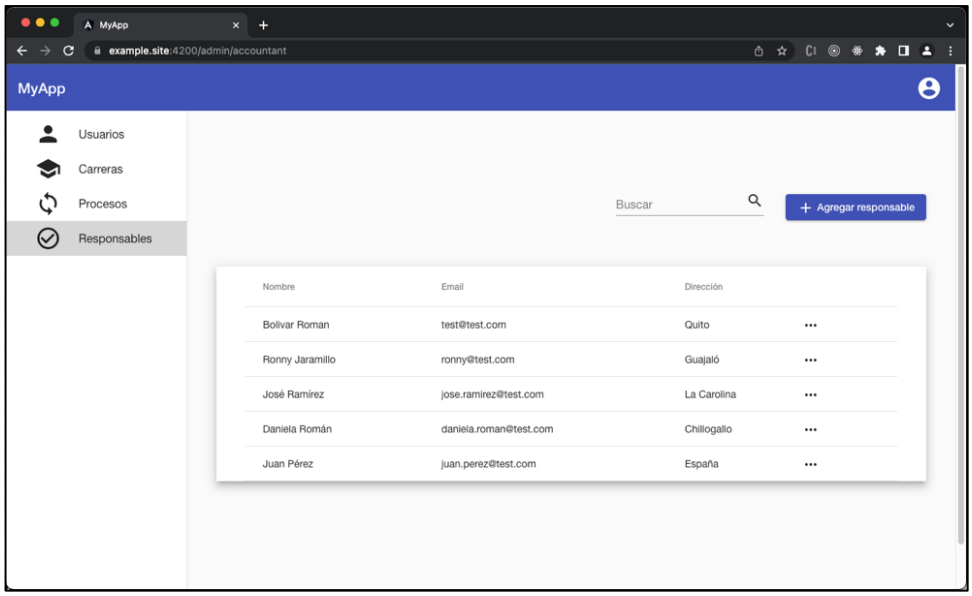

Figura 10. Maquetado principal del módulo de encargados

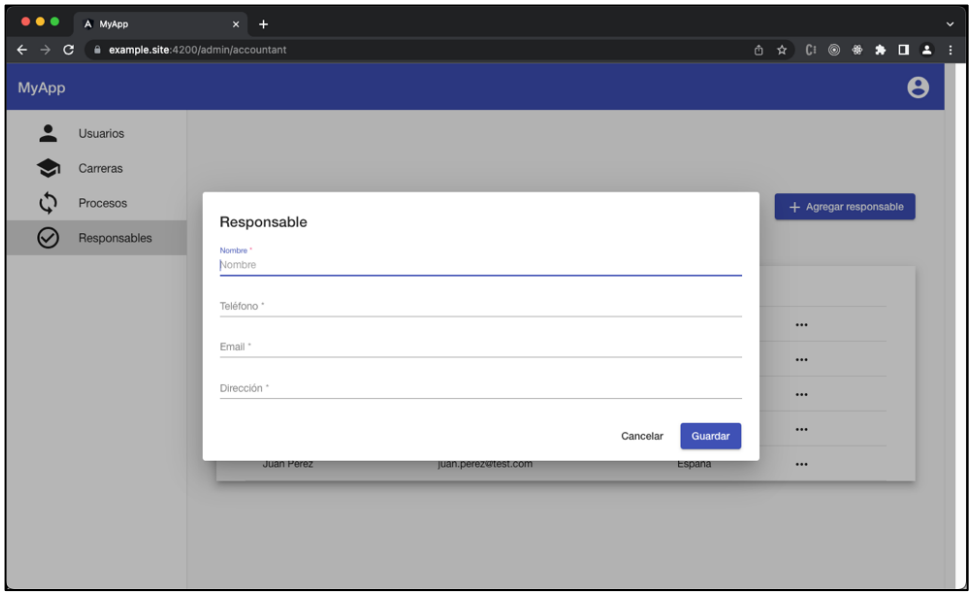

Figura 11. Maquetado de formulario de creación/edición de encargados

- **Sprint Review**

  Al finalizar el sprint, se validaron los criterios de aceptación de la única historia que se desarrolló. Los resultados obtenidos se encuentran en la tabla 14.

Historia	Observación
HF009	Se cumplieron los criterios de aceptación planteados para esta historia. Sin embargo, si es que se prueba el sistema con otra conexión a internet podrían surgir errores de acceso al clúster de MongoDB. Tomar en cuenta esta consideración para pruebas con el product owner.

Tabla 14. Resultados – Sprint Review 2

- **Sprint Retrospective**

  Al inicio de este sprint se presentaron algunos inconvenientes que detuvieron el progreso del sprint como se esperaba.

  El inconveniente principal era el de conexión hacia el clúster de base de datos de MongoDB. Estos problemas surgieron debido a el proveedor de

internet actual que bloqueaba la salida de conexión. Esto se solventó usando un servicio de VPN que no bloquee la conexión hacia el clúster. El resumen de avance de este sprint se encuentra detallado en la figura 12.

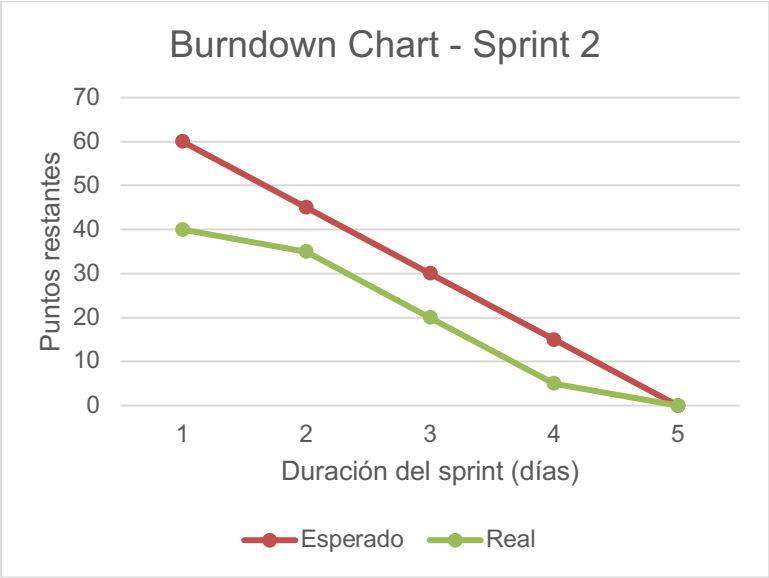

Figura 12. Burndown Chart – Sprint 2

3.8.5 Sprint 3

- **Objetivo del sprint:** implementar un CRUD para el módulo de procesos y subprocesos del sistema principal.

- **Sprint backlog**

  Para el desarrollo de este sprint se tienen las historias HF007 y HF008 detalladas en las tablas 15 y 16 respectivamente.

Historia de Usuario	HF007		
Descripción	Como administrador, deseo registrar, actualizar y eliminar procesos en el sistema.		
Prioridad	Media	**Costo**	40
Criterios de aceptación			

- El administrador puede ver, agregar, editar y eliminar procesos al sistema.
- Al momento de agregar o editar un proceso en el sistema, se deben validar los datos de entrada en el formulario: nombre y versión, luego actualizar y mostrar el nuevo registro.

Tareas a realizar

- Crear el documento para almacenar los datos de procesos en el proyecto del back-end tomando en cuenta los siguientes campos: *name, version.* Considerar a todos los campos como requeridos.
- Crear los puntos de entrada en el API que cumplan las siguientes características:
    - Lectura de todos los procesos, GET
    - Lectura de un proceso determinado con ID, GET
    - Creación de un proceso con los campos especificados, POST
    - Editar un proceso con ID y con los campos especificados, PATCH
    - Eliminar un proceso con ID, DELETE
- Validar los datos para los puntos de entrada detallados anteriormente.
- Crear el documento para almacenar los datos de procesos en el proyecto del back-end tomando en cuenta los siguientes campos: *name, version.* Considerar a todos los campos como requeridos.
- Crear los puntos de entrada en el API que cumplan las siguientes características:
    - Lectura de todos los procesos, GET
    - Lectura de un proceso determinado con ID, GET
    - Creación de un proceso con los campos especificados, POST
    - Editar un proceso con ID y con los campos especificados, PATCH
    - Eliminar un proceso con ID, DELETE
- Validar los datos para los puntos de entrada detallados anteriormente.

• Validar los permisos usando el siguiente esquema (endpoint, permiso)
  o GET (), “read:process”
  o GET (:/id), “read:process”
  o POST (), “create:process”
  o PATCH (:/id), “update:process”
  o DELETE (:id), “delete:process”
• En el back-end, dentro del servicio de procesos, crear las funciones y lógica necesarias para crear, leer, editar y eliminar procesos según corresponda.
• En el front-end, usar el componente de procesos y maquetar la vista según corresponda (revisar mock-ups).
• Dentro del módulo compartido que tiene las interfaces, añadir según sea necesario para mapear los datos de procesos que se reciben del back-end en cada request.
• Usar el servicio de API y crear las funciones necesarias para consumir los métodos del back-end usando las variables de ambiente creadas anteriormente.
• Para la creación de procesos, maquetar el formulario de creación y usar ReactiveForms de Angular. Validar que todos los campos sean requeridos y además el formato y tipo de dato según corresponda.
• Después de una operación de inserción o actualización, actualizar el DataSource para mostrar los cambios realizados por el usuario.
• Escribir pruebas unitarias tanto para el código generado en el front-end y en el back-end.

Tabla 15. Historia de usuario HF007

Historia de Usuario	HF008		
Descripción	Como administrador, deseo agregar subprocesos a un proceso en el sistema.		
Prioridad	Media	**Costo**	20
Criterios de aceptación			
• El administrador puede ver, agregar, editar y eliminar subprocesos a un proceso en el sistema.			

- Después de agregar un subproceso, se deben actualizar los datos para visualizar los nuevos cambios.
- Al momento de agregar un subproceso, se deben mostrar los encargados disponibles.
- Se deben validar como requeridos los datos de entrada en el formulario para la creación de un nuevo subproceso.

Tareas a realizar

- Crear un documento para almacenar los datos de subprocesos en el proyecto del back-end tomando en cuenta los siguientes campos: *name, description, accountantId, processId.* Considerar a todos los campos como requeridos. Además, tomar en cuenta que los campos de ID deben hacer referencia al identificador de un registro en otra colección. Para más información revisar la documentación de MongoDB.
- Crear los puntos de entrada en el API que cumplan las siguientes características:
    - Lectura de un subproceso determinado con ID, GET.
    - Creación de un subproceso con los campos especificados y con un ID de proceso, POST.
    - Editar un subproceso con ID y con los campos especificados, PATCH.
- Eliminar un proceso con ID, DELETE.
- Validar los datos para los puntos de entrada detallados anteriormente.
- Validar los permisos usando el siguiente esquema (endpoint, permiso)
    - GET (:/id), “read:subprocess”
    - POST (:/id), “create:subprocess”
    - PATCH (:/id), “update:subprocess”
    - DELETE (:id), “delete:subprocess”
- En el back-end, dentro del servicio de subprocesos, crear las funciones y lógica necesarias para crear, leer, editar y eliminar procesos según corresponda.

• En el front-end, usar el componente de procesos y maquetar la vista según corresponda (revisar mock-ups). Tomar en cuenta que el módulo de subprocesos hace parte del módulo de procesos. • Dentro del módulo compartido que tiene las interfaces, añadir según sea necesario para mapear los datos de procesos que se reciben del back-end en cada request. • Usar el servicio de API y crear las funciones necesarias para consumir los métodos del back-end usando las variables de ambiente creadas anteriormente. • Para la creación de subprocesos, maquetar el formulario de creación y usar ReactiveForms de Angular. Validar que todos los campos sean requeridos y además el formato y tipo de dato según corresponda. • Después de una operación de inserción o actualización, actualizar el DataSource para mostrar los cambios realizados por el usuario.

Tabla 16. Historia de usuario HF008

- **Daily Scrum**

  El resumen de daily scrum se encuentra debidamente detallado en la tabla 17.

Daily Scrum – Sprint 3			
Día	**¿Qué se hizo ayer que ayudó al equipo a lograr el objetivo del sprint?**	**¿Qué se hará hoy para ayudar al equipo de desarrollo a lograr el objetivo del sprint?**	**¿Se ve algún impedimento que evite que el equipo logre el objetivo del sprint?**
1	N/A	Se crearán los documentos, lógica y puntos de entrada para el módulo de procesos.	N/A

Día	¿Qué se hizo ayer que ayudó al equipo a lograr el objetivo del sprint?	¿Qué se hará hoy para ayudar al equipo de desarrollo a lograr el objetivo del sprint?	¿Se ve algún impedimento que evite que el equipo logre el objetivo del sprint?
2	Se creó la lógica del módulo de procesos y sus puntos de entrada en el API.	Se crearán los documentos, lógica y puntos de entrada para el módulo de subprocesos.	Al momento de guardar el objeto de subprocesos no se está haciendo referencia al ObjectId del proceso.
3	Se creó la lógica del módulo de subprocesos y sus puntos de entrada en el API.	Maquetado de la interfaz principal del módulo de procesos en el front-end.	N/A
4	Maquetado de la interfaz del módulo de procesos.	Maquetado de la interfaz principal del módulo de subprocesos en el front-end.	N/A
5	Maquetado de la interfaz del módulo de subprocesos.	Validación al momento de guardar objetos con referencia un ObjectId y pruebas unitarias.	N/A

Tabla 17. Daily Scrum – Sprint 3

- **Codificación**

  En las figuras 13, 14, 15 y 16 se puede observar el resultado del maquetado y creación de formularios especificados previamente en las historias de usuario.

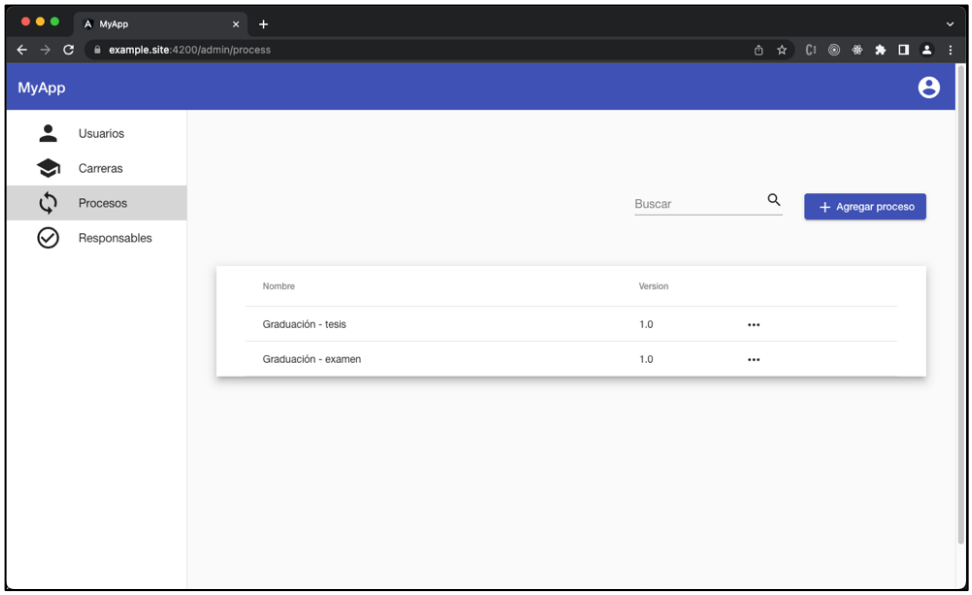

Figura 13. Maquetado principal del módulo de procesos

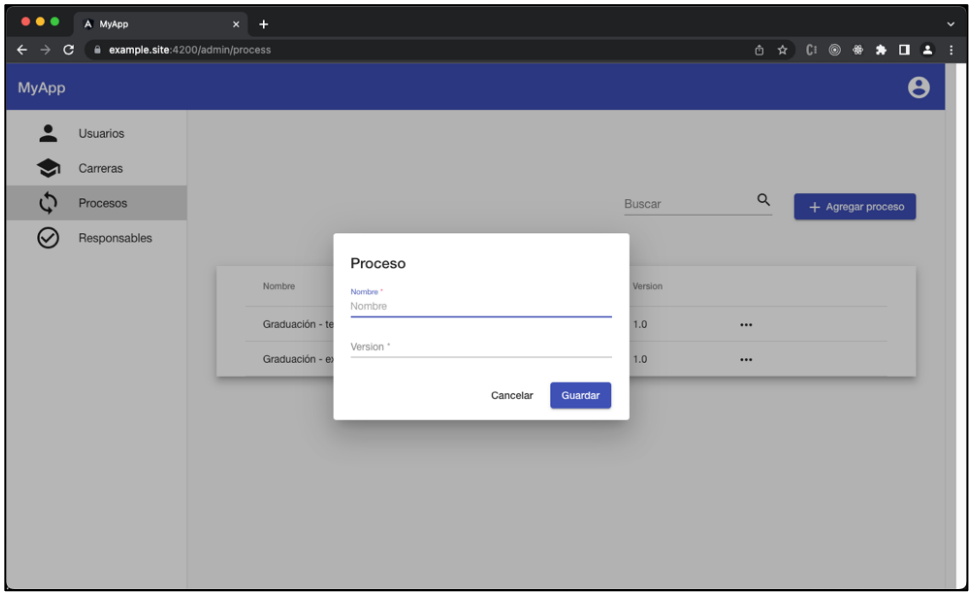

Figura 14. Maquetado de formulario de creación/edición de procesos

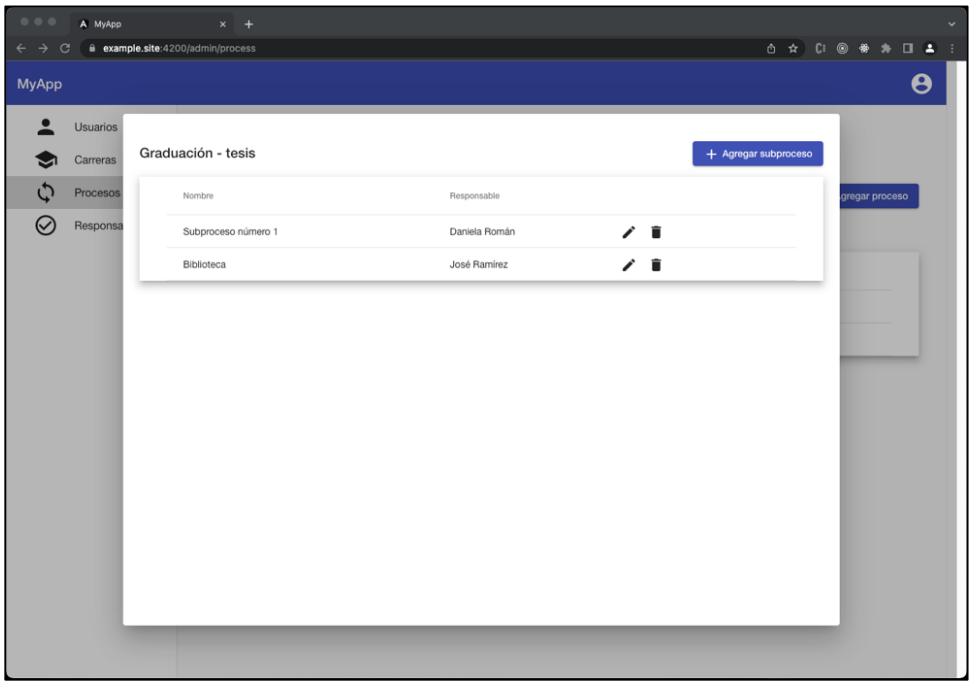

Figura 15. Maquetado principal del módulo de subprocesos

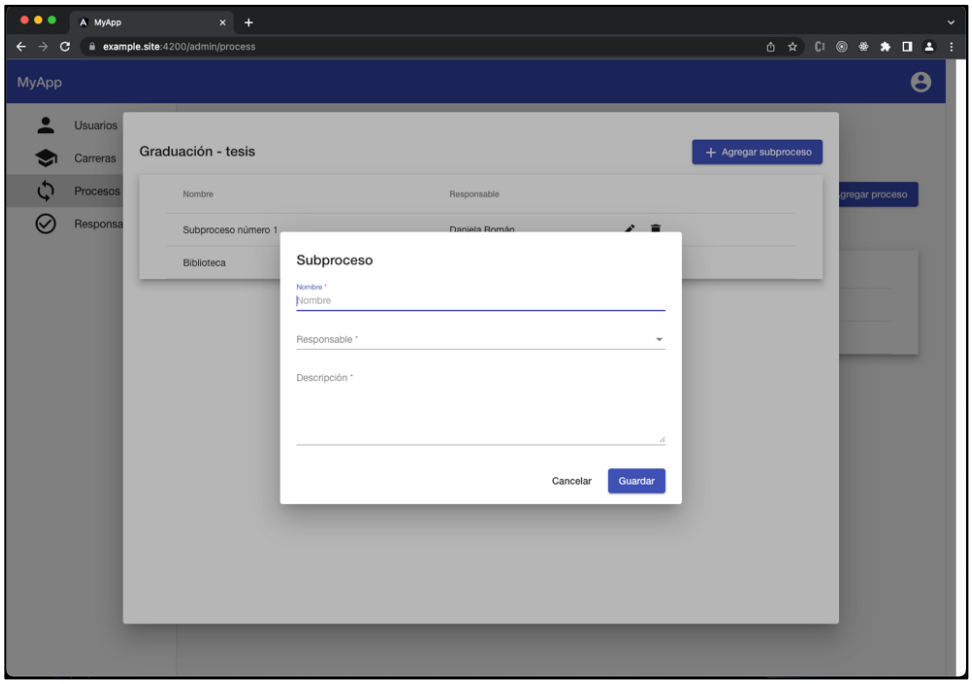

Figura 16. Maquetado de formulario de creación/edición de subprocesos

- **Sprint Review**

  Al finalizar el sprint, se validaron los criterios de aceptación de las historias desarrolladas. Los resultados obtenidos se detallan en la tabla 18.

Historia	Observación
HF007	Se creó correctamente el módulo de procesos, no se presentaron inconvenientes mayores ya que se recicló la mayoría de la lógica implementada en el módulo de encargados desarrollado el anterior sprint.
HF008	Se desarrolló el módulo de subprocesos en el back-end de forma separada del módulo de procesos para concordar con el diseño del API propuesto. En el front-end se integraron los dos módulos en una sola ventana debido a la relación de uno con el otro.

Tabla 18. Resultados – Sprint Review 3

- **Sprint Retrospective**

  Al principio de este sprint no se presentaron inconvenientes en el desarrollo. Cuando se implementó el módulo de subprocesos surgieron algunos inconvenientes, incluyendo la referencia al ObjectId del proceso que se encontraba en otra colección. Se solventaron estos errores vinculando el documento de procesos con el de subprocesos. Se encontraron varias soluciones al problema en diferentes blogs de internet. El resumen de avance de este sprint se encuentra detallado en la figura 17.

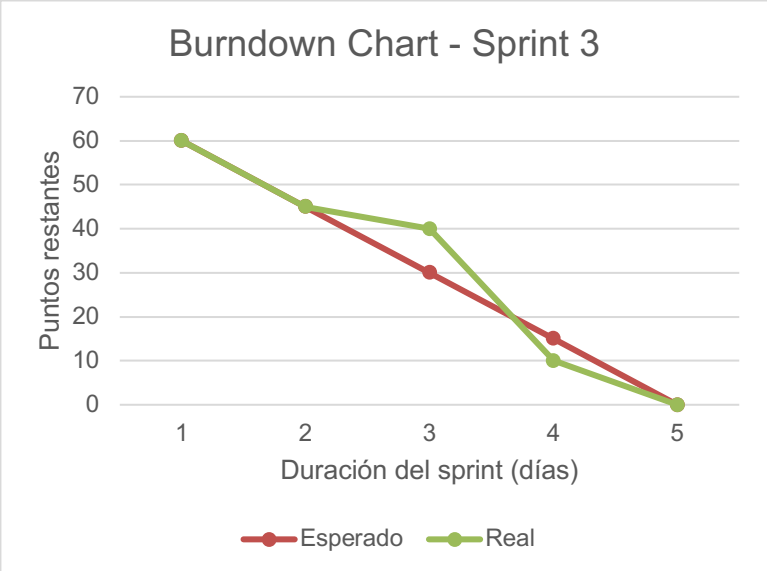

Figura 17. Burndown Chart – Sprint 3

3.8.6 Sprint 4

- **Objetivo del sprint:** implementar un CRUD para el módulo de carreras y del sistema principal y asignación de procesos a una determinada carrera.

- **Sprint backlog**

  Para el desarrollo de este sprint se tienen las historias HF005 y HF006 detalladas en las tablas 19 y 20 respectivamente.

Historia de Usuario	HF005		
Descripción	Como administrador, deseo registrar, actualizar y eliminar carreras en el sistema.		
Prioridad	Media	**Costo**	40
Criterios de aceptación			
• El administrador puede registrar, actualizar y eliminar carreras en el sistema. • Después de agregar una carrera, se deben actualizar los datos para visualizar los nuevos cambios. • Se deben validar como requeridos los datos de entrada en el formulario para la creación de una nueva carrera.			
Tareas a realizar			

• Crear un documento para almacenar los datos de carreras en el proyecto del back-end tomando en cuenta los siguientes campos: *name, processes.* Considerar a todos los campos como requeridos. Además, tomar en cuenta que el campo *processes* debe hacer referencia al identificador de un registro de un proceso en otra colección. Para más información revisar la documentación de MongoDB.
• Crear los puntos de entrada en el API que cumplan las siguientes características:
  o Lectura de todas las carreras, GET.
  o Lectura de una carrera determinada con ID, GET.
  o Creación de una carrera con los campos especificados, POST.			

○ Editar una carrera con ID y con los campos especificados, PATCH. ○ Eliminar una carrera con ID, DELETE. • Validar los datos para los puntos de entrada detallados anteriormente. • Validar los permisos usando el siguiente esquema (endpoint, permiso) ○ GET (), “read:degree” ○ GET (:/id), “read:degree” ○ POST (:/id), “create:degree” ○ PATCH (:/id), “update:degree” ○ DELETE (:id), “delete:degree” • En el back-end, dentro del servicio de carreras, crear las funciones y lógica necesarias para crear, leer, editar y eliminar procesos según corresponda. • En el front-end, usar el componente de carreras y maquetar la vista según corresponda (revisar mock-ups). • Dentro del módulo compartido que tiene las interfaces, añadir según sea necesario para mapear los datos de carreras que se reciben del back-end en cada request. • Usar el servicio de API y crear las funciones necesarias para consumir los métodos del back-end usando las variables de ambiente creadas anteriormente. • Para la creación de carreras, maquetar el formulario de creación y usar ReactiveForms de Angular. Validar que todos los campos sean requeridos y además el formato y tipo de dato según corresponda. • Después de una operación de inserción o actualización, actualizar el DataSource para mostrar los cambios realizados por el usuario.

Tabla 19. Historia de usuario HF005

Historia de Usuario	HF006		
Descripción	Como administrador, deseo asignar uno o varios procesos a una carrera.		
Prioridad	Media	**Costo**	20
Criterios de aceptación			

• El administrador puede agregar uno o varios procesos a una carrera. • Después de agregar un proceso, se deben actualizar los datos para visualizar los nuevos cambios. • Al momento de agregar un proceso a una carrera, se deben mostrar los procesos disponibles. • Se deben validar como requeridos los datos de entrada en el formulario para la creación de una nueva carrera.
Tareas a realizar
• Revisar la implementación del documento para las carreras y agregar la referencia de ObjectId del proceso correspondiente. • Validar los datos de entrada del formulario para la creación de carreras y procesos. • En el front-end, modificar el formulario de creación para desplegar las carreras disponibles y habilitar la selección múltiple. (Una carrera puede tener varios procesos).

Tabla 20. Historia de usuario HF006

- **Daily Scrum**

  El resumen de daily scrum se encuentra debidamente detallado en la tabla 21.

Daily Scrum – Sprint 4			
Día	**¿Qué se hizo ayer que ayudó al equipo a lograr el objetivo del sprint?**	**¿Qué se hará hoy para ayudar al equipo de desarrollo a lograr el objetivo del sprint?**	**¿Se ve algún impedimento que evite que el equipo logre el objetivo del sprint?**
1	N/A	Se crearán los documentos, lógica y puntos de entrada para el módulo de carreras.	N/A

Día	¿Qué se hizo ayer que ayudó al equipo a lograr el objetivo del sprint?	¿Qué se hará hoy para ayudar al equipo de desarrollo a lograr el objetivo del sprint?	¿Se ve algún impedimento que evite que el equipo logre el objetivo del sprint?
2	Se creó la lógica del módulo de carreras y sus puntos de entrada en el API.	Se crearán los documentos, lógica y puntos de entrada para el módulo de carreras.	Al momento de guardar el objeto carrera no se está haciendo referencia al ObjectId del proceso.
3	Se crearán los documentos, lógica y puntos de entrada para el módulo de carreras.	Maquetado de la interfaz principal del módulo de carreras en el front-end.	N/A
4	Maquetado de la interfaz del módulo de carreras.	Maquetado de la interfaz principal del módulo de carreras en el front-end.	N/A
5	Maquetado de la interfaz principal del módulo de carreras en el front-end.	Validación de guardado de objetos en MongoDB con sus respectivas refererencias y pruebas unitarias.	N/A

Tabla 21. Daily Scrum – Sprint 4

- **Codificación**

  En las figuras 18 y 19 se puede observar el resultado del maquetado y creación de formularios especificados previamente en las historias de usuario.

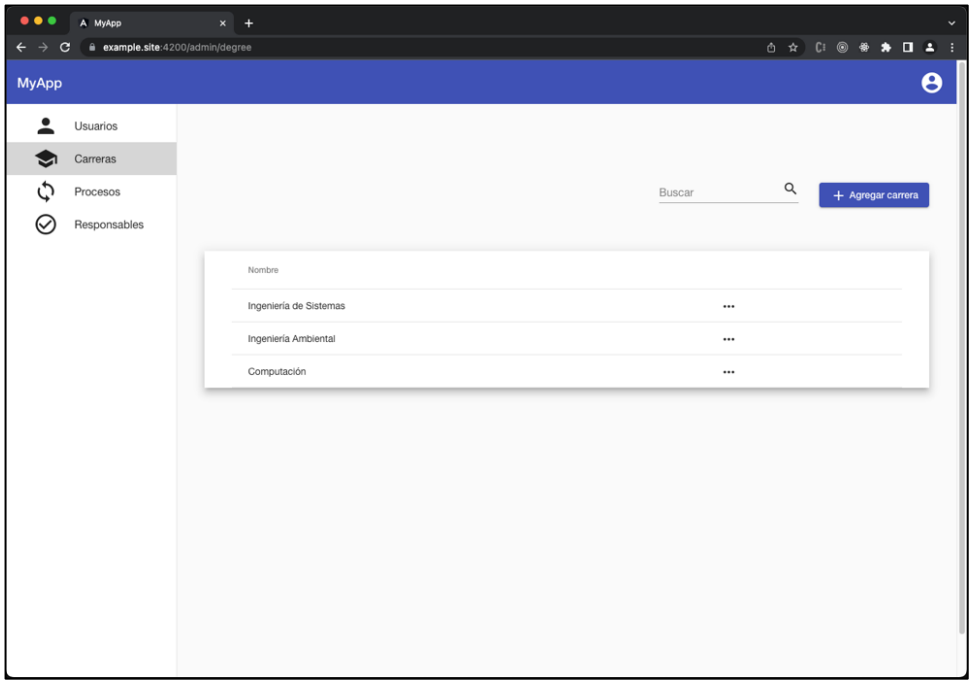

Figura 18. Maquetado principal del módulo de carreras

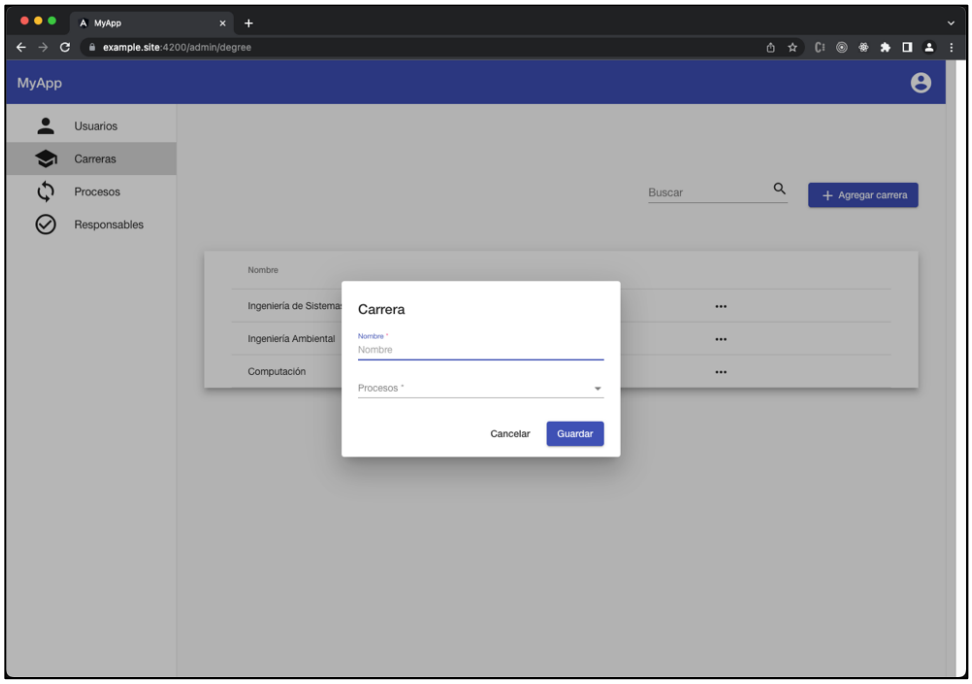

Figura 19. Maquetado de formulario de creación/edición de carreras

- **Sprint Review**

  Al finalizar el sprint, se validaron los criterios de aceptación de las historias desarrolladas. Los resultados obtenidos se encuentran a continuación en la tabla 22.

Historia	Observación
HF005	Se creó correctamente el módulo de carreras, no se presentaron inconvenientes mayores ya que se recicló la mayoría de la lógica implementada en el módulo de procesos desarrollado el anterior sprint.
HF006	Se desarrolló esta historia tomando en cuenta que se debía almacenar un array de procesos por carrera, de esta manera, se tenía que hacer referencia al ObjectId de cada proceso dentro de un array. Se presentaron inconvenientes al principio principalmente por intentar guardar un array de referencias.

Tabla 22. Resultados – Sprint Review 4

- **Sprint Retrospective**

  Durante el desarrollo de este sprint no se presentaron mayores inconvenientes. Se tuvo un problema al intentar guardar un array de ObjectId que hacían referencia a los IDs de los diferentes procesos que se han registrado. Sin embargo, se encontró una solución para este problema rápidamente ya que es un problema común en MongoDB. El resumen de avance de este sprint se encuentra detallado en la figura 20.

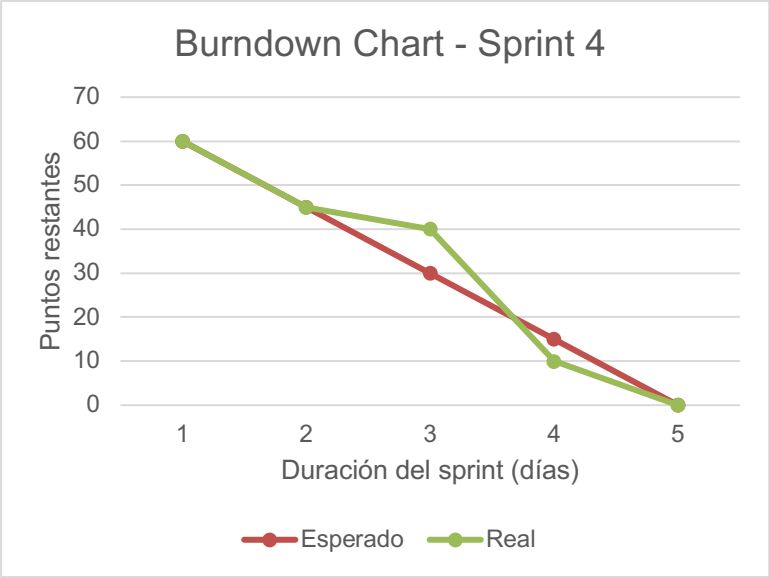

Figura 20. Burndown Chart - Sprint 4

3.8.7 Sprint 5

- **Objetivo del sprint**: integrar los servicios de administración de usuarios de Auth0 con la aplicación creada.

- **Sprint backlog**

  Para el desarrollo de este sprint se tiene la historia HF004 detallada en la tabla 23.

Historia de Usuario	HF004		
Descripción	Como administrador, deseo registrar y eliminar usuarios en el sistema.		
Prioridad	Alta	**Costo**	40
Criterios de aceptación			
• El administrador puede registrar y eliminar usuarios en el sistema. • Después de agregar un usuario, se deben actualizar los datos para visualizar los nuevos cambios. • Se deben validar como requeridos los datos de entrada en el formulario para la creación de una nueva carrera.			

Tareas a realizar

• Crear un documento para almacenar los datos de administración en el proyecto del back-end tomando en cuenta los siguientes campos: *name, email.* Considerar a todos los campos como requeridos.
• Crear los puntos de entrada en el API que cumplan las siguientes características:
  ○ Lectura de todos los usuarios, GET.
  ○ Lectura de un usuario determinado con ID, GET.
  ○ Creación de un usuario con los campos especificados, POST.
  ○ Asignar una carrera a un usuario determinado con ID, POST.
  ○ Eliminar un usuario con ID, DELETE.
• Validar los datos para los puntos de entrada detallados anteriormente.
• Validar los permisos usando el siguiente esquema (endpoint, permiso)
  ○ GET (), “read:management”
  ○ GET (:/id), “read:management”
  ○ POST (), “create:management”
  ○ POST (:/id), “assign:management”
  ○ DELETE (:id), “delete:management”
• En el back-end, dentro del servicio de administración, crear las funciones y lógica necesarias para crear, leer, asignar carrera y eliminar usuarios según corresponda.
• En el front-end, usar el componente de usuarios y maquetar la vista según corresponda (revisar mock-ups).
• Dentro del módulo compartido que tiene las interfaces, añadir según sea necesario para mapear los datos de usuarios que se reciben del back-end en cada request.
• Para la creación de usuarios, maquetar el formulario de creación y usar ReactiveForms de Angular. Validar que todos los campos sean requeridos y además el formato y tipo de dato según corresponda.

• En el back-end, revisar la conexión y comunicación con el API de administración de Auth0. Se debe requerir un token de autenticación extra debido a que se harán peticiones a otra API. Revisar el diagrama de arquitectura de aplicación y la documentación del API de Auth0. • En el front-end, usar el componente de usuarios y maquetar la vista según corresponda (revisar mock-ups). • Usar el servicio de API y crear las funciones necesarias para consumir los métodos del back-end usando las variables de ambiente creadas anteriormente. • Después de una operación de inserción o actualización, actualizar el DataSource para mostrar los cambios realizados por el usuario.

Tabla 23. Historia de usuario HF004

- **Daily Scrum**

  El resumen de daily scrum se encuentra debidamente detallado en la tabla 24.

Daily Scrum – Sprint 5			
Día	**¿Qué se hizo ayer que ayudó al equipo a lograr el objetivo del sprint?**	**¿Qué se hará hoy para ayudar al equipo de desarrollo a lograr el objetivo del sprint?**	**¿Se ve algún impedimento que evite que el equipo logre el objetivo del sprint?**
1	N/A	Se crearán los documentos, lógica y puntos de entrada para el módulo de administración.	Se está rechazando el request de token al API de Auth0.

Día	¿Qué se hizo ayer que ayudó al equipo a lograr el objetivo del sprint?	¿Qué se hará hoy para ayudar al equipo de desarrollo a lograr el objetivo del sprint?	¿Se ve algún impedimento que evite que el equipo logre el objetivo del sprint?
2	Se creó la lógica del módulo de administración y sus puntos de entrada en el API.	Se investigará sobre cómo solicitar el token de acceso para el API de Auth0.	Se debe añadir el token en todos los request de salida, se prevé añadirlos manualmente o añadir un interceptor para añadir los tokens.
3	Se investigó sobre cómo solicitar el token para el API Auth0 y cómo hacer su implementación	Refactorizar la lógica del CRUD de usuarios para añadir el token al request hacia Auth0.	Actualmente existe un problema en el desarrollo del flujo para añadir usuarios debido a que se tiene previsto que llegue un correo con un enlace de invitación y cambio de contraseña.
4	Se refactorizó la lógica del servicio de administración y se probaron las diferentes funciones.	Maquetado de la interfaz principal del módulo de administración en el front-end.	N/A

Día	¿Qué se hizo ayer que ayudó al equipo a lograr el objetivo del sprint?	¿Qué se hará hoy para ayudar al equipo de desarrollo a lograr el objetivo del sprint?	¿Se ve algún impedimento que evite que el equipo logre el objetivo del sprint?
5	Maquetado de la interfaz principal del módulo de administración en el front-end.	Validación de guardado de objetos en MongoDB con sus respectivas refererencias y pruebas unitarias.	N/A

Tabla 24. Daily Scrum – Sprint 5

- **Codificación**

  En las figuras 21 y 22 se puede observar el resultado del maquetado y creación de formularios especificados previamente en las historias de usuario.

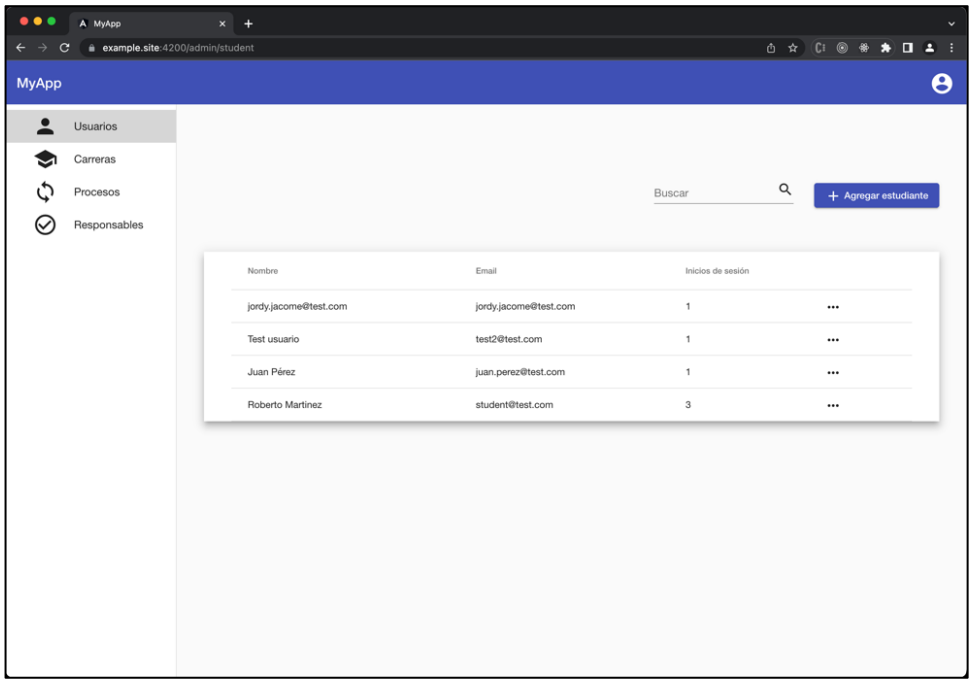

Figura 21. Maquetado principal del módulo de administración

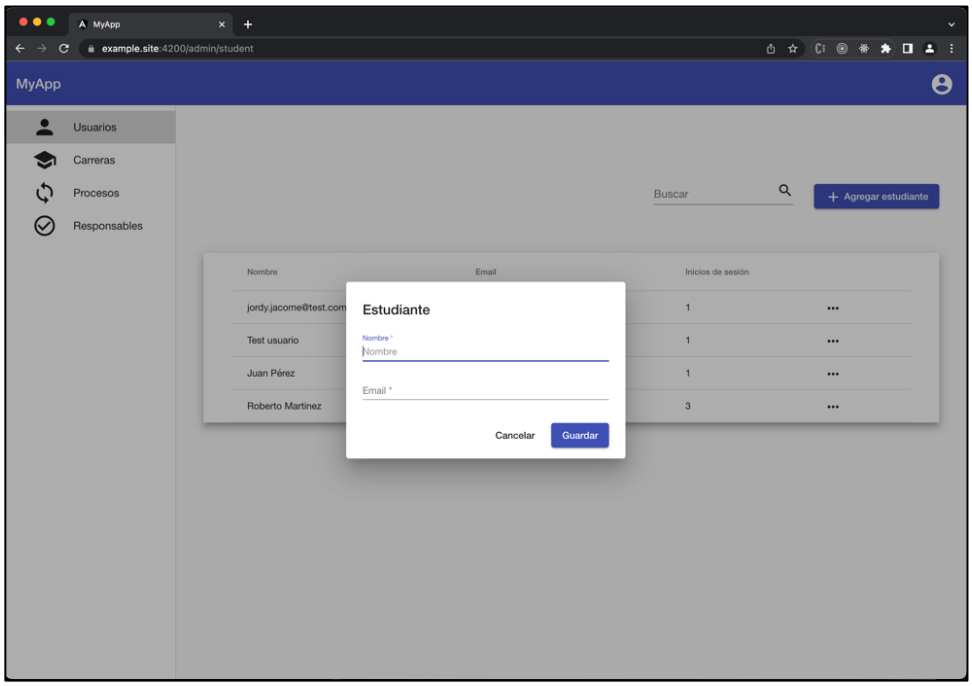

Figura 22. Maquetado de formulario de creación de usuarios

- **Sprint Review**

  Al finalizar este sprint, se validaron los criterios de aceptación de la historia desarrollada. Los resultados obtenidos se resumen a continuación en la tabla 25.

Historia	Observación
HF004	Se creó correctamente el módulo de administración, al principio del desarrollo se presentaron inconvenientes implementando la lógica necesaria para obtener el token de acceso para el API de Auth0, sin embargo, se pudo resolver este problema sin mayor contratiempo. No se implementaron las funciones de CRUD al 100% debido a que Auth0 también cuanta con un dashboard de administración que ya tiene todas estas operaciones implementadas y más funcionalidades como verificación de email, reseteo de contraseña, activación de inicio de sesión con autenticación de 2 pasos, etc. El usuario no está recibiendo el mail con el enlace para completar su proceso de verificación de cuenta y cambio de contraseña. Este problema persiste debido a los inconvenientes encontrados en el sprint 1.

Tabla 25. Resultados – Sprint Review 5

- **Sprint Retrospective**

  Durante el desarrollo de este sprint solamente se tuvo un inconveniente obteniendo el token de autenticación para el API de Auth0. Fuera de ello, no se presentaron mayores problemas. Además, se tenía provisto no realizar una implementación de un CRUD al 100% para esta historia porque las funcionalidades de administración de usuarios ya están provistas por el dashboard de administración de Auth0.

  También fue necesario revisar el desarrollo de las historias del sprint 1 ya que esta historia complementa el flujo de implementación de usuarios que se había definido inicialmente. La causa principal del problema reside en que se necesita un dominio registrado y un servidor de correo para que se complete el flujo de autenticación. Se probó con una cuenta de correo custom pero el servidor de correo de Auth0 rechaza estas peticiones desde

este correo debido a que no está públicamente registrado. El resumen de avance de este sprint se encuentra detallado en la figura 23.

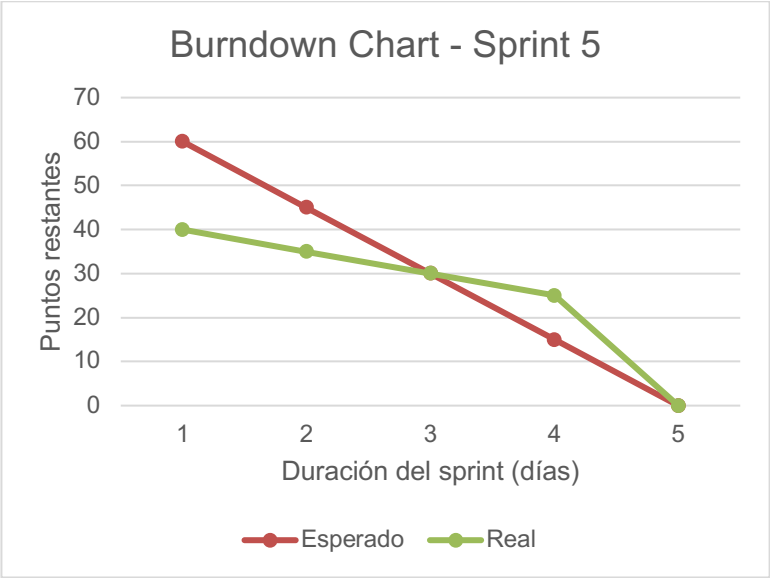

Figura 23. Burndown Chart - Sprint 5

3.8.8 Sprint 6

- **Objetivo del sprint**: implementar el módulo de estudiantes en donde se pueda visualizar la información de carrera y procesos.

- **Sprint backlog**

  Para el desarrollo de este sprint se tiene la historia HF003 detallada en la tabla 26.

Historia de Usuario	HF003		
Descripción	Como estudiante, deseo visualizar la información de carrera, proceso y subprocesos en los que estoy inscrito. En caso de no tener registro, el sistema debe permitirme escoger una carrera y un proceso para inscribirme.		
Prioridad	Alta	**Costo**	40

Criterios de aceptación
• Una vez ingresado, el sistema permite visualizar información general de mi proceso. • Si se inicia sesión por primera vez después del registro, el sistema permitirá escoger una carrera y proceso.
Tareas a realizar
• En el módulo de estudiantes del proyecto del front-end, maquetar la vista según corresponda. (Revisar mock-up). • Después del inicio de sesión, validar los datos que vienen en el request del back-end. En caso de tener datos en el objeto *management* que contiene los datos sobre la carrera y procesos asignados a un estudiante, leerlos y llenar el formulario de la vista principal. Caso contrario, desplegar un formulario de registro para una carrera y proceso. • Añadir un loader al formulario de datos de procesos para mostrar una carga mientras se obtienen los datos desde el back-end. • Debido a que este módulo es netamente informativo, desactivar las entradas de los formularios y únicamente habilitarlos para su lectura. • Usar el servicio de API y crear las funciones necesarias para consumir los métodos del back-end usando las variables de ambiente creadas anteriormente para consumir los datos del usuario. • Validar con los mock-ups las opciones de menú lateral que se mostrarán en el módulo de estudiantes.

Tabla 26. Historia de usuario HF003

- **Daily Scrum**

  El resumen de daily scrum se encuentra debidamente detallado en la tabla 27.

Daily Scrum – Sprint 6			
Día	**¿Qué se hizo ayer que ayudó al equipo a lograr el objetivo del sprint?**	**¿Qué se hará hoy para ayudar al equipo de desarrollo a lograr el objetivo del sprint?**	**¿Se ve algún impedimento que evite que el equipo logre el objetivo del sprint?**
1	N/A	Se maquetará la interfaz principal en el módulo de estudiantes.	N/A
2	Se maquetó la interfaz principal en el módulo de estudiantes.	Se harán las validaciones correspondientes para mostrar el formulario de registro de carrera por primera vez.	N/A
3	Se validó el escenario en el que el usuario que inicia sesión por primera vez no cuenta con un registro de carrera.	Se probará el flujo de inicio de sesión por primera vez y se revisarán las reglas de autenticación de usuarios	N/A
4	Se probó y revisó el flujo de autenticación de usuarios.	Se intentará corregir el error de flujo de autenticación de usuarios.	N/A

Día	¿Qué se hizo ayer que ayudó al equipo a lograr el objetivo del sprint?	¿Qué se hará hoy para ayudar al equipo de desarrollo a lograr el objetivo del sprint?	¿Se ve algún impedimento que evite que el equipo logre el objetivo del sprint?
5	Se intentó revisar diferentes formas para la implementación del flujo de autenticación de usuarios.	Pruebas unitarias y validación general del módulo de estudiantes.	N/A

Tabla 27. Daily Scrum – Sprint 6

- **Codificación**

  En las figuras 21 y 22 se puede observar el resultado del maquetado y creación de formularios especificados previamente en las historias de usuario.

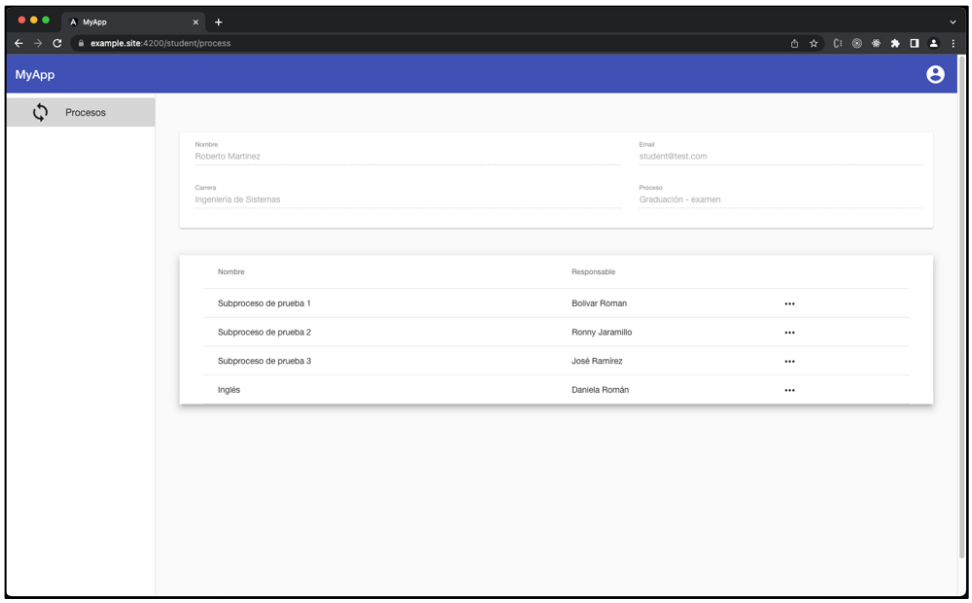

Figura 24. Maquetado principal del módulo de estudiantes

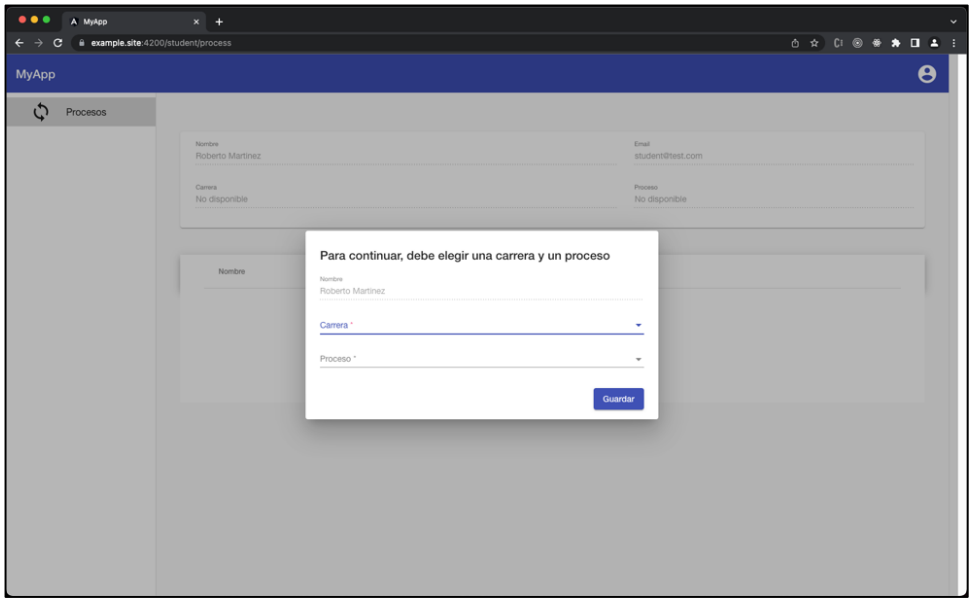

Figura 25. Formulario de registro de usuario por primera vez

- **Sprint Review**

  Al finalizar este sprint, se validaron los criterios de aceptación de la historia desarrollada. Los resultados obtenidos se resumen a continuación en la tabla 28.

Historia	Observación
HF003	Se creó correctamente el módulo de estudiantes, sin mayores contratiempos. Se revisaron los inconvenientes presentados en los sprints previos y se intentó darles solución a estos problemas sin éxito, sin embargo, se propone una solución para este problema en las conclusiones de este documento.

Tabla 28. Resultados – Sprint 6

- **Sprint Retrospective**

  Durante el desarrollo de este sprint no se presentaron mayores inconvenientes. Se completaron las tareas asignadas en este sprint en los primeros días y el tiempo restante se empleó en revisar los inconvenientes encontrados anteriormente. El resumen de avance de este sprint se encuentra detallado en la figura 26.

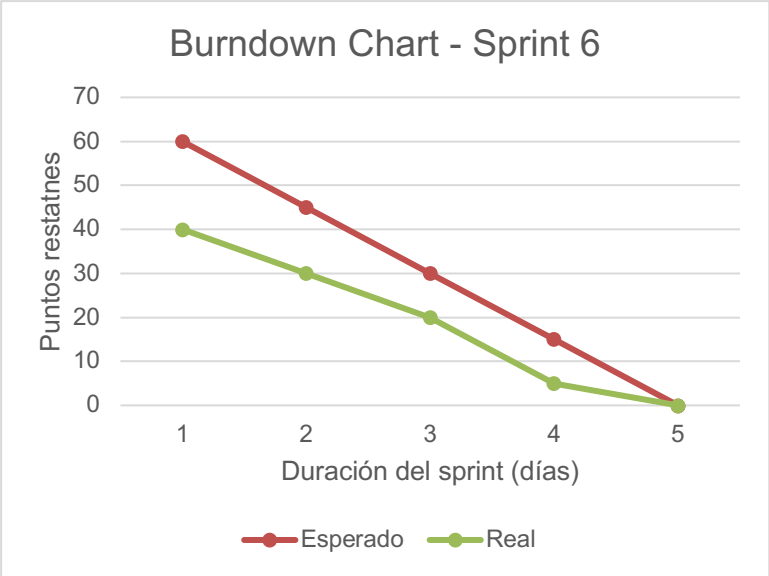

Figura 26. Burndown Chart – Sprint 6

4 RESULTADOS

Las pruebas de software definen si es que un producto cumple o no con las funciones para las que fue creado. En simples palabras, estas pruebas pueden definirse en un documento que defina lo que un usuario puede hacer o no, incluyendo escenarios de la vida real en donde el software será puesto a prueba [21]. Las pruebas de funcionalidad se pueden realizar mediante 2 técnicas populares:

- **Pruebas basadas en requisitos**
  Estas pruebas contienen las especificaciones funcionales que forman una base para todas las pruebas a realizar.

- **Pruebas basadas en escenarios de negocio**
  Contiene la información sobre cómo se percibirá el sistema desde una perspectiva de proceso de negocio.

Este tipo de pruebas tiene varias categorías y pueden ser usadas según el escenario, entre las más destacadas tenemos:

- **Pruebas unitarias**
  Estas pruebas generalmente son escritas por el equipo de desarrollo en donde se escriben diferentes escenarios que prueban directamente el código que se escribió de manera aislada de otros módulos de código. Por lo general, se escriben pruebas que llaman a una función o método en específico que después validan parámetros de entrada y/o salida, así como datos de retorno esperados. Estas pruebas evalúan línea a línea el código escrito. [21]

- **Pruebas de usabilidad (aceptación)**
  Estas pruebas exponen el producto creado a un ambiente real en donde se prueba que el software cumpla con las funciones para las que fue creado. Generalmente estas pruebas son comparadas con las pruebas de aceptación del usuario y normalmente se hacen en un sistema completo o cuando todos los módulos de este están completamente integrados. [21]

4.1 Pruebas unitarias

Tanto para el proyecto del front-end, como el back-end, se escribieron pruebas unitarias durante el desarrollo de cada sprint. En las figuras 27 y 28 se puede observar un resumen de su ejecución.

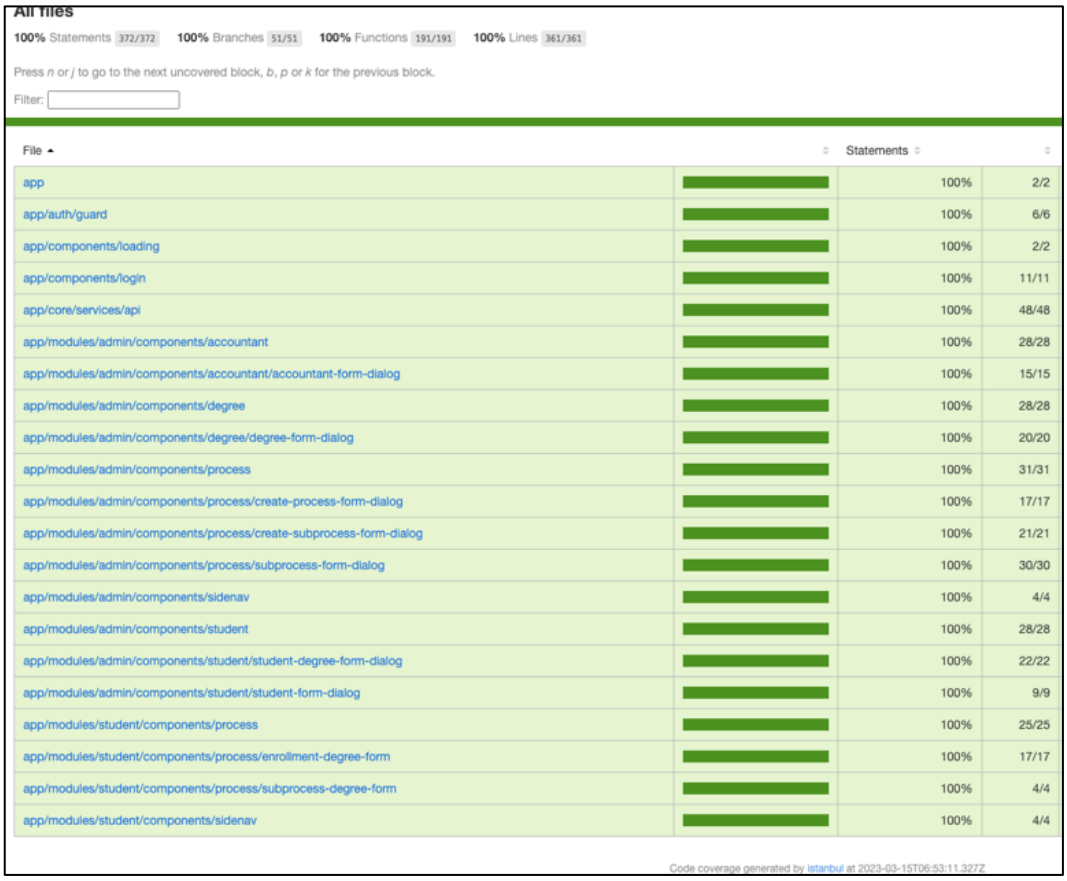

Figura 27. Resumen de pruebas unitarias en el front-end.

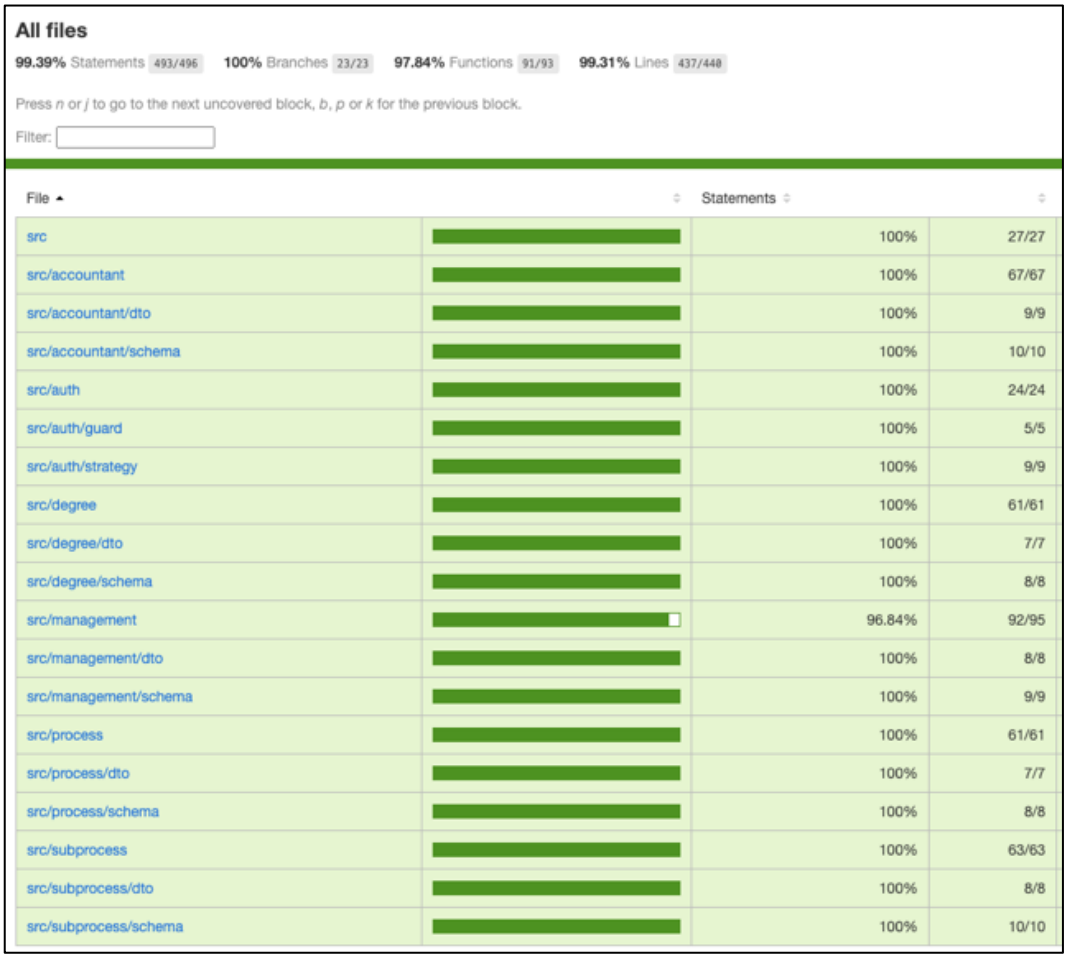

Figura 28. Resumen de pruebas unitarias en el back-end.

4.2 Pruebas de aceptación

Las pruebas de aceptación de un software son una buena práctica dentro de la industria del desarrollo de software. Cuando los desarrolladores escriben pruebas unitarias del código prueban que este cumpla su propósito. Las pruebas de aceptación verifican que el sistema se comporte de manera esperada, las pruebas de aceptación muchas veces también son conocidas como pruebas funcionales.
Estas pruebas representan los intereses de los clientes y le brindan un grado de confianza en el que se verifica que el sistema desarrollado cumpla sus objetivos. Por ejemplo, una prueba de aceptación podría verificar que, eliminando un elemento de una tabla, actualiza y elimina correctamente este elemento. [22]

En consecuencia, en las siguientes tablas se han especificado una serie de pruebas de aceptación del sistema y se verifica si se cumplió o no con el objetivo de dicha prueba. Además, estas pruebas se han realizado de manera incremental durante el

desarrollo del proyecto, es por esto que han sido divididas por sprints. Los casos de prueba marcados en color verde se cumplieron exitosamente mientras que los casos marcados en rojo no.

4.2.1 Pruebas de aceptación – Sprint 1

Las pruebas de aceptación correspondientes al sprint 1 se encuentran en la tabla 29.

Sprint 1				
Código	**Descripción**	**Prerrequisitos**	**Resultados**	
			Esperado	**Obtenido**
CP001	Verificar que el sistema permita ingresar con un correo electrónico y una contraseña	Usuario y contraseña de un usuario ya registrado.	El sistema permite ingresar a un usuario.	El sistema permitió ingresar a un nuevo usuario.
CP002	Verificar que el sistema permita registrarse con un correo electrónico y una contraseña	N/A	El sistema permite registrar a un usuario.	El sistema permitió registrar a un usuario.
CP003	Verificar que el sistema muestre un error cuando se quiere hacer un	Usuario y contraseña de un usuario ya registrado.	El sistema muestra error cuando se registra un usuario ya registrado.	El sistema mostró error cuando se registró un usuario

	registro con un correo de una cuenta ya existente			ya registrado.
CP004	Verificar que el sistema envíe un correo de invitación cuando el administrador registra un usuario	N/A	El sistema debe enviar un correo de invitación cuando el administrador agrega un usuario.	No se envió el correo, se mostró rechazado por el servidor SMTP.
CP005	Verificar que el sistema envía un correo de cambio de contraseña cuando un usuario se registra por sí mismo.	N/A	El sistema debe enviar un correo de cambio de contraseña después de que un usuario se registre.	No se envió el correo, se mostró rechazado por el servidor SMTP.

Tabla 29. Pruebas de aceptación – Sprint 1

4.2.2 Pruebas de aceptación – Sprint 2

Las pruebas de aceptación correspondientes al sprint 2 se encuentran en la tabla 30.

Sprint 2				
Código	**Descripción**	**Prerrequisitos**	**Resultados**	
			Esperado	**Obtenido**
CP006	Verificar que el sistema permita registrar un nuevo encargado en el sistema.	El administrador debe iniciar sesión previamente.	El sistema permite ingresar un nuevo encargado.	El sistema permitió ingresar un nuevo encargado.
	Verificar que el sistema permita visualizar a los encargados registrados.	El administrador debe iniciar sesión previamente y se debe tener un encargado registrado.	El sistema permite visualizar los datos de un encargado ya registrado.	El sistema permitió visualizar los datos de un encargado ya registrado.
CP007	Verificar que el sistema permita editar los datos de un encargado.	El administrador debe iniciar sesión previamente y se debe tener un encargado registrado.	El sistema permite editar los datos de un encargado ya registrado.	El sistema permitió editar los datos de un encargado ya registrado.

CP008	Verificar que el sistema permita eliminar a un encargado.	El administrador debe iniciar sesión previamente y se debe tener un encargado registrado.	El sistema permite eliminar los datos de un encargado ya registrado.	El sistema permitió eliminar los datos de un encargado ya registrado.

Tabla 30. Pruebas de aceptación – Sprint 2

4.2.3 Pruebas de aceptación – Sprint 3

Las pruebas de aceptación correspondientes al sprint 3 se encuentran en la tabla 31.

Sprint 3				
Código	**Descripción**	**Prerrequisitos**	**Resultados**	
			Esperado	**Obtenido**
CP009	Verificar que el sistema permita registrar un nuevo proceso en el sistema.	El administrador debe iniciar sesión previamente.	El sistema permite ingresar un nuevo proceso.	El sistema permitió ingresar un nuevo proceso.
CP010	Verificar que el sistema permita visualizar los procesos registrados.	El administrador debe iniciar sesión previamente y se debe tener	El sistema permite visualizar los datos de un	El sistema permitió visualizar los datos de un

		un proceso registrado.	proceso ya registrado.	proceso ya registrado.
CP011	Verificar que el sistema permita editar los datos de un proceso.	El administrador debe iniciar sesión previamente y se debe tener un proceso registrado.	El sistema permite editar los datos de un proceso ya registrado.	El sistema permitió editar los datos de un proceso ya registrado.
CP012	Verificar que el sistema permita eliminar un proceso.	El administrador debe iniciar sesión previamente y se debe tener un proceso registrado.	El sistema permite eliminar los datos de un proceso ya registrado.	El sistema permitió eliminar los datos de un proceso ya registrado.
CP013	Verificar que el sistema permita registrar un nuevo subproceso en el sistema.	El administrador debe iniciar sesión previamente y se debe tener un proceso registrado.	El sistema permite ingresar un nuevo subproceso	El sistema permitió ingresar un nuevo subproceso
CP014	Verificar que el sistema permita visualizar los subprocesos	El administrador debe iniciar sesión previamente y se debe tener	El sistema permite visualizar los datos de un subproceso	El sistema permitió visualizar los datos de un subproceso

	de un proceso.	un proceso y subproceso registrado.	ya registrado.	ya registrado.
CP015	Verificar que el sistema permita editar los datos de un subproceso de un proceso.	El administrador debe iniciar sesión previamente y se debe tener un proceso y subproceso registrado.	El sistema permite editar los datos de un subproceso ya registrado.	El sistema permitió editar los datos de un subproceso ya registrado.
CP016	Verificar que el sistema permita eliminar un subproceso de un proceso.	El administrador debe iniciar sesión previamente y se debe tener un proceso y subproceso registrado.	El sistema permite eliminar los datos de un subproceso ya registrado.	El sistema permitió eliminar los datos de un subproceso ya registrado.

Tabla 31. Pruebas de aceptación – Sprint 3

4.2.4 Pruebas de aceptación – Sprint 4

Las pruebas de aceptación correspondientes al sprint 4 se encuentran en la tabla 32.

Sprint 4				
Código	**Descripción**	**Prerrequisitos**	**Resultados**	
			Esperado	**Obtenido**
CP017	Verificar que el sistema permita registrar una nueva carrera.	El administrador debe iniciar sesión previamente y se debe tener registrado un proceso.	El sistema permite registrar una nueva carrera.	El sistema permitió registrar una nueva carrera.
CP018	Verificar que el sistema permita visualizar una carrera registrada.	El administrador debe iniciar sesión previamente y se debe tener una carrera y proceso registradas.	El sistema permite visualizar una carrera registrada.	El sistema permitió visualizar una carrera registrada.
CP019	Verificar que el sistema permita editar una carrera ya registrada.	El administrador debe iniciar sesión previamente y se debe tener una carrera registrada.	El sistema permite editar una carrera ya registrada.	El sistema permite editar una carrera ya registrada.

CP020	Verificar que el sistema permita eliminar una carrera ya registrada.	El administrador debe iniciar sesión previamente y se debe tener una carrera registrada.	El sistema permite eliminar una carrera ya registrada.	El sistema permitió eliminar una carrera ya registrada.

Tabla 32. Pruebas de aceptación – Sprint 4

4.2.5 Pruebas de aceptación – Sprint 5

Las pruebas de aceptación correspondientes al sprint 5 se encuentran en la tabla 33.

Sprint 5				
Código	**Descripción**	**Prerrequisitos**	**Resultados**	
			Esperado	**Obtenido**
CP021	Verificar que el sistema permita registrar un usuario con un nombre y una contraseña.	El administrador debe iniciar sesión previamente.	El sistema permite eliminar una carrera ya registrada.	El sistema permitió eliminar una carrera ya registrada.
CP022	Verificar que el sistema permita visualizar a los usuarios registrados.	El administrador debe iniciar sesión previamente y debe haber al menos un	El sistema permite visualizar a los usuarios registrados.	El sistema permitió visualizar a los usuarios registrados.

		usuario registrado.		
CP023	Verificar que el sistema permita asignar una carrera a un usuario.	El administrador debe iniciar sesión previamente y debe haber un usuario ya registrado.	El sistema permite asignar una carrera a un usuario.	El sistema permitió asignar una carrera a un usuario.
CP024	Verificar que el sistema permita eliminar a un usuario registrado.	El administrador debe iniciar sesión previamente y debe haber un usuario ya registrado.	El sistema permite eliminar un usuario.	El sistema permitió eliminar un usuario.

Tabla 33. Pruebas de aceptación – Sprint 5

4.2.6 Pruebas de aceptación – Sprint 6

Las pruebas de aceptación correspondientes al sprint 6 se encuentran en la tabla 34.

Sprint 6				
Código	**Descripción**	**Prerrequisitos**	**Resultados**	
			Esperado	**Obtenido**
CP025	Verificar que el sistema permita registrarse en una	El estudiante debe iniciar sesión previamente y no debe estar	El sistema permite registrar una carrera a un	El sistema permitió registrar una carrera a un

	carrera cuando no ha sido asignada ninguna carrera previamente.	registrado en una carrera aún.	usuario nuevo.	usuario nuevo.
CP026	Verificar que el sistema permita visualizar la información de la carrera registrada.	El estudiante debe iniciar sesión previamente y debe tener registrada una carrera.	El sistema permite visualizar la información de carrera registrada.	El sistema permitió visualizar la información de carrera registrada.

Tabla 34. Pruebas de aceptación – Sprint 6

5 CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

Tras haber finalizado el desarrollo de esta tesis, es pertinente evaluar el cumplimiento de los objetivos planteados al inicio de este documento. Se ha desarrollado un sistema que permite gestionar los trabajos de titulación dentro de la facultad de sistemas de la EPN. Además, a lo largo de este documento se han definido aspectos específicos para su implementación como su arquitectura y decisiones para el uso de diferentes herramientas. Al final del desarrollo, se realizaron pruebas funcionales y pruebas unitarias que evalúan el código y el comportamiento del sistema. Adicionalmente, se han tomado en cuenta una serie de consideraciones que se establecen a continuación:

- El desarrollo de este sistema ha sido guiado tomando en cuenta la necesidad de los estudiantes que deben seguir un proceso en general dentro de la facultad de sistemas. Gracias a su diseño, este sistema se puede adaptar no solamente a procesos de titulación dentro de la facultad sino para darle seguimiento a procesos de forma más genérica dentro de la EPN.
- El uso de un gestor de autenticación como Auth0 permite añadir a un sistema características de inicio de sesión de diferentes proveedores incluyendo los proveedores de correo de la EPN, autenticación de 2 pasos, verificación de correo electrónico y número telefónico asociado a una cuenta, etc. De esta manera, también se provee un control sobre los recursos del proyecto, como el API que puede ser fácilmente reutilizada por otra aplicación.
- Algunas de las características mencionadas anteriormente pueden ser habilitadas solamente en la versión de pago de esta herramienta, el sistema es factible de esta implementación en la posteridad siempre y cuando se cuente con la versión de pago de Auth0.
- El desarrollo de esta herramienta no representa de ninguna manera la automatización para el envío de documentos y culminación de un proceso determinado, para ello, se puede emplear otro tipo de software que se encuentra listo y desarrollado para este objetivo.

- El uso de la metodología SCRUM para el desarrollo de este proyecto nos permite ser flexibles al momento de implementar cambios requeridos por los clientes o por el sistema en sí. La implementación de revisiones al final de cada sprint y el análisis de la retrospectiva permite mejorar la forma de trabajo a lo largo de los sprints. De esta manera, se trabaja eficientemente y prestando atención a todas las variables que pueden cambiar durante el desarrollo.
- Las pruebas de aceptación y pruebas unitarias del código aseguran que se cumple el objetivo del sistema. Sin embargo, las pruebas que no fueron aceptadas no se completaron exitosamente debido a que este sistema no se probó en un ambiente productivo. Para ello, se debe contar con un dominio de internet debidamente registrado, así como una plataforma de software que permita desplegar el API y el front-end para que el sistema sea accesible desde la web.
- Debido a que el proveedor de autenticación fue configurado en un ambiente local, al momento de iniciar sesión encontramos una alerta sobre los certificados que se usan para establecer la conexión segura del servidor de front-end y back-end. Estos certificados deben ser administrados correctamente y almacenados en el servicio de software que permita desplegar la aplicación.

5.2 Recomendaciones

Para incrementar la seguridad de los usuarios, se recomienda habilitar el inicio de sesión de diferentes proveedores que ya cuentan con otras capas de verificación y autenticación.

Para un despliegue correcto de la aplicación y para que esta sea accesible por la web, se recomienda revisar e implementar las mejores prácticas de la industria y usar proveedores de este servicio como Heroku.

6 REFERENCIAS BIBLIOGRÁFICAS

[1] "REGLAMENTO DE RÉGIMEN ACADÉMICO DE LA ESCUELA POLITÉCNICA NACIONAL (EPN)." Escuela Politécnica Nacional, abril 2021. [Online]. Available: https://www.epn.edu.ec/wp-content/uploads/2021/04/reglamento_de_re%CC%81gimen_acade%CC%81mico_epn_reforma_abril_2021.pdf

[2] G. Rojas and S. Merino, "PROTOTIPO DE APLICACIÓN WEB PARA LA AUTOMATIZACIÓN DE LOS TRÁMITES ESTUDIANTILES INGRESADOS EN LAS SECRETARÍAS DE LAS CARRERAS INGENIERÍA EN SISTEMAS COMPUTACIONALES E INGENIERÍA EN NETWORKING Y TELECOMUNICACIONES DE LA UNIVERSIDAD DE GUAYAQUIL," Universidad de Guayaquil, Guayaquil, 2018. [Online]. Available: http://repositorio.ug.edu.ec/bitstream/redug/28218/1/B-CISC-PTG.%201500%20Rojas%20Alvarado%20Giannella%20Patricia.pdf

[3] "LEY ORGÁNICA PARA LA OPTIMIZACIÓN Y EFICIENCIA DE TRÁMITES ADMINISTRATIVOS." ASAMBLEA NACIONAL REPÚBLICA DEL ECUADOR, Oct. 16, 2018. [Online]. Available: https://www.gob.ec/sites/default/files/regulations/2020-03/Documento_%20LEY-ORG%C3%81NICA-OPTIMIZACION-EFICIENCIA-TRAMITES.pdf

[4] J. Conallen, "Modeling web application architectures with UML," *Communications of the ACM*, vol. 42, no. 10, pp. 63–70, 1999.

[5] ardalis, "Overview of ASP.NET Core MVC." https://docs.microsoft.com/en-us/aspnet/core/mvc/overview (accessed Sep. 21, 2021).

[6] J. Deacon, "Model-view-controller (mvc) architecture," *Online][Citado em: 10 de março de 2006.] http://www. jdl. co. uk/briefings/MVC. pdf*, vol. 28, 2009.

[7] R. S. Pressman, *INGENIERIA DE SOFTWARE*. McGraw-Hill Interamericana de España S.L., 2010. [Online]. Available: https://books.google.com.ec/books?id=deuwcQAACAAJ

[8] M. Biehl, *API Architecture*, vol. 2. API-University Press, 2015.

[9] E. A. Scott Jr, *SPA Design and Architecture: Understanding single-page web applications*. Simon and Schuster, 2015.

[10] K. Schwaber and J. Sutherland, "The scrum guide," *Scrum Alliance*, vol. 21, no. 19, p. 1, 2011.

[11] Atlassian, "User Stories | Examples and Template," *Atlassian*. https://www.atlassian.com/agile/project-management/user-stories (accessed Jun. 17, 2022).

[12] "Documentation | NestJS - A progressive Node.js framework," *Documentation | NestJS - A progressive Node.js framework*. https://docs.nestjs.com (accessed Jun. 19, 2022).

[13] "Angular - What is Angular?" https://angular.io/guide/what-is-angular (accessed Sep. 21, 2021).

[14] Auth0, "Auth0 Overview," *Auth0 Docs*. https://auth0.com/docs/ (accessed Nov. 25, 2022).

[15] "What is Auth0? — ForwardAuth for Auth0 documentation." https://traefik-forward-auth0.readthedocs.io/en/latest/auth0/auth0.html (accessed Nov. 25, 2022).

[16] "Why You Should Use Typescript for Your Next Project," *Codemotion Magazine*, Feb. 23, 2022. https://www.codemotion.com/magazine/backend-dev/why-you-should-use-typescript-for-your-next-project/ (accessed Jun. 19, 2022).

[17] "What Is MongoDB?," *MongoDB*. https://www.mongodb.com/what-is-mongodb (accessed Mar. 06, 2023).

[18] “Getting started with WebStorm | WebStorm,” *WebStorm Help*. https://www.jetbrains.com/help/webstorm/getting-started-with-webstorm.html (accessed Mar. 06, 2023).
[19] “What Is GitHub? A Beginner’s Introduction to GitHub,” *Kinsta*®, Dec. 13, 2022. https://kinsta.com/knowledgebase/what-is-github/ (accessed Mar. 06, 2023).
[20] K. S. Rubin, *Essential Scrum: A practical guide to the most popular Agile process*. Addison-Wesley, 2012.
[21] J. A, “Functional Testing: A Complete Guide with Types and Example,” *Software Testing Help*, Mar. 23, 2023. https://www.softwaretestinghelp.com/guide-to-functional-testing/ (accessed Apr. 05, 2023).
[22] R. Miller and C. T. Collins, “Acceptance testing,” *Proc. XPUniverse*, vol. 238, 2001.

6.1 ANEXOS

6.1.1 **Anexo 1**. Enlace de mock-ups iniciales del sistema.

https://www.figma.com/file/AM0QIAy5WR2LfrNFwG0FgE/Tesis?node-id=0%3A1&t=fD7Gr4uw5EW2ulP9-1

6.1.2 **Anexo 2**. Enlace de repositorio de código para el front-end.

https://www.figma.com/file/AM0QIAy5WR2LfrNFwG0FgE/Tesis?node-id=0%3A1&t=fD7Gr4uw5EW2ulP9-1

6.1.3 **Anexo 3**. Enlace de repositorio de código para el back-end.

https://www.figma.com/file/AM0QIAy5WR2LfrNFwG0FgE/Tesis?node-id=0%3A1&t=fD7Gr4uw5EW2ulP9-1