Actualmente en la facultad de ingeniería y específicamente en el departamento de Ingeniería de Sistemas la gestión curricular es poco eficiente, difícilmente gestionable bajo estándares de calidad y para nada amigable con los docentes y administrativos que hacen uso de ella. Es por esto que en el presente documento mostraremos a detalle la arquitectura y diseño de una plataforma web dedicada exclusivamente a la gestión de la información del micro currículo de la Universidad de Antioquia, con énfasis en el Departamento de Ingeniería de Sistemas.

# Documento de Arquitectura

## SIMOCU

Juan Sebastián Peláez Villa
Deyber Sepúlveda
Raúl Andrés Arcila
David Alejandro Marín
Fredy Esteban Arroyave
Diego Iván Oliveros

# Contenido

## Tabla de control de versiones

| Nombre del documento | Acción | Fecha | Nombre | Rol |
|---|---|---|---|---|
| Arquitectura.pdf | Emisión | Fecha | Equipo de arquitectura | Arquitectos y diseñadores |
| 20191209_ArquitecturaV2F.docx | Revisión | 09-12-2019 | Diego Iván Oliveros Acosta | Director |
| ArquitecturaV3F.docx | Revisión | 10-10-2019 | Equipo de arquitectura | Arquitectos y diseñadores |
| | | | | |

# Introducción

Por medio del proyecto SIMOCU se busca llevar a cabo el desarrollo de una plataforma web cuyo objetivo es gestionar la información del micro, meso y maso currículo de los programas de pregrado de la Universidad de Antioquia, con énfasis especial e inicial el programa de Ingeniería de Sistemas administrado por el Departamento de Ingeniería de Sistemas.

En el presente documento se expone y argumenta la arquitectura propuesta por el equipo de *Arquitectura y Diseño*. Primero se realiza una investigación, clasificación y evaluación de plataformas y tecnologías opcionadas en las cuales se podría desarrollar el proyecto. Seguidamente se selecciona una de ellas, tanto para el Frontend como para el Backend.

Adicionalmente para poder cumplir a cabalidad los requisitos no funcionales propuestos para el proyecto se modelan y explican un conjunto de diagramas entre los cuales se encuentra: Diagrama de paquetes, diagrama de despliegue, diagrama de clases, diagrama de actividades, entre otros.

De forma adicional, para tener una adecuada implementación de la arquitectura se propone un conjunto de patrones de diseño.

# Objetivos

El proyecto SIMOCU tiene como objetivo general:

> Elaborar una herramienta para la gestión de la información para el desarrollo del currículo con el fin de apoyar los procesos del comité curricular (micro meso macro) y los procesos de acreditación.

Adicionalmente, sus objetivos específicos son:

- Levantamiento de requerimientos y estimación de los siguientes módulos
- Administración y control de micro currículo
- Administración y registro de plan de estudio
- Reportes y diseño gráfico del proceso macro curricular
- Modelado de procesos Meso currículo
- Desarrollo, integración y prueba de los módulos
- Integración con los sistemas de la Universidad
- Despliegue de la herramienta y comercialización.

# EVALUACIÓN DE PLATAFORMAS Y TECNOLOGÍAS

Una plataforma es un sistema que sirve como base para hacer funcionar diferentes componentes de un aplicativo y normalmente se trata de una combinación de sistema operativo y lenguaje de programación. El proceso de desarrollo de un producto de software siempre incluye la elección y evaluación de una plataforma, la más adecuada para el tipo de aplicativo que se quiere construir. Normalmente la elección de una plataforma está ligada a los conocimientos técnicos del equipo, la capacidad adquisitiva de la empresa y la experiencia del arquitecto. Teniendo en cuenta esto, las instrucciones proporcionadas por el equipo de gestión y el dueño del producto, en este caso el profesor, se determinó la plataforma y las

tecnologías con las cuales se desarrollará el proyecto. Estas instrucciones indican que el Backend se desarrollará en Java y la tecnología para el Frontend será Angular.
En este documento se evaluarán las tecnologías anteriormente mencionadas y el porqué de su uso en el proyecto.

## Plataforma y Tecnología para el Backend

Cómo se mencionó anteriormente el Backend del proyecto se desarrollará en el lenguaje de programación Java y dado que es un proyecto web se debe usar la versión **JAVA EE.**
Java EE es una arquitectura multicapa (Esto quiere decir que podemos separar el desarrollo de la aplicación en diferentes capas según su función) para implementar aplicaciones de tipo empresarial y aplicaciones basadas en la Web. Esta tecnología soporta una gran variedad de tipos de aplicaciones desde aplicaciones Web de gran escala a pequeñas aplicaciones cliente-servidor.
Java EE cuenta con una serie de servidores de aplicaciones que permiten desplegar de manera ágil y simplificada cualquier aplicación, entre sus principales exponentes están:
- Glassfish
- Oracle Web Logic
- Tomcat
- JBoss

Es fundamental establecer parámetros que permitan dar más peso al criterio de selección de tecnología, es por esto que para fortalecer los argumentos a favor del uso de Java como tecnología para el Backend se deja en evidencia los siguientes parámetros:
- Licenciamiento: Java es un lenguaje de programación gratuito, no requiere de licencias específicas para el proyecto que se va a desarrollar.
- Documentación: La comunidad de Java a nivel mundial es gigantesca, cuenta con uno de los repositorios de información más grande y completo.
- Facilidad de uso: Dado que la curva de aprendizaje es fundamental, Java es un lenguaje de programación que se aprende desde que se empieza a programar, así que la mayoría de los desarrolladores conocen su funcionamiento.

- Escalabilidad: En la industria tecnológica cuando se construye algo no se puede dejar a un lado las proyecciones futuras, Java ha demostrado que es una de las plataformas más estables a la hora crecer.
- Manejo de errores: Java permite la captura y personalización de los errores para un adecuado tratamiento y presentación de estos.
- Manejo de versiones: Java cuenta con un historial de actualizaciones bastante documentado y de transición poco brusca, lo cual permite usar las últimas versiones sin miedo a quedar obsoletos por la dificultad de migración.

Finalmente, dado que el Backend que se construirá será orientado a servicios, se hará uso de un Framework llamado Spring, ya que dada su flexibilidad es posible construir desde Apis REST, WebSockets, hasta Backend con micro servicios. Cabe resaltar que de lado del acceso a datos permite integrar bases de datos relacionales (SQL) y no relacionales (Nosql).

## Plataforma y Tecnología para el Frontend

Cómo se mencionó anteriormente el Frontend del proyecto se desarrollará por medio del framework Angular.
Este framework fue desarrollado y actualmente está soportado por Google. Angular se ha venido posicionando en los primeros lugares como framework por preferencia para la construcción de SPA

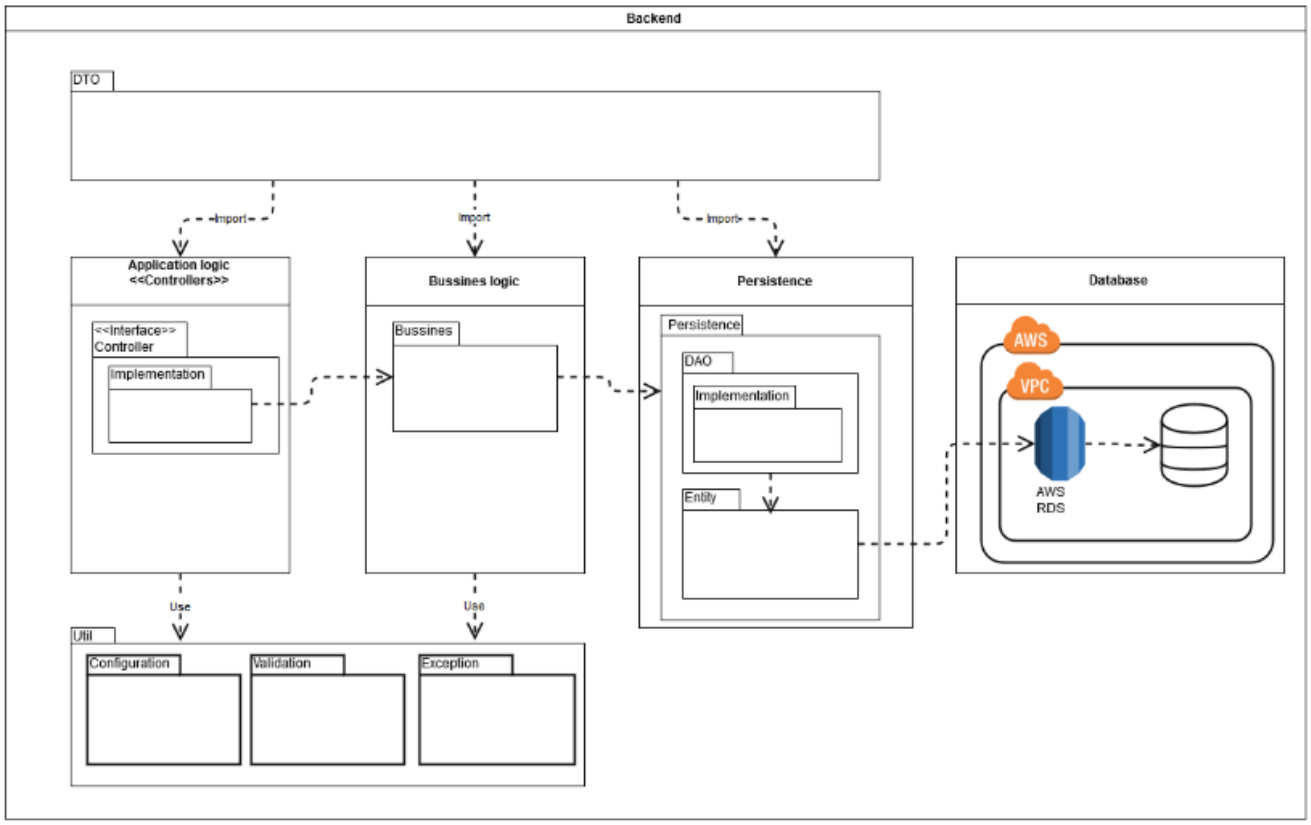
(Single Page Aplicación) sobresaliendo por encima de sus competidores por su rápida evolución y adaptación de nuevas funcionalidades que hace de esta herramienta una de las más robustas. Desde la liberación de la versión 2, Type Script se convierte en el lenguaje de programación base de Angular dándole así la posibilidad de integrar e implementar el paradigma objetual del lado del Frontend. Además del Core cuenta con librerías auxiliares como los es Angular Material, también cuenta con su propio CLI (Command-Line Interface) lo que facilita la construcción de diversos elementos como servicios, componentes, módulos, entre otros.

Las características de Angular con las cuales se puede concluir que es una tecnología apta para el proyecto que se va a desarrollar son:

- Productividad: Angular tiene una de las mejores relaciones de trabajo/carga y tiempo/carga en todo el espectro de tecnologías web.
- Percepción del Desarrollador: La mayoría de los desarrolladores web Frontend, determinaron que Angular es ideal para desarrollar SPA, transmite comodidad y confianza.
- Curva de Aprendizaje: Basta con conocer el funcionamiento básico de HTML y JavaScript para usar este framework, no se requiere de un gran conocimiento a profundidad.
- Popularidad: Este framework es altamente reconocido y referenciado por las críticas positivas con las que cuenta en la comunidad.
- Plantillas: Gracias a CLI Angular es un framework dispuesto para que cualquiera construya una gran app sobre la base predefinida proporcionada por su equipo desarrollador.
- Componentes: Todos los elementos de los que dispone el framework son útiles para el proyecto y se integran con facilidad a diferentes entornos.
- Escalabilidad: Debido a sus componentes Angular es un framework con una gran capacidad de escalar con respecto al tiempo y las necesidades.

# ARQUITECTURA PROPUESTA PARA EL BACKEND Y EL FRONTEND

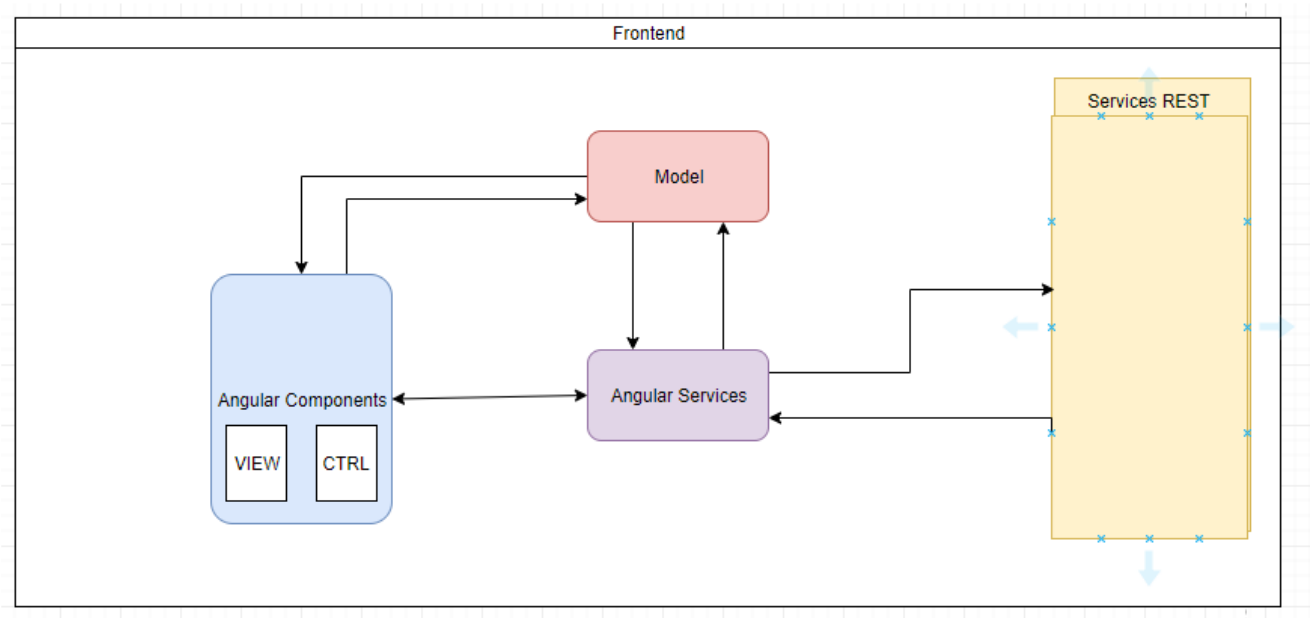
## Arquitectura a nivel Backend




Se propone una arquitectura derivada de MVC, en la cual se tiene la siguiente distribución:

- La capa Controller maneja la conexión con el cliente, recibiendo las peticiones que son enviadas para recuperar los datos. Esto mediante la implementación de interfaces.
- La capa Business procesa las validaciones de los datos que se enviaron a través de la petición, y envía o recibe datos de la capa Persistence
- La capa Persistence tiene dos elementos importantes:
    - Los archivos DAO los cuales son interfaces implementadas con las que se realizan las consultas a la base de datos, haciendo uso de la información recibida y las entidades.
    - El paquete Entity maneja todo el mapeo ORM que se realiza con las clases de la aplicación y las tablas en la base de datos.
- Simultáneamente se manejan dos capas de forma transversal:
    - El paquete DTO se encarga de todo el paso de información a lo largo de las capas, de esta manera sólo se utilizan objetos de la entidad en la capa DAO.
    - El paquete Util contiene todo lo referente a seguridad y configuración. En él se manejan las validaciones, las constantes, las excepciones y demás elementos necesarios para tener una aplicación controlada.
- Por último, para la base de datos, se utilizó un motor SQL mediante la implementación de PostgreSQL. Para un mejor despliegue y facilidad tanto en desarrollo como en producción, se utiliza AWS para almacenar la base de datos, mediante el servicio RDS. Así se tiene la base de datos en la nube, disponible en todo momento.

## Arquitectura a nivel Frontend (Angular 7)




● Modules:

se usan para exportar funcionalidades propias hacia otros módulos para ser utilizados.

● Templates, Directives and Data Binding:

combina HTML con Angular Markup y modificar elementos HTML antes de mostrarlos. Las directivas de plantilla proporcionan lógica de programa, y el marcado vinculante conecta los datos de su aplicación y el DOM. ***Event Binding*** se usa para enlazar eventos a su aplicación y responder a la entrada del usuario en el entorno de destino actualizando los datos de su aplicación. ***Data Binding*** se utiliza para pasar datos de la clase de componente y le permite interpolar valores que se calculan a partir de los datos de su aplicación en el HTML.

● Services and dependency injection:

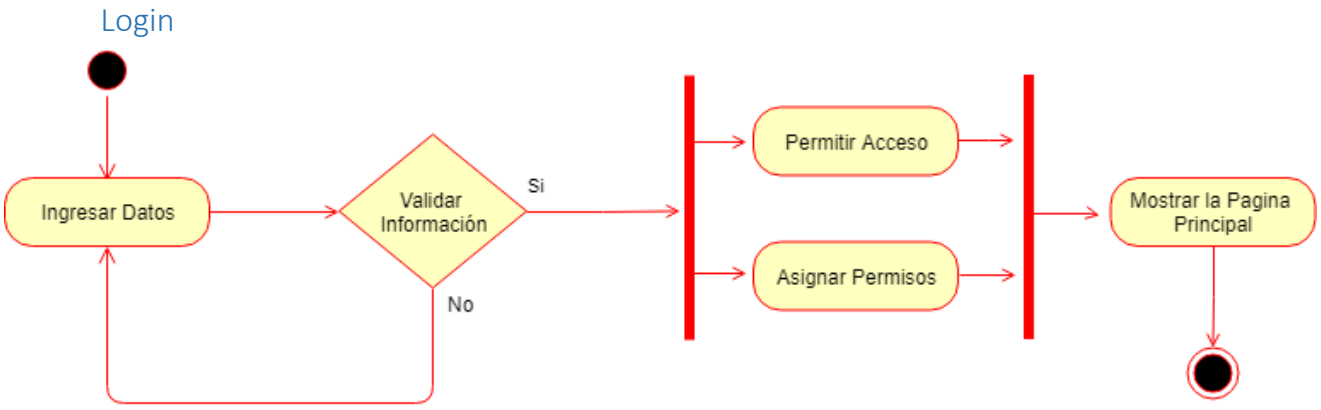
***Services*** son clases que se crean para datos o lógica que no tiene una vista específica y se desea compartir con otros componentes.

● Roting:

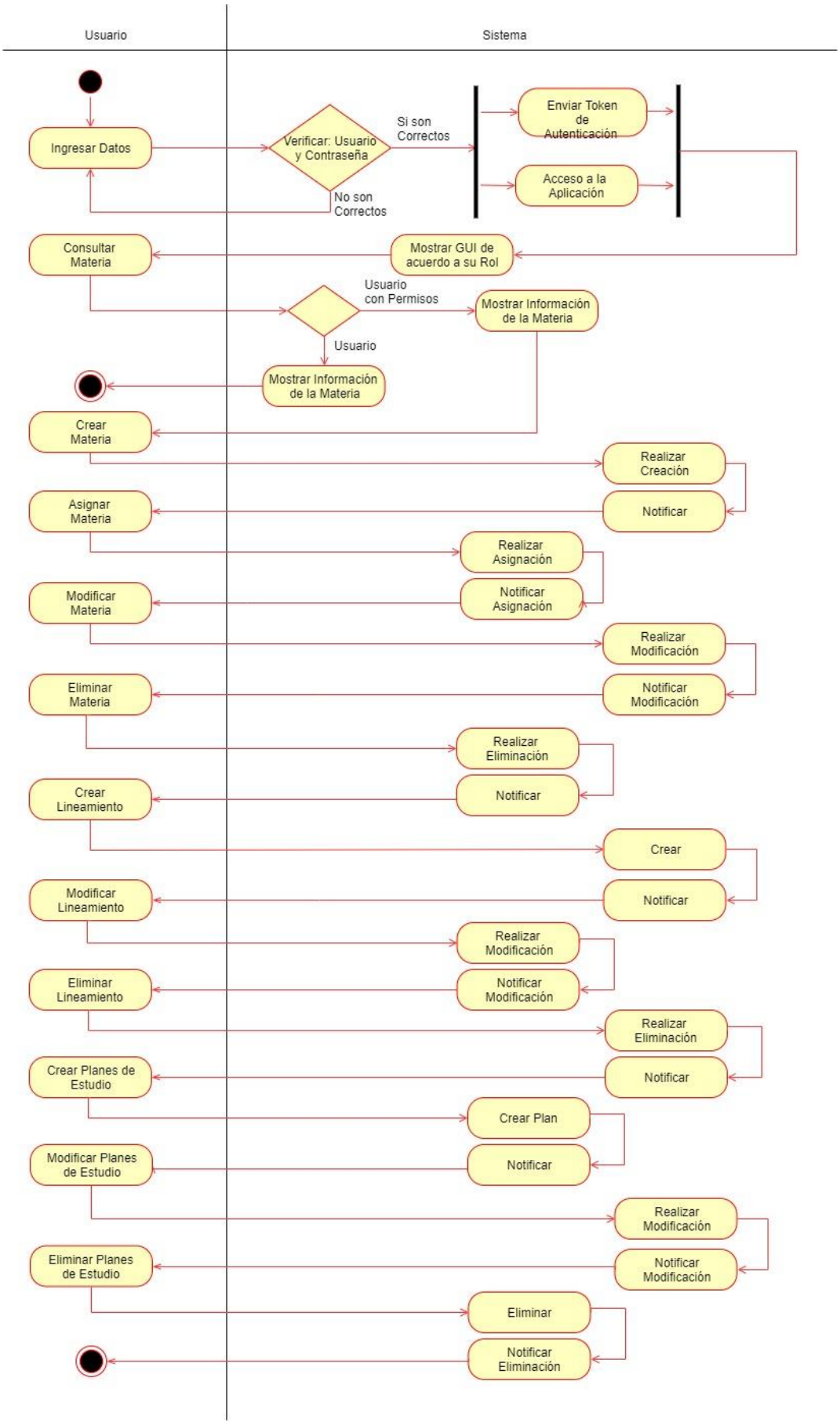
es un servicio de angular que proporciona la navegación de la aplicación por url

# Diagramas Equipo de Arquitectura y Diseño

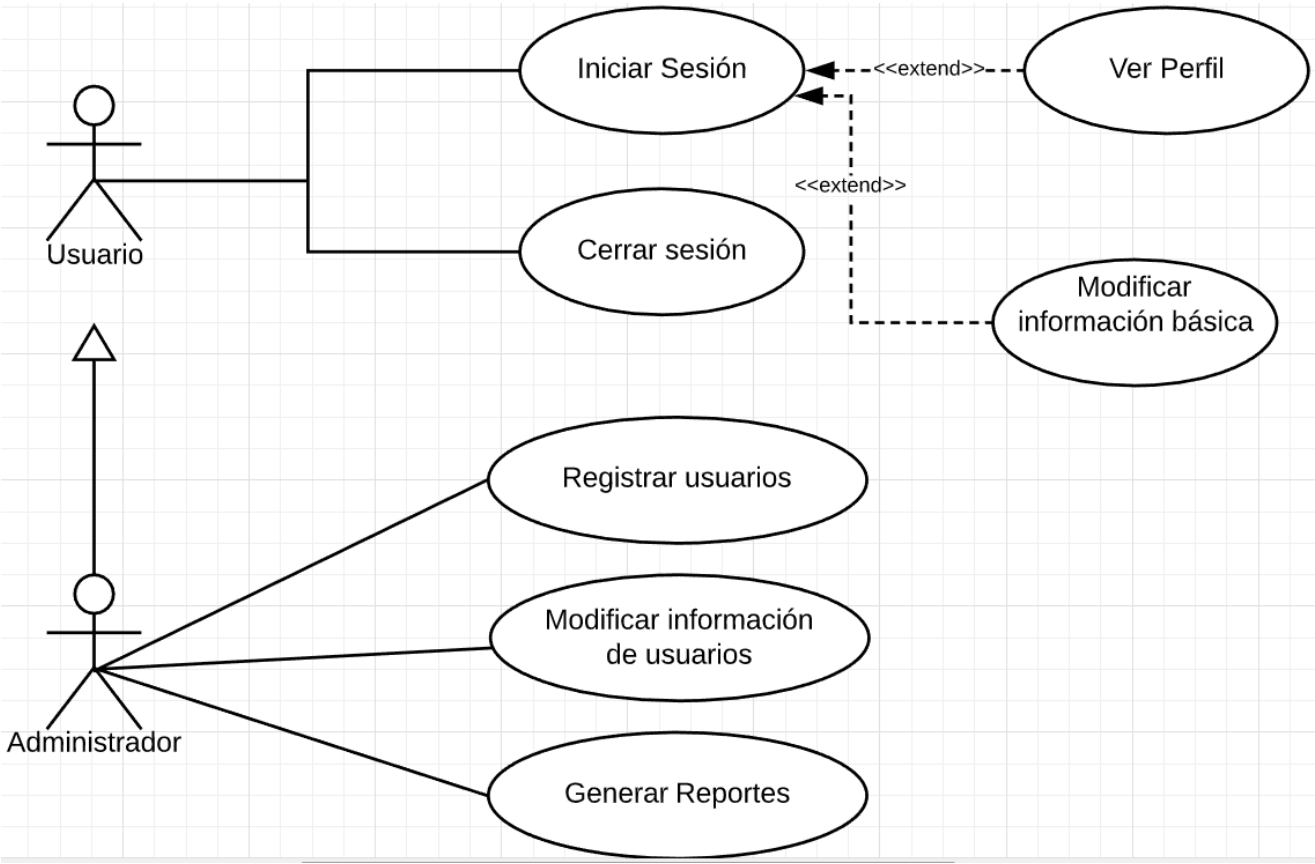
## Diagrama de Actividades:



## MicroCurriculo

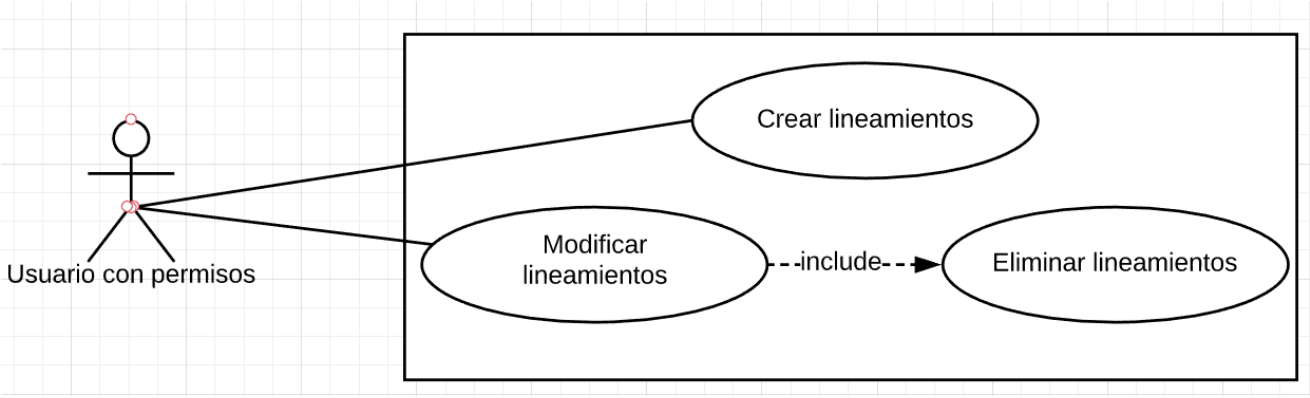


## Diagrama de casos de uso:




| | | |
|---|---|---|
| **CU-001** | INICIO DE SESION Y OPERACIONES USUARIO ADMINISTRADOR | |
| **Versión** | Sprint2 - 08-10-2019 | |
| **Autores** | Alejandro Marin | |
| **Descripción** | El sistema deberá tener un formulario de inicio de sesión con dos roles usuario y administradores tanto los usuarios como los administradores pueden ver su perfil y modificar la información básica, pero los administradores podrán hacer tres operaciones más registrar usuario que pueden ser otros administradores o usuarios, también deberá modificar toda información de los usuario y generar reportes. | |
| **Precondición** | | |
| **Secuencia** | **Paso** | **Acción** |
| **Normal** | 1 | El usuario o administrador, entra al formulario de inicio de sesión ingresa su nombre de usuario, contraseña y da clic en iniciar sesión. |

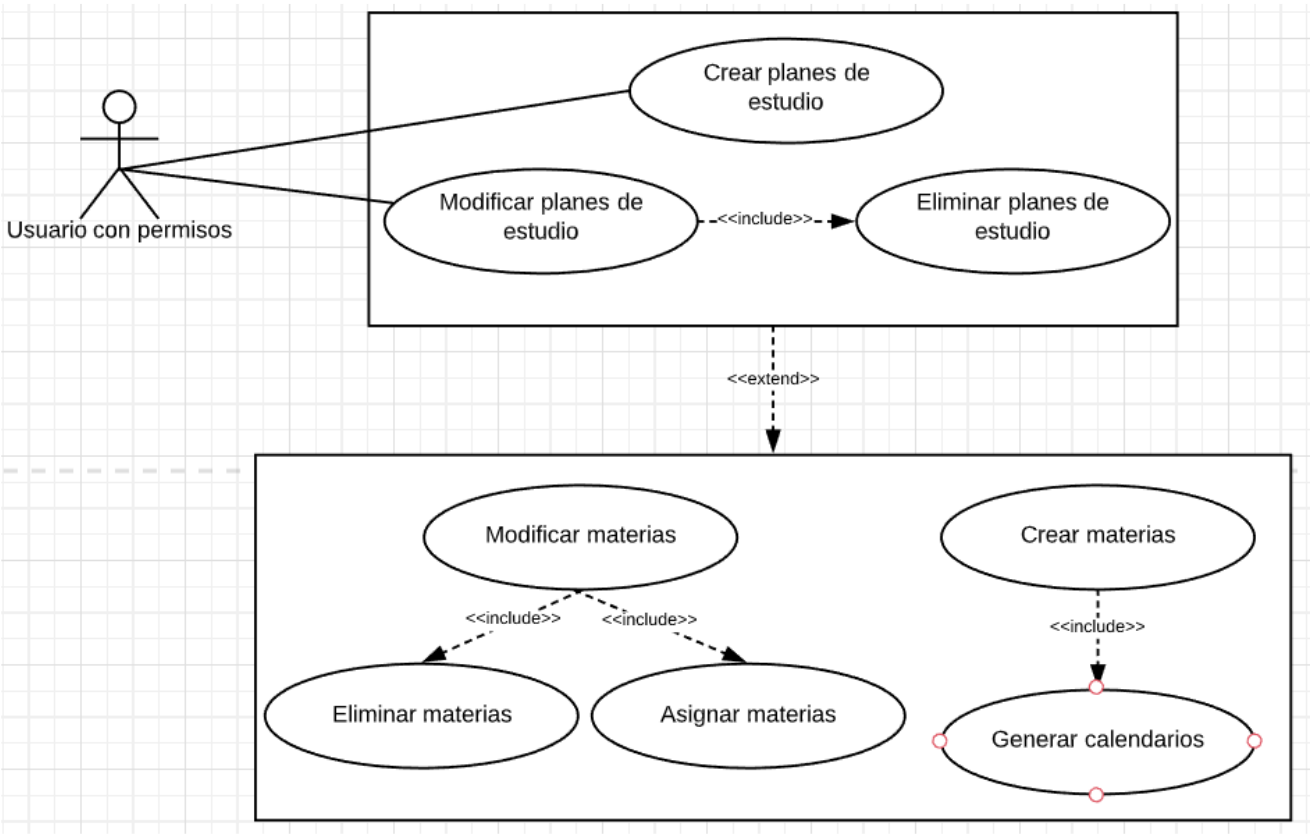
| | | |
|---|---|---|
| | 2 | Si la información es correcta, el sistema redimirá al usuario o administrador a la pantalla principal mostrando el menú de operaciones que puede realizar según el roll. |
| | 3 | Si la información es invalida, el sistema mostrara una alerta informando que el nombre de usuario o contraseña es invalido. |
| | 4 | Si el usuario es administrador muestra las opciones registra usuarios, modificar información usuarios, generar reportes. |
| | 5 | Si el usuario no es administrador solo debe mostrar ver perfil y modificar información básica. |
| **Excepciones** | **Paso** | **Acción** |
| | 1 | |
| | 2 | |
| | 3 | |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | 1 | 30 segundos |
| | 2 | 5 segundos |
| | 3 | 5 segundos |
| | 4 | 2 segundos |
| | 5 | 2 segundos |
| **Frecuencia esperada** | 1000 usuarios / hora | |
| **Importancia** | vital | |
| **Urgencia** | hay presión | |
| **Comentarios** | | |




| | | |
|---|---|---|
| **CU-002** | OPERACIONES ADMINISTRADOR CRUD LINEAMIENTO | |
| **Versión** | Sprint2 - 08-10-2019 | |
| **Autores** | Sebastián Peláez | |
| **Descripción** | El sistema deberá soportar crear, modificar y eliminar lineamientos para el micro-currículo siempre y cuando el usuario tenga permisos de administrador. | |
| **Precondición** | | |
| **Secuencia**<br>**Normal** | **Paso** | **Acción** |
| | 1 | El administrador, entra al formulario de creación de lineamientos y diligenciar la información del mismo |
| | 2 | Si la información es correcta, el sistema debe guardar esta información en base de datos y mostrar un mensaje de que fue creado con éxito. |
| | 3 | Si la información la información es invalida, el sistema mostrara una alerta informando que campo está mal diligenciada |
| | 4 | El administrador, entra al formulario de visualización de los lineamientos. |
| | 5 | Si el sistema encuentra lineamientos debería listarlos con sus respectivos de botones de edición y eliminación. |
| | 6 | El usuario selecciona el lineamiento y le da clic en modificar. |
| | 7 | El sistema debe mostrar la información del lineamiento seleccionado que puede ser modificada. |

| | | |
|---|---|---|
| | 8 | El usuario debe modificar la información del lineamiento |
| | 9 | Si la información es correcta, el sistema debe actualizar esta información en base de datos y mostrar un mensaje de que fue modificado con éxito. |
| | 10 | Si la información la información es invalida, el sistema mostrara una alerta informando que campo está mal diligenciada |
| | 11 | El usuario selecciona el lineamiento y hace clic en eliminar |
| | 12 | El sistema debe preguntar si desea eliminar el lineamiento |
| | 13 | Si el usuario hace clic en SI |
| | 14 | El sistema debe ocultar la alerta y eliminar el lineamiento y mostrar un mensaje de confirmación que fue eliminado |
| | 15 | Si el usuario hace clic en NO |
| | 16 | El sistema debe ocultar la alerta y no borrar el lineamiento. |
| **Excepciones** | **Paso** | **Acción** |
| | 1 | |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | 1 | 2 minutos |
| | 2 | 5 segundos |
| | 3 | 5 segundos |
| | 4 | 2 segundos |
| | 5 | 5 segundos |
| | 6 | 3 segundos |
| | 7 | 2 segundos |
| | 8 | 2 minutos |
| | 9 | 3 segundos |
| | 10 | 3 segundos |
| | 11 | 2 segundos |

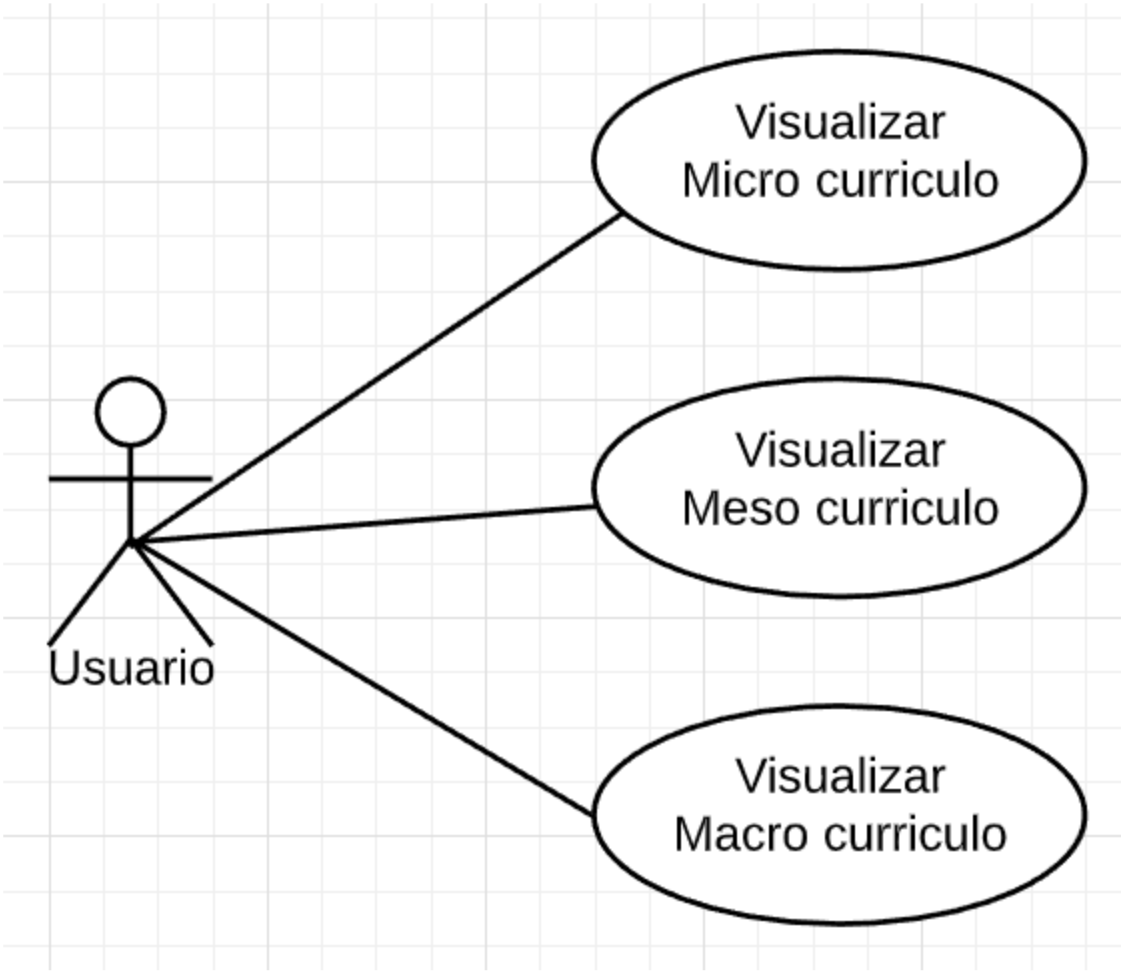
| | | |
|---|---|---|
| | 12 | 3 segundos |
| | 13 | 2 segundos |
| | 14 | 5 segundos |
| | 15 | 2 segundos |
| | 16 | 0 segundos |
| **Frecuencia esperada** | 1000 usuarios / hora | |
| **Importancia** | vital | |
| **Urgencia** | hay presión | |
| **Comentarios** | | |




| | |
|---|---|
| **CU-003** | OPERACIONES ADMINISTRADOR CRUD PLANES DE ESTUDIO |
| **Versión** | Sprint2 - 08-10-2019 |
| **Autores** | Deyber Sepulveda |

| | | |
|---|---|---|
| **Descripción** | El sistema deberá soportar crear, modificar y eliminar planes de estudio siempre y cuando el usuario tenga permisos de administrador. | |
| **Precondición** | | |
| **Secuencia**<br><br>**Normal** | **Paso** | **Acción** |
| | 1 | El administrador, entra al formulario de creación de planes de estudio y diligenciar la información del mismo |
| | 2 | Si la información es correcta, el sistema debe guardar esta información en base de datos y mostrar un mensaje de que fue creado con éxito. |
| | 3 | Si la información la información es invalida, el sistema mostrara una alerta informando que campo está mal diligenciada |
| | 4 | El administrador, entra al formulario de visualización de los planes de estudio. |
| | 5 | Si el sistema encuentra planes de estudio debería listarlos con sus respectivos de botones de edición y eliminación. |
| | 6 | El usuario selecciona el plan de estudio y le da clic en modificar. |
| | 7 | El sistema debe mostrar la información del plan de estudio seleccionado que puede ser modificada. |
| | 8 | El usuario debe modificar la información del plan de estudio |
| | 9 | Si la información es correcta, el sistema debe actualizar esta información en base de datos y mostrar un mensaje de que fue modificado con éxito. |
| | 10 | Si la información la información es invalida, el sistema mostrara una alerta informando que campo está mal diligenciada |
| | 11 | El usuario selecciona el plan de estudio y hace clic en eliminar |
| | 12 | El sistema debe preguntar si desea eliminar el plan de estudio |
| | 13 | Si el usuario hace clic en SI |

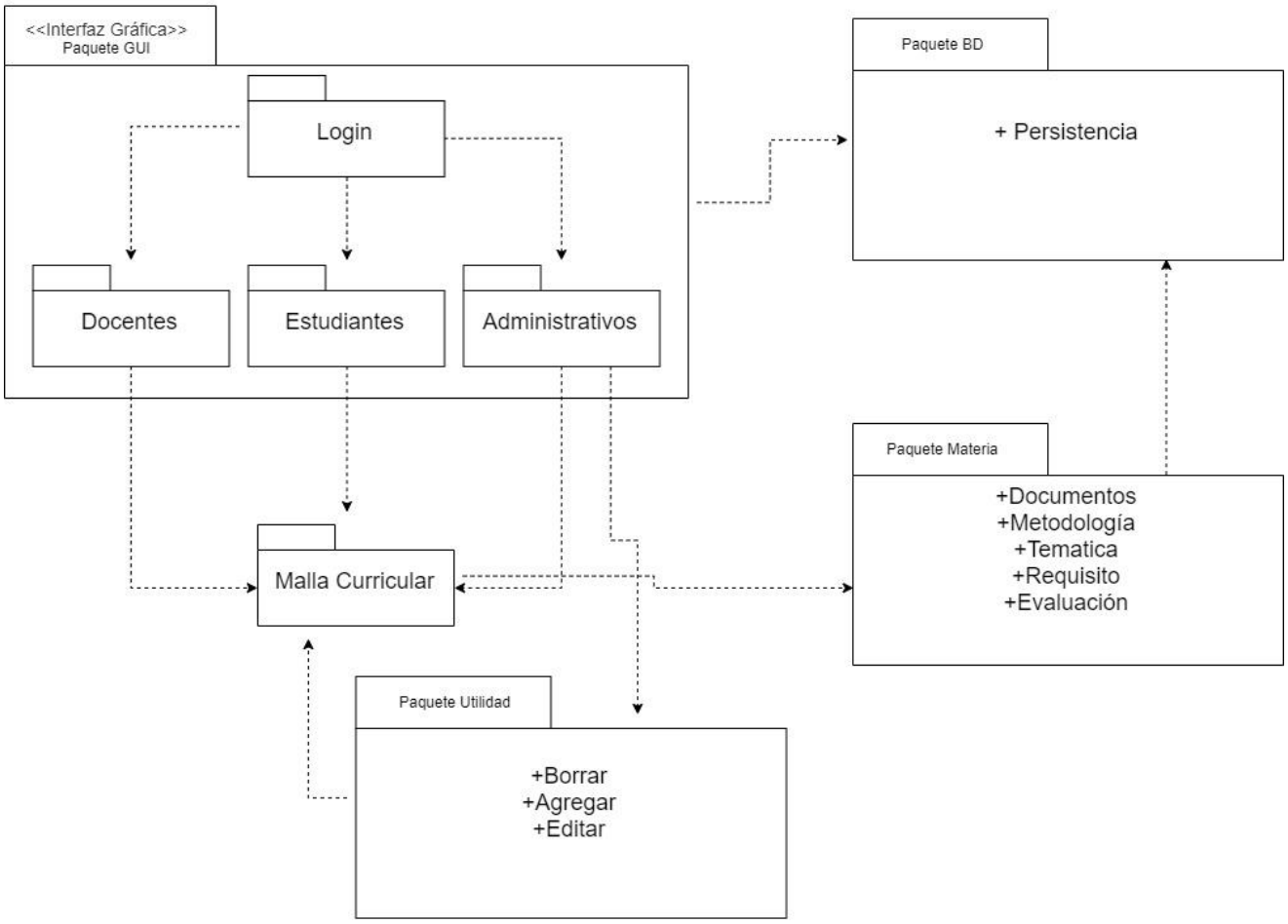
| | | |
|---|---|---|
| | 14 | El sistema debe ocultar la alerta y eliminar el plan de estudio y mostrar un mensaje de confirmación que fue eliminado |
| | 15 | Si el usuario hace clic en NO |
| | 16 | El sistema debe ocultar la alerta y no borrar el plan de estudio. |
| **Excepciones** | **Paso** | **Acción** |
| | 1 | |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | 1 | 2 minutos |
| | 2 | 5 segundos |
| | 3 | 5 segundos |
| | 4 | 2 segundos |
| | 5 | 5 segundos |
| | 6 | 3 segundos |
| | 7 | 2 segundos |
| | 8 | 2 minutos |
| | 9 | 3 segundos |
| | 10 | 3 segundos |
| | 11 | 2 segundos |
| | 12 | 3 segundos |
| | 13 | 2 segundos |
| | 14 | 5 segundos |
| | 15 | 2 segundos |
| | 16 | 0 segundos |
| **Frecuencia esperada** | 1000 usuarios / hora | |
| **Importancia** | vital | |
| **Urgencia** | hay presión | |
| **Comentarios** | | |



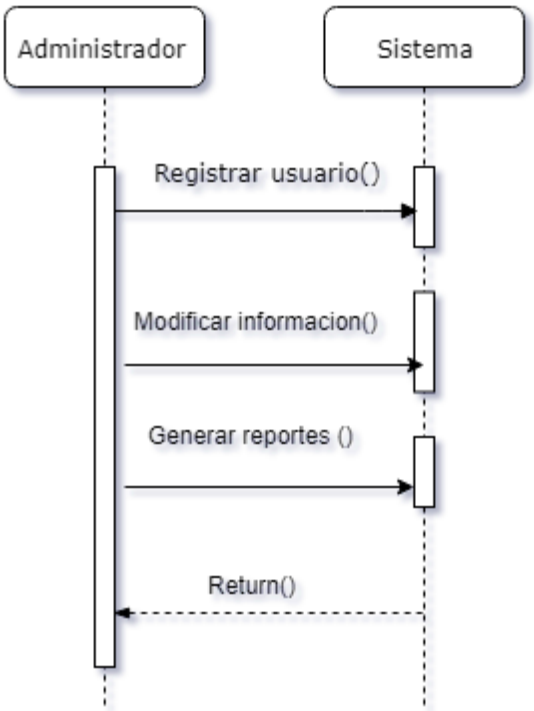

| | | |
|---|---|---|
| **CU-004** | VISUALIZACION DEL MICRO MACRO Y MESO CURRICULO | |
| **Versión** | Sprint2 - 08-10-2019 | |
| **Autores** | Raúl Arcila | |
| **Descripción** | El sistema deberá mostrarle al usuario el micro, macro y meso currículo | |
| **Precondición** | | |
| **Secuencia** | **Paso** | **Acción** |
| **Normal** | 1 | Cuando el usuario hace clic en mostrar los micro currículos. |
| | 2 | El sistema debe mostrar los micro currículos correspondiente a su programa a académico. |
| | 3 | Cuando el usuario hace clic en ver algún micro currículo. |
| | 4 | El sistema deberá mostrarle la información asociada a es micro currículo. |

| | | |
|---|---|---|
| | 5 | Cuando el usuario hace clic en mostrar el macro currículo |
| | 6 | El sistema debe mostrar el macro currículo correspondiente a su programa a académico. |
| | 7 | Cuando el usuario hace clic en mostrar el mezo currículo |
| | 8 | El sistema debe mostrar el mezo currículo correspondiente a su programa a académico. |
| **Excepciones** | **Paso** | **Acción** |
| | 1 | |
| **Rendimiento** | **Paso** | **Cota de tiempo** |
| | 1 | 2 segundos |
| | 2 | 5 segundos |
| | 3 | 2 segundos |
| | 4 | 5 segundos |
| | 5 | 2 segundos |
| | 6 | 5 segundos |
| | 7 | 2 segundos |
| | 8 | 5 segundos |
| **Frecuencia esperada** | 1000 usuarios / hora | |
| **Importancia** | vital | |
| **Urgencia** | hay presión | |
| **Comentarios** | | |

## Diagrama de paquetes:



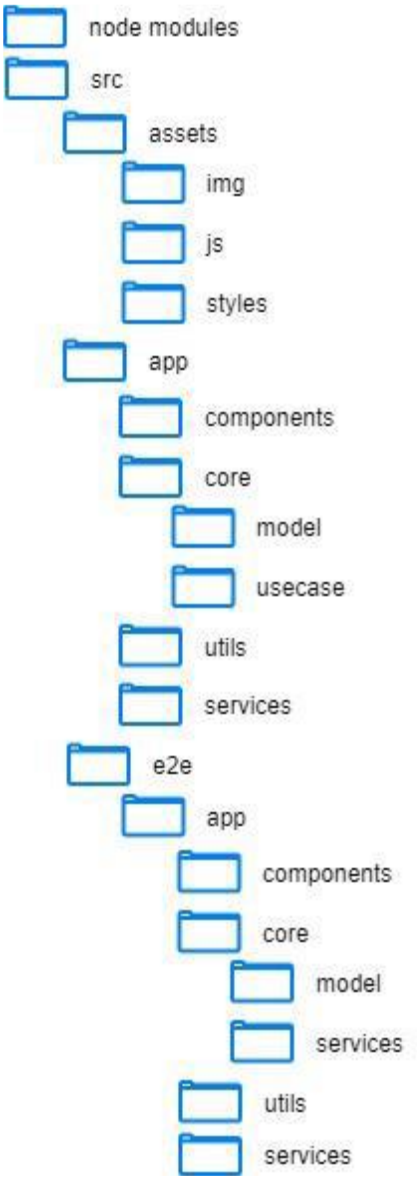
## Diagrama de secuencia:



# ESTRUCTURACIÓN DE LA APLICACIÓN

**Sprint 1**

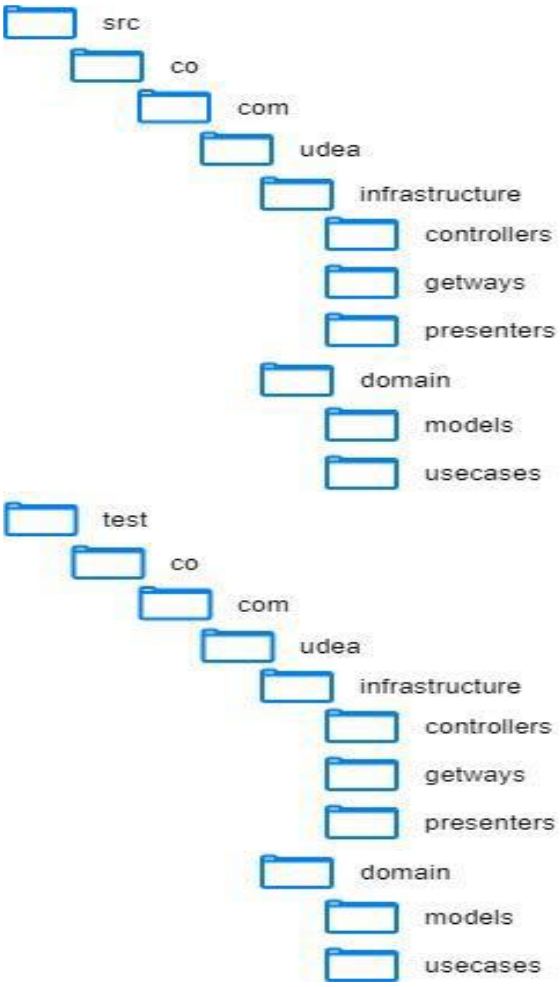
## Frontend




**Node modules:** directorio en el cual se guardan las librerías de nodejs, como las de angular y las de terceros.
**Src:** directorio en el cual van todos los recursos de la aplicación y también la aplicación misma con sus respectivas pruebas.

**Assets:** directorio donde se guardan los recursos de la aplicación, como imágenes, estilos genéricos y scripts genéricos.
**Img:** directorio donde se encuentran todas las imágenes usadas en la aplicación, tanto iconografía en svg como en formato de imagen.
**Js:** directorio en el cual se consignan las librerías Js que no están en hechas para angular, pero se quieren usar en la aplicación.
**Styles:** directorio en el cual se consignan los estilos de ccs de estas librerías o styles genéricos para la aplicación.
**App:** directorio donde se encuentran todos los scripts de la aplicación.
**Components:** directorio para las páginas y componentes más pequeños debido a que angular está orientado a componentes.
**Core:** directorio en el cual va consignada toda la lógica de aplicación y el modelo.
**Model**: directorio en el cual van las entidades de la aplicación.
**Usecase:** directorio en el cual va la lógica de negocio independiente a la lógica de la capa de presentación.
**Utils:** directorio en el cual van los validadores, o utilidades para la aplicación como mapeo de objetos a otros, etc.
**Services:** directorio en el cual van todos los servicios o providers de angular para traer la información o llevar la información al Backend.
**E2e:** directorio en el cual van todas las pruebas de nuestra aplicación con Jasmin y Karma, la estructura es igual a la estructura de la aplicación para que sea más comprensible encontrar el test de los diferentes directivas, servicios o controladores de nuestra aplicación.

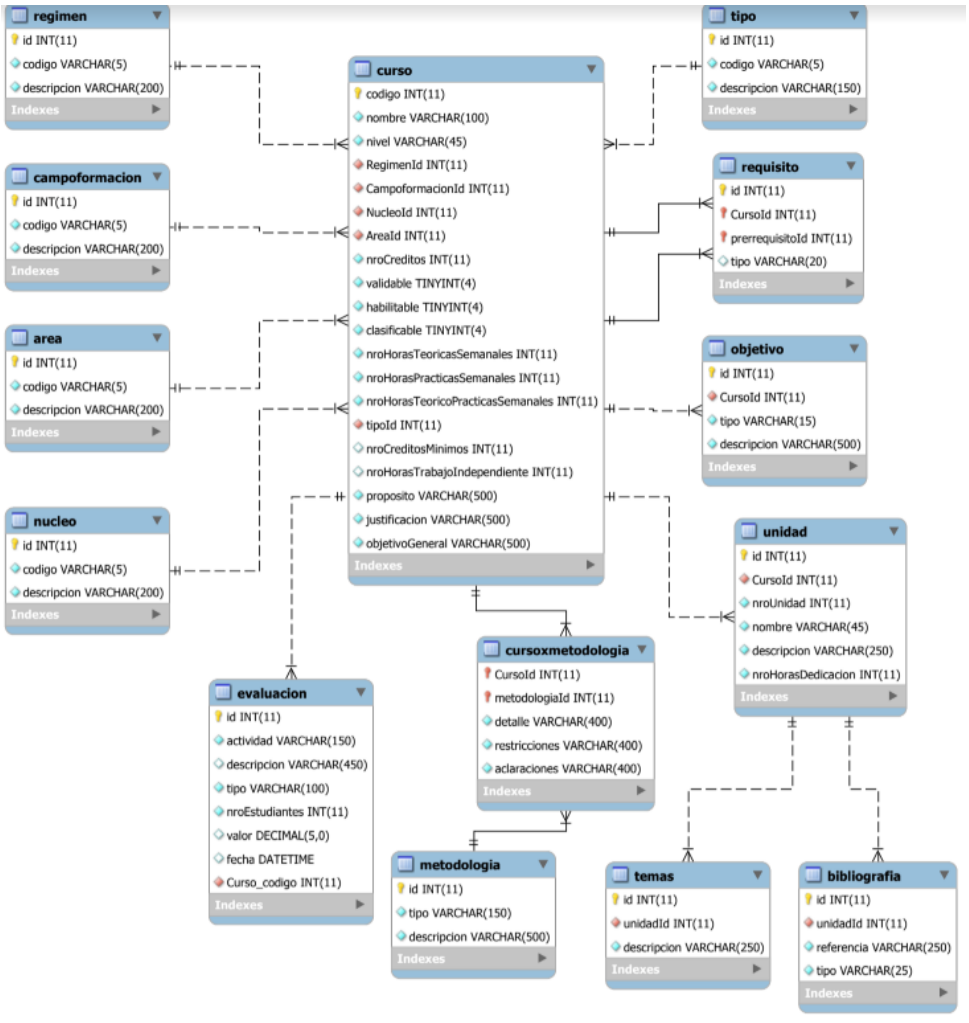
## Backend



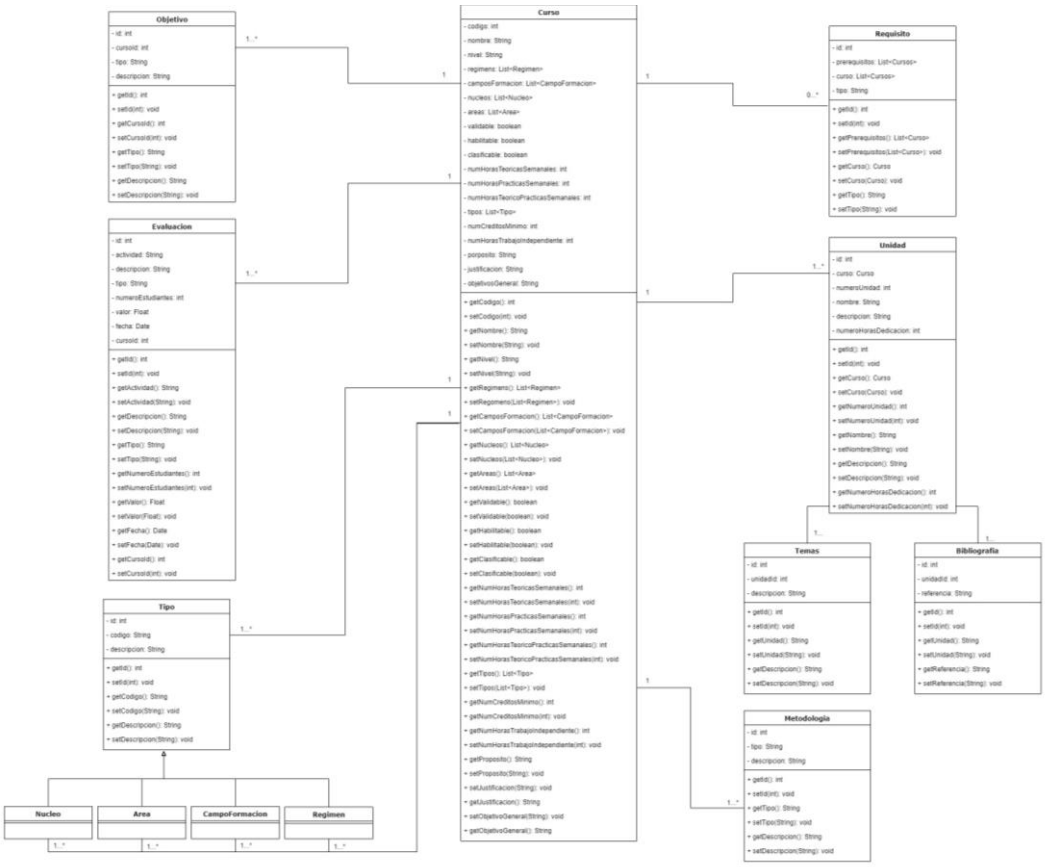
**src:** directorio en el cual se encuentra todo el código fuente de la aplicación.

**test:** directorio en el cual se encuentran los test de cada una de las clases de java de la aplicación.
**co.com. udea.infrastructure:** en este paquete van las conexiones con Apis externas, servicios rest, la capa de presentación, etc.
**co.com.udea.infrastructure.controllers:** paquete en el cual van los REST y/o SOAP expuestos por nuestra aplicación, también la conexión con dispositivos externos.
**co.com.udea.infrastructure.getways:** paquete en el cual va la implementación de los repositorios de base de datos y servicios externos.
**co.com.udea.infrastructure.presenters:** paquete en el cual va la interfaz de usuario o mensajes del backend.
**co.com.udea.domain:** paquete en el cual se encuentra el core de la aplicación y las interfaces para los repositorios y/o servicios externos.
**co.com.udea.domain.model:** paquete en el cual se encuentran las entidades de la aplicación, no necesariamente deben ser las mismas que usan los repositorios.
**co.com.udea.domain.usecase:** paquete en el cual va toda la lógica del negocio.

## DIAGRAMA ENTIDAD RELACIÓN



## DIAGRAMA DE CLASES




Para visualizar mejor el diagrama se recomienda consultar el siguiente link. shorturl.at/nBMVX

## PATRONES DE DISEÑO

### Front-End

Para la parte visual del Front todo lo que tiene que ver con el diseño manejaremos el patrón de diseño “Mostly Fluid” que lo que nos propone es que en pantallas grandes todo va agrupado en un contenedor que va centrado en la página con un tamaño de ancho ya fijado, y al reducirse a tamaños pequeños, como Smartphone todo pasa a ser en una única columna y lo que iba en diferentes filas que en distintos bloques dentro de la columna, y a medida que crezca los bloques se agrupan hasta ir ocupando toda la pantalla. Este patrón es implementado en HTML y CSS utilizando FlexBox. En la parte lógica del front-end será manejada en Angular 7, ya definido el método de trabajo de este, manejaremos lo que es el Patrón MVC, que este ya lo trae “implementado” que trabaja con un método muy similar, manejando la Vista con los componentes de Angular, el Modelo, y la conexión con Angular Services, lo que se traspola a un un MVC.

**¿Que es MVC?**
Modelo Vista Controlador (MVC) es patrón de diseño que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Se trata de un modelo muy maduro y que ha demostrado su validez a lo largo de los años en todo tipo de aplicaciones, y sobre multitud de lenguajes y plataformas de desarrollo.

● El **Modelo** que contiene una representación de los datos que maneja el sistema, su lógica de negocio, y sus mecanismos de persistencia.

● La Vista**,** o interfaz de usuario, que compone la información que se envía al cliente y los mecanismos interacción con éste.
● El Controlador**,** que actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

## Bibliografía

Azaustre, C. (noviembre 4, 2015). Carlos Azaustre: Los 5 patrones del responsiva designó. Recuperado de: https://carlosazaustre.es/los-5-patrones-del-responsive-design/

Universidad de Alicante. (septiembre 10, 2019). Universidad de Alicante: UA: Servicio de informática ASP.NET MVC 3 Framework: Modelo vista controlador (MVC). Recuperado de: https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html