CONSTRUCCIÓN DE UNA PLATAFORMA WEB PARA LA GESTIÓN DE LOS PLANES DE ESTUDIO Y SUS CONTENIDOS PARA EL PROGRAMA DE INGENIERÍA DE SISTEMAS DE LA FACULTAD DE INGENIERÍA DE LA UNIVERSIDAD CATÓLICA DE ORIENTE.

JUAN FERNANDO RESTREPO

JOHN ALEXANDER CASTAÑO HENAO

MAURICIO CARDONA JARAMILLO

UNIVERSIDAD CATÓLICA DE ORIENTE

FACULTAD DE INGENIERÍAS

PROGRAMA DE INGENIERÍA DE SISTEMAS

RIONEGRO ANTIOQUIA

2019

CONSTRUCCIÓN DE UNA PLATAFORMA WEB PARA LA GESTIÓN DE LOS PLANES DE ESTUDIO Y SUS CONTENIDOS PARA EL PROGRAMA DE INGENIERÍA DE SISTEMAS DE LA FACULTAD DE INGENIERÍA DE LA UNIVERSIDAD CATÓLICA DE ORIENTE.

JUAN FERNANDO RESTREPO

JOHN ALEXANDER CASTAÑO HENAO

MAURICIO CARDONA JARAMILLO

Trabajo de grado para optar por el título de:

Ingeniero de sistemas

Asesor

Luz Mery Ríos Alzate

UNIVERSIDAD CATÓLICA DE ORIENTE

FACULTAD DE INGENIERÍAS

PROGRAMA DE INGENIERÍA DE SISTEMAS

RIONEGRO ANTIOQUIA

2019

Nota de aceptación:

_____

_____

_____

_____

_____

Firma del presidente del jurado

_____

Firma del jurado

_____

Firma del jurado

Rionegro Agosto de 2019

CONTENIDO

		Pág.
ANTECEDENTES		5
1.	PLANTEAMIENTO DEL PROBLEMA	6
2.	JUSTIFICACIÓN	7
3.	OBJETIVOS	9
3.1.	General	9
3.2.	Específicos	9
4.	MARCO TEÓRICO	10
4.1.	Planes de Estudios	10
5.	DISEÑO METODOLÓGICO	16
6.	RESULTADOS	18
6.1.	Requisitos	18
6.2.	Diseño	27
6.2.1	*Base de datos*	*27*
6.3.	Back End	28
6.3.1	*Conceptos previos*	*28*
6.3.2	*Front End*	*35*
6.4.	Desarrollo	37
6.4.1	*Back End*	*37*
6.4.2	*Front End*	*38*
7.	CONCLUSIONES	56
REFERENCIAS BIBLIOGRÁFICAS		58

ANTECEDENTES

La calidad de gestión de los procesos educativos desde hace unas décadas ha sido influenciada altamente por las tendencias tecnológicas emergentes. Rico (2016) afirma que la gestión es exitosa en la medida que las instituciones de educación superior diseñen y lleven a cabo políticas de inversión y desarrollo orientadas al progreso y aporten al cumplimiento de los objetivos garantizando la innovación tecnológica.

Velásquez, Pérez y Ortega (2009), describen en su publicación denominada "Perfeccionamiento de los procesos universitarios. Un acercamiento a las herramientas de gestión "el acercamiento a algunas de las herramientas actuales para el perfeccionamiento de la gestión universitaria. Donde hacen uso de tendencias principales del desarrollo de la universidad contemporánea, así como de tecnologías de gestión, con énfasis en el cuadro de mando integral para evaluar el desempeño del proceso de perfeccionamiento; con el ánimo de que sirva de guía a directivos docentes y administrativos de las instituciones educativas.

Ugarte et al. , (2016) Realizaron un trabajo con el objetivo de contribuir a mejorar la calidad de la educación superior a través de la generación de información confiable, que permitiera identificar los factores que afectan la calidad de la docencia y su peso relativo en los distintos países, analizarlos en el marco de un modelo de interpretación común y apoyar el desarrollo de mecanismos para su utilización eficaz en los procesos de toma de decisiones relacionadas con la gestión de la docencia.

En la publicación "La gestión educativa: Hacia la optimización de la formación docente en la educación superior en Colombia ", Rico (2016) analiza la gestión educativa como un componente importante en la educación. Ofrecer formación de calidad indica una gestión precisa y acorde con la dinámica social y los avances de la ciencia y la tecnología.

1. PLANTEAMIENTO DEL PROBLEMA

Actualmente la coordinación del programa de Ingeniería de Sistemas de la Universidad Católica de Oriente presenta una oportunidad de mejora en los procesos relacionados, con la gestión de las diferentes versiones de los planes de estudio del programa y sus contenidos. El proceso que se realiza actualmente para consultar el plan de estudio, se hace mediante una imagen compuesta por cuadrículas que representan las materias correspondientes a cada semestre con sus respectivas características, como el número de créditos, componente de formación, horas teóricas, horas prácticas y su número de identificación. Adicional a esto, para conocer en detalle el plan de las diferentes asignaturas es necesario entrar a un link ubicado en cada cuadrícula donde se describe la materia, este link es el enlace a un documento en Word, allí se encuentra especificado el perfil de la materia. Es preciso destacar que este enlace es una forma muy poco confiable de ingresar al plan de asignatura puesto que la ruta donde está almacenado el documento en la máquina local puede quedar obsoleta al mover de ubicación el archivo.

El proceso actual impide tener una correcta trazabilidad de los diferentes planes de estudio con sus respectivos planes de asignatura y los cambios que han tenido en el tiempo y no es posible acceder a toda esta información de una manera más segura y centralizada.

El proceso actual, dadas las condiciones tecnológicas del momento y debido a que no se realiza de una manera automatizada; no es garante de cumplir con las necesidades la gestión requerida, por lo tanto, implica disponer de una herramienta de gestión que permita la visibilidad y trazabilidad, inmediata y constante del proceso con los impactos de los cambios que se realicen.

2. JUSTIFICACIÓN

Esta iniciativa pretende mejorar la forma de gestionar los planes de estudio con sus materias y sus respectivos micro currículos, que se hace desde la coordinación del programa de Ingeniería de Sistemas, impactando de forma positiva la labor que tiene el coordinador y generando una mayor productividad a la hora de hacer trazabilidad de los diversos cambios hechos a cada plan de asignatura de las diferentes materias del programa; esto se pretende lograr mediante una solución de software en la cual podamos plasmar los conocimientos adquiridos por el aprendizaje universitario y la experiencia laboral que día a día nos lleva a aportar soluciones empresariales de gran envergadura y que requieren decisiones responsables.

Con este proyecto estamos buscando un reto técnico que produciría una mejora en la calidad de la formación de los estudiantes porque se van a tener los contenidos de cada asignatura actualizados y evidenciar si cada docente está cumpliendo con los puntos y actividades propuestos en el planeador del curso cada semestre basados en el micro currículo vigente para la asignatura. Este aporte ayuda a la Institución Universitaria con el propósito de ser prestadora integral del servicio de aprendizaje y formadora de personas capacitadas. Es importante resaltar que mediante este proyecto se aumentará la visibilidad y la trazabilidad histórica de los cambios que se realicen sobre el programa, lo que hará de esta, una herramienta que permitirá, desde una visión crítica, informar a las direcciones para tomar decisiones sobre la calidad de la educación que se está dando y el nivel de actualización de los programas respecto al mercado, esto permitirá a la universidad formar estudiantes con conocimientos más actualizados, que estén a la vanguardia y volverlos más competitivos en el mercado.

La Universidad Católica de Oriente, se encuentra ubicada en una región agro-industrial con altos índices de crecimiento y desde el punto de vista formativo ha sido una entidad que se preocupa por el buen nivel académico de los habitantes del Oriente Antioqueño y el crecimiento

social de su región; por lo tanto, es de utilidad todo aquel mecanismo que le permita mejorar su labor social para brindar un buen servicio. La realización de esta aplicación impacta directamente la calidad en el programa de Ingeniería de Sistemas, ya que brinda una plataforma donde se puede hacer la gestión de todo el plan de estudio de forma centralizada, además, se evitará la pérdida de información debido a la deficiencia de trazabilidad con la que cuenta la coordinación de sistemas actualmente para enterarse qué cambios han tenido los planes de estudio y los micro currículos de cada asignatura en el tiempo.

3. OBJETIVOS

3.1. General

Construir una plataforma web para la gestión de los planes de estudio y sus contenidos para el programa de ingeniería de sistemas de la facultad de ingeniería de la Universidad Católica de Oriente.

3.2. Específicos

- Definir los requerimientos necesarios tomados de el coordinador del programa de sistemas para el diseño y la construcción del producto.
- Diseñar de manera detallada los modelos de arquitectura apropiados para la implementación del producto de software.
- Desarrollar mediante un lenguaje de programación la solución del problema.

4. MARCO TEÓRICO

4.1. Planes de Estudios

Según Rico (2016) La gestión educativa es entendida como un proceso organizado y orientado a la optimización de procesos y proyectos internos de las instituciones, con el objetivo de perfeccionar los procedimientos pedagógicos, directivos, comunitarios y administrativos que en ella se movilizan. Igualmente, Blanco & Quesada (s.f.) definen la gestión como un elemento vital para la organización y la calidad de desempeño de cualquier institución.

Teniendo en cuenta lo anterior, con un enfoque más específico al proceso de la planeación del contenido de un programa universitario, es pertinente destacar que al interior de las Instituciones educativas se desarrollan planes de estudio que ayudan a evidenciar los contenidos que se abordan en los cursos ofrecidos por la entidad educativa.

El Ministerio de educación nacional de Colombia (s.f) afirma que el plan de estudios es el esquema estructurado de las áreas obligatorias y fundamentales y de áreas optativas con sus respectivas asignaturas que forman parte del currículo de los establecimientos educativos. El plan de estudios debe contener al menos los siguientes aspectos:

- La intención e identificación de los contenidos, temas y problemas de cada área, señalando las correspondientes actividades pedagógicas.
- La distribución del tiempo y las secuencias del proceso educativo, señalando en qué grado y período lectivo se ejecutarán las diferentes actividades.
- Los logros, competencias y conocimientos que los educandos deben alcanzar y adquirir al finalizar cada uno de los períodos del año escolar, en cada área y grado, según hayan sido definidos en el proyecto educativo institucional-PEI- en el marco de las normas técnicas curriculares que expida el Ministerio de Educación Nacional. Igualmente incluirá los criterios y

los procedimientos para evaluar el aprendizaje, el rendimiento y el desarrollo de capacidades de los educandos.

- El diseño general de planes especiales de apoyo para estudiantes con dificultades en su proceso de aprendizaje.
- La metodología aplicable a cada una de las áreas, señalando el uso del material didáctico, textos escolares, laboratorios, ayudas audiovisuales, informática educativa o cualquier otro medio que oriente soporte la acción pedagógica.
- Indicadores de desempeño y metas de calidad que permitan llevar a cabo la autoevaluación institucional.

Los planes de estudio desarrollados por los establecimientos educativos son un ítem esencial que hace parte del proceso de gestión educativa. En la actualidad las tecnologías de la información juegan un papel importante en la realización de procesos de gestión porque facilitan el desarrollo de proyectos y actividades tales como la búsqueda, la selección, la organización, el almacenamiento, la recuperación y la visualización de la información.

De acuerdo con lo anterior, es posible desarrollar productos de software que solucionan necesidades específicas según se requiera; un ejemplo de esto, son aquellos que ayudan en la gestión de los procesos de cualquier empresa o entidad que preste algún servicio.

Un buen software de gestión educativa es aquel que se integra de manera adecuada con las actividades de la institución que presta el servicio y cubre las necesidades que surgen. Se encarga de la gestión diaria y continua de los diferentes escenarios y procesos, que sean requeridos. Las aplicaciones de gestión documental, son un tipo de software de gestión que tienen la finalidad de organizar, facilitar el acceso y la búsqueda de la documentación de la institución (Martín, 2018).

Para construir un producto de software es necesario el uso de diferentes herramientas, una de estas son los lenguajes de programación, que se pueden entender básicamente, como un sistema estructurado de comunicación, similar al humano, el cual permite hacer que los computadores ejecuten determinadas instrucciones.

Siendo Scala un lenguaje de programación que combina la orientación a objetos y el paradigma de programación funcional, es bastante útil su uso en aplicaciones concurrentes; es un lenguaje de alto nivel y su sistema de tipos estáticos ayudan a evitar bugs en aplicaciones complejas. Además, Scala posee una herramienta de construcción interactiva llamada SBT (Scala build tool) que permite compilar el código y gestionar el manejo de dependencias en un proyecto con este lenguaje.

Como es común en la programación, se pueden encontrar diversas librerías y frameworks que se pueden complementar con la variedad de lenguajes que existen en la actualidad, estas herramientas ayudan a desarrollar diferentes tareas de forma mas rapida. A Continuación se menciona las librerías más importantes utilizadas en el proyecto.

Akka Http es una librería enfocada en ofrecer un conjunto de herramientas para construir y consumir servicios basados en el protocolo HTTP. Si la intención de la aplicación es interactuar con el mundo externo exponiendo alguna API Akka Http es una opción viable (Mishra, 2017)

Es común que se necesite proteger ciertos recursos del sistema evitando que usuarios no autenticados realicen operaciones que deriven en algún cambio de estado de la aplicación, para esto hay diferentes estándares diseñados para ser usados con el protocolo http.

OAuth es un estándar abierto del protocolo de autorización que proporciona a las aplicaciones la capacidad de "acceso designado seguro". Este no comparte datos de contraseñas, sino que utiliza tokens de autorización para probar una identidad entre los consumidores y los proveedores de servicios. En otras palabras, es un protocolo de autenticación que permite aprobar

una aplicación que interactúa con otra en su nombre sin revelar ninguna contraseña de usuario (Sobers, 2018).

Otra librería que proporciona abstracciones para la programación funcional en el lenguaje de Scala teniendo como fundamento la teoría de categorías es Cats.

Cats contiene una amplia variedad de herramientas de programación funcional y permite a los desarrolladores escoger y usar la que más se acomoda a la aplicación. La mayoría de estas herramientas están entregadas en forma de type classes que pueden ser aplicadas a los tipos existentes de Scala (Welsh M., Gurnell D., 2016).

Type Classes es un patrón de programación funcional originado en Haskell, estas permiten extender librerías existentes con nueva funcionalidad sin usar la herencia tradicional y sin alterar la fuente de código original que provee la librería (Welsh M., Gurnell D., 2016).

En el ecosistema de Scala existen diferentes tipos a cargo de abstraer operaciones de entrada y salida en la aplicación. La mayoría de estos tipos están expuestos en librerías de alta calidad que evitan efectos no deseados dentro del programa.

Monix es una librería de alto desempeño para scala con la intención de componer programas asíncronos y basados en eventos. Task es un tipo de dato perteneciente a esta librería para controlar computaciones asíncronas y efectos secundarios evitando operaciones no-deterministas (Monix, s.f.)

En la mayoría de los casos cuando se está construyendo algún producto software se tiene la necesidad de utilizar algún recurso que permita hacer la integración con la base de datos usada en el proyecto.

Slick es una librería la cual provee acceso a bases de datos relacionales desde aplicaciones que usan Scala, está fuertemente basada en el paradigma de programación funcional. Esta herramienta tiene la promesa de reinventar la forma de acceder a las bases de datos relacionales

mediante operaciones regulares sobre colecciones, esto es familiar para cualquier desarrollador de scala, además tiene un fuerte énfasis en la seguridad de los tipos (Jenifer, 2016).

PostgreSQL es un sistema de administración de base de datos relacional de tipo empresarial, es especial porque no es solo una base de datos; es además una plataforma de aplicación. PostgreSQL es rápida, en los benchmarks iguala o supera el desempeño de muchas otras bases de datos de código abierto o de propiedad(Obe R., Hsu L., 2017).

Por otra parte, Docker es una herramienta muy útil para tener un ambiente acondicionado en el cual se pueden ejecutar diferentes tipos de aplicaciones. Desde un enfoque distribuido, cada contenedor representaría un nodo el cual es una aplicación en sí.

Docker permite ingresar en un contenedor (“una caja”, algo autocontenido, cerrado) todas aquellas cosas que la aplicación necesita para ser ejecutada (Scala, Sbt, Akka) y la propia aplicación, así, este contenedor puede ser trasladado a cualquier máquina que tenga instalado Docker y ejecutar la aplicación (García, 2015).

Kubernetes es un sistema para automatizar despliegues, escalar, y administrar aplicaciones contenidas. Un ejemplo de estas, son aplicaciones que viven dentro de un contenedor de Docker.

Al tener un cluster de grupos de hosts que ejecutan contenedores de Linux, Kubernetes ayuda a administrar con facilidad y eficacia estos clústeres (RedHat, s.f.).

Según Roche (2013) el software de control de versiones más popular en la actualidad se llama Git. Es un sistema distribuido cuyo objetivo es mantener una gran cantidad de código eficientemente desarrollado por diferentes programadores. Hay dos características de Git que ayudan a entender esta definición de manera simple:

- Git se diferencia con respecto a otros sistemas de control de versiones, en la forma que maneja los cambios en los ficheros, mientras otras aplicaciones similares almacenan los

archivos originales, conservando una lista de los cambios realizados en cada versión; Git guarda una “foto” (snapshot) del estado de cada archivo en un momento concreto.

- Git almacena una copia completa del repositorio en la máquina de forma local, incluido el historial de cambios. Esto implica que muchas de las operaciones realizadas sobre el código fuente no tienen lugar en la red, permitiendo que la velocidad de proceso dependa únicamente en los recursos locales.

5. DISEÑO METODOLÓGICO

Actualmente la coordinación del programa de Ingeniería de Sistemas de la Universidad Católica de Oriente, en el proceso de gestión de los planes de estudio, presenta varias oportunidades de mejora; como solución a esta necesidad, se va a desarrollar una aplicación web, en donde se permita ver con claridad y de forma centralizada, los diferentes planes de estudio con sus respectivos INPs, además será posible acceder individualmente a cada plan de estudio para visualizar todas las asignaturas correspondientes a estos, estructuradas por su respectivo semestre académico.

En detalle la aplicación web contará con una pantalla de inicio de sesión, en la cual el usuario podrá ingresar haciendo uso del servicio de oAuth de Google; en caso de que la clave y la contraseña sean correctos, se abre una pantalla donde el usuario puede empezar a realizar el proceso de gestión; en este punto el usuario cuenta con dos opciones, visualizar programa y agregar programa, de esta manera, la aplicación permitirá al usuario, agregar nuevos programas con sus respectivos INPs, planes de estudio y asignaturas; además de gestionar el programa que se encuentre configurado, en este caso el de ingeniería de Sistemas.

Cuando se elige el programa deseado, será posible ver una pantalla con sus INPs asociados, para que el usuario tenga la opción de elegir el INP que desea gestionar. Después de seleccionar un INP determinado se podrá ver todos los semestres que tiene este en vigencia. Al seleccionar uno de los semestres se desplegará una pantalla en la que se cargaran las asignaturas pertenecientes a este, cada curso contará con la opción de cargar archivos con la finalidad de subir su respectivo plan de asignatura; como puede existir más de un plan de asignatura para un único curso, va a haber un campo de texto en la parte posterior, donde se debe agregar una breve descripción del cambio que se hizo en el plan de asignatura que se esté subiendo con respecto al

anterior. Los archivos quedarán listados en esta pantalla para evidenciar la trazabilidad, con campos informativos como fecha de carga y descripción del cambio.

Para el Front-end de la aplicación se utilizó React, que permitirá tener una muy buena experiencia de usuario con altas propiedades de mantenibilidad y concurrencia de datos; por medio de esta, se hicieron las peticiones a la capa de servidor (Back-end), con una arquitectura de servicios REST.

En el Back-end se utilizó Scala, un lenguaje de programación de alto nivel, que, gracias a la combinación de programación orientada a objetos y programación funcional, permite realizar aplicaciones robustas con un alto índice de control de errores en tiempo de compilación; lo que garantiza aplicaciones de alta calidad y con un soporte de concurrencia de datos y peticiones robustas gracias al paradigma funcional y reactivo.

Como motor de base de datos se utilizó PostgreSQL, lo cual nos permitirá un dinamismo al momento de utilizar los datos según las necesidades de rendimiento e integridad estructural, utilizando cada tecnología de manera acertada en cuanto a las razones por la que fue creada y sus características de funcionamiento.

La herramienta de control de versiones que utilizamos fue GIT.

6. RESULTADOS

6.1. Requisitos

Historia de Usuario 1 Como administrador quiero poder loguearme a la plataforma de Gesiones UCO a traves de mi cuenta Gmail para tener trazabilidad de los planes de estudio de las asignaturas para el programa de Ingeniería de Sistemas	Criterios de aceptación: Al hacer click sobre el bóton de Login una ventana preguntará por las credenciales de mi cuenta Gmail. Si las credenciales son erroneas la API de Google no permitirá entra a la plataforma. La vida del token es de 1 hora, es decir, que pasado este tiempo las únicas funciones que com puedo hacer sobre la plataforma son consulta más no modificaciones.
Historia de Usuario 2 Como usuario deseo poder crear nuevos componentes de formación para poder ser asignados a las asignaturas.	Criterios de aceptación: Validar que los campos no esten vacios. No permitir su creación si el componente ya existia.

Historia de Usuario 3

Como usuario quiero poder ver los componentes de formación.

Historia de Usuario 4

Como usuario quiero poder actualizar componentes de formacion

Criterios de aceptación:

- Validar que los nuevos valores para los campos actualizados no sean vacíos.
- Validar que el componente de formación exista.

Historia de Usuario 5 Como usuario quiero poder agregar un programa.	Criterios de aceptación: Validar que los campos no esten vacios. Validar que el programa no existiera previamente Crear una carpeta en drive con el nombre del programa. Almacenar el programa en la base de datos.
Historia de Usuario 6 Como usuario quiero poder actualizar un programa.	Criterios de aceptación: Validar que los campos nuevos no esten vacios Validar que el programa existiera previamente Actualizar la carpeta en drive en caso de que haya habido un cambio en el nombre del programa. Actualizar el programa en la base de datos.

Historia de Usuario 7

Como usuario quiero obtener los programas existentes.

Historia de Usuario 8

Como usuario quiero eliminar un programa.

Criterios de aceptación:

Validar que el programa exista.

Eliminar el programa de la base de datos

Historia de Usuario 9

Como usuario quiero poder agregarle un plan de estudio a un programa.

Criterios de aceptación:

- Validar que el plan de estudio para dicho programa no exista previamente.
- Validar que el programa exista.
- Validar que los campos del plan de estudio sean correctos y no vacíos.
- Crear una nueva carpeta dentro del programa con el nuevo plan de estudio.
- Almacenar en la base de datos el nuevo plan de estudio.

Historia de Usuario 10

Como usuario quiero poder ver los planes de estudio

Historia de Usuario 11 Como usuario quiero poder eliminar un plan de estudio de un programa.	Criterios de aceptación: Validar que el plan de estudio exista. Eliminar de drive la carpeta creade del plan de estudio.
Historia de Usuario 12 Como usuario quiero poder agregar una asignatura a un plan de estudio de un programa.	Criterios de aceptación: Verificar que el programa y el plan de estudio existan. Validar que la asignatura no existiera previamente. Validar que los campos de la asignatura son correctos y no vacios. Crear carpeta de asignatura sobre el plan de estudio y el programa especificado en el endpoint. Sumar créditos totales del plan de estudio Almacenar asignatura en la base de datos. Actualizar plan de estudio con el nuevo valor de créditos

Historia de Usuario 13 Como usuario quiero poder asignar requisitos a una asignatura.	Criterios de aceptación: Verificar que la asignatura exista. Validar los campos del requisito. Dejar comentario de la asignación del requisito. Almacenar requisito en la base de datos.
Historia de Usuario 14 Como usuario quiero poder actualizar las condiciones(prerequisitos, corequisitos y requisitos de nivel) a una asignatura.	Criterios de aceptación: Verificar que la asignatura exista. Verificar campos nuevos. Actualizar en la base de datos.

Historia de Usuario 15

Como usuario quiero poder eliminar condiciones(prerequisitos, corequisitos y requisitos de nivel) a asignatura.

Criterios de aceptación:

Verificar que la asignatura exista.

Verificar que el requisito exista

Eliminar requisito de la base de datos para la asignatura.

Historia de Usuario 16

Como usuario quiero poder listas asignaturas y mostrarlas en una matriz dinámica.

Historia de Usuario 17

Como usuario quier tener visibilidad de la descripción de cambios sobre una asignatura basado en las fechas.

Historia de Usuario 18 Como usuario quiero poder eliminar asignatura.	Criterios de aceptación: Verificar que la asignatura exista. Eliminar carpeta de asignatura en drive. Eliminar asignatura de la base de datos.
Historia de Usuario 19 Como usuario quiero poder subir planes de estudio de asignaturas en las asignaturas.	Criterios de aceptación: Se selecciona el archivo que se desea subir y se ubica en la carpeta de la asignatura en google drive.

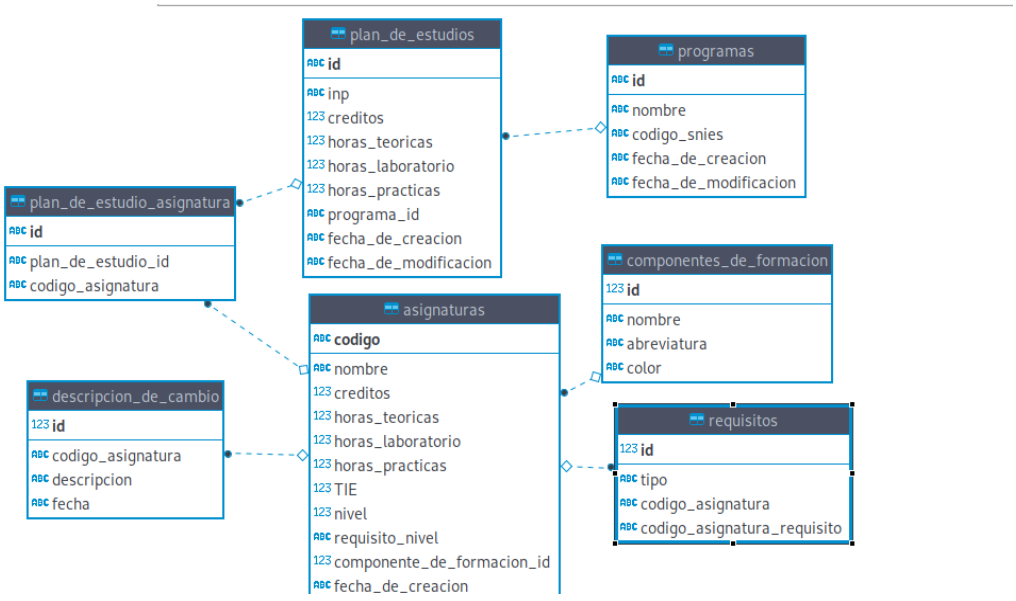
Historia de Usuario 20

Como usuario quiero poder visualizar los archivos que se han subido sobre una asignatura.

6.2. Diseño

6.2.1 Base de datos

Se usó PostgreSQL como motor de base de datos y el diagrama Entidad Relación es el siguiente:



Las razones por las cuales se eligió PostgreSQL como gestor de base datos son: software libre, modelado de datos (no solo es relacional sino también orientado a objetos lo cual brinda funcionalidades extra como tipos, funciones y operadores) y su integridad de datos (ACID;

Atómico, Íntegro, Aislado y Durable). ¿Por qué SQL y no no-SQL? La cantidad de datos que la plataforma va a manejar y las operaciones sobre la base de datos son bajas, como consecuencia, solo se necesita una instancia de esta lo cual la hace adecuada para el tipo de solución desarrollada.

El modelo está normalizado y refleja el negocio. Una tabla programas que contiene 0 o muchos planes de estudio, cada plan de estudio, a su vez, contiene 0 o muchas asignaturas, estas asignaturas pueden tener 0 o muchos requisitos(los cuales tienen un campo 'tipo' que identifica al requisito de tres formas: co-requisito, pre-requisito o requisito de nivel. Un co-requisito es una asignatura que debe ser vista a su vez en el mismo nivel con cierta asignatura, es decir, si la asignatura 'A' tiene como co-requisito a la asignatura 'B', estas dos materias deben ser vistas en el mismo nivel. Un pre-requisito es una asignatura que debió haber sido cursada previamente, es decir, si la asignatura 'A' tiene como requisito a la asignatura 'B', la asignatura 'B' debió haber sido cursada uno o más niveles anterior que la asignatura 'A'. Un requisito de nivel es una asignatura que debe cursarse en cierto nivel específico, es decir, si la asignatura 'A' únicamente se puede cursar en el nivel 3 o posterior). Descripción de cambios, contiene los comentarios de las modificaciones que las asignaturas han tenido desde el momento de su creación.

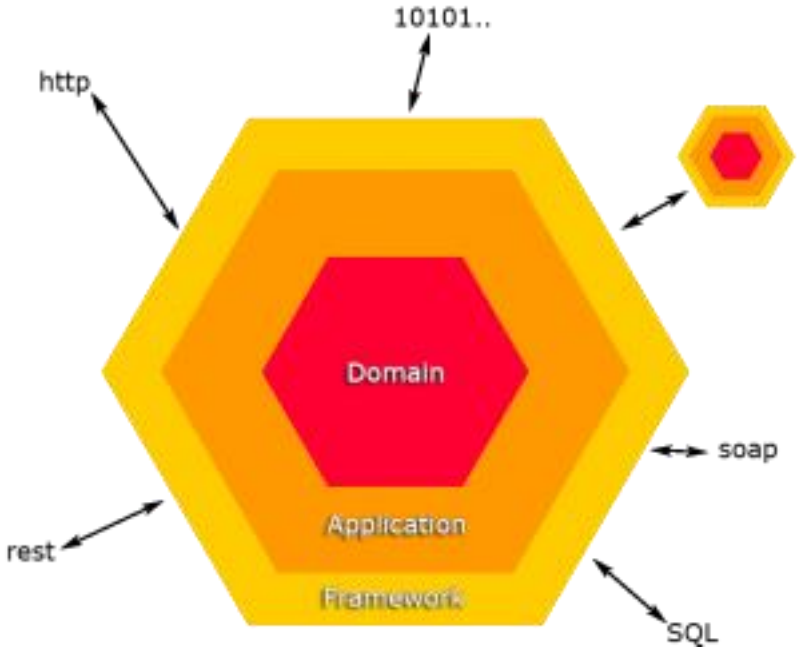
6.3. Back End

6.3.1 Conceptos previos

sbt(simple build tool): Herramienta de construcción (build tool) para Scala y más lenguajes

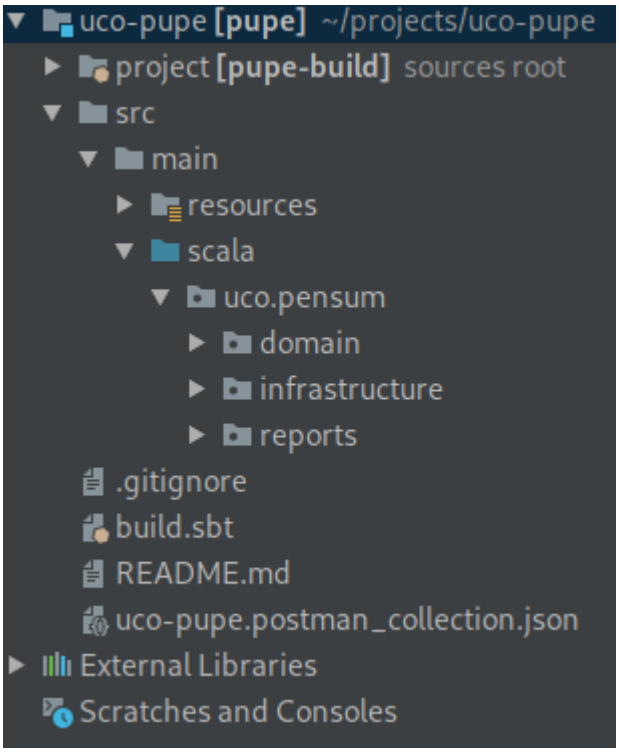
build tool: Programa para automatizar ejecutables de aplicaciones sobre código fuente. La construcción incluye compilación, enlaces, empaquetamiento e instalación de librerías externas basadas en repositorios propios.

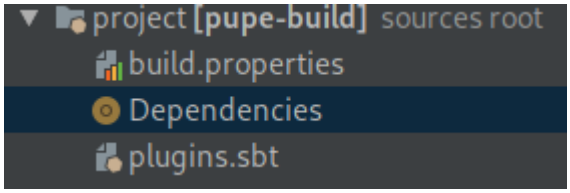
El diseño del proyecto se enfocó completamente orientado al dominio, arquitectura hexagonal y capas de cebolla. Esto en conjunto brinda fácil entendimiento del funcionamiento del código, capas para la protección de datos, modularidad (pequeños componentes, lo cual ayuda al fácil entendimiento del código), separación de responsabilidades, y como su nombre lo indica, enfoque en el diseño, capa que únicamente entiende lenguaje del negocio y la cual contiene todos los servicios y reglas de negocio. La siguiente imagen es fiel reflejo de la arquitectura:



En el centro se encuentra el dominio, el cual contiene todas las entidades o objetos de valor (value objects, en inglés). La capa exterior a esta, la capa de aplicación, contiene los servicios sobre estas entidades o reglas de negocio. La siguiente, marco de trabajo (framework), se puede ver como la capa anticorrupción, que es la que evita que algo no deseado entre a nuestro dominio, más adelante se hablará de los TypeClasses que son los encargados de lograr este propósito. Finalmente, por cada cara del hexágono habrán puertos que son “clientes” que necesitan hacer uso de nuestras reglas de negocio de alguna u otra forma. Acá, estaŕa nuestro puerto HTTP que es por medio del cual nuestro cliente Front se comunicará con el back y además nuestro puerto con la base de datos para la persistencia de los datos.

La estructura del empaquetamiento de la aplicación se muestra a continuación





La carpeta project contiene archivos necesarios para la construcción del proyecto con sbt: build.properties contiene la versión de sbt `sbt.version=1.2.6`

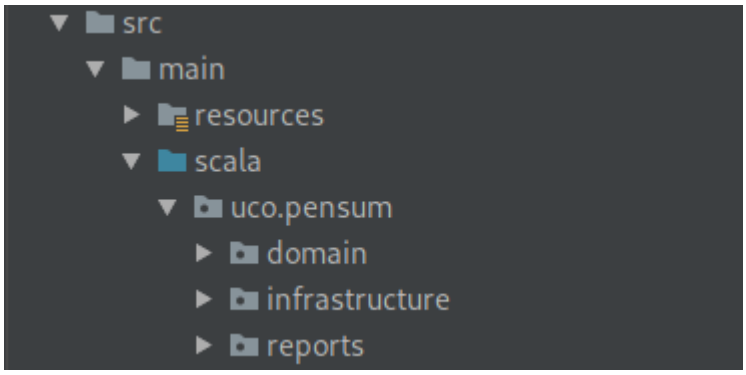
Dependencies es un objeto con variables de todas las librerías que usa el proyecto

```
object Dependencies {

  val akkaHttpVersion = "10.1.6"
  val circeVersion = "0.9.3"
  val monixVersion = "3.0.0-RC2"

  lazy val postgresql  = "org.postgresql" % "postgresql" % "42.2.5"
  lazy val scalaTest = "org.scalatest" %% "scalatest" % "3.0.5"
  lazy val httpCirce = "de.heikoseeberger" %% "akka-http-circe" % "1.23.0"
  lazy val cats = "org.typelevel" %% "cats-core" % "1.5.0"
  lazy val akkaHttp = "com.typesafe.akka" %% "akka-http" % akkaHttpVersion
  lazy val httpCors = "ch.megard" %% "akka-http-cors" % "0.4.0"
  lazy val akkaHttpTestkit = "com.typesafe.akka" %% "akka-http-testkit" % akkaHttpVersion
  lazy val monix = "io.monix" %% "monix" % monixVersion
  lazy val monixCats ="io.monix" %% "monix-cats" % monixVersion
  lazy val slick = "com.typesafe.slick" %% "slick" % "3.3.0"
  lazy val pureConfig = "com.github.pureconfig" %% "pureconfig" % "0.11.1"

  lazy val logBack = "ch.qos.logback" % "logback-classic" % "1.2.3"
  lazy val scalaLogging = "com.typesafe.scala-logging" %% "scala-logging" % "3.9.2"
  lazy val slf4j = "org.slf4j" % "slf4j-api" % "1.7.10"

  lazy val circeCore = "io.circe" %% "circe-core" % circeVersion
  lazy val circeGeneric = "io.circe" %% "circe-generic" % circeVersion
  lazy val circeParser ="io.circe" %% "circe-parser" % circeVersion
  lazy val circeJava ="io.circe" %% "circe-java8" % circeVersion
  lazy val jwtCirce = "com.pauldijou" %% "jwt-circe" % "2.1.0"

  lazy val googleApiClient = "com.google.api-client" % "google-api-client" % "1.28.0"
  lazy val googleApiServices = "com.google.apis" % "google-api-services-drive" % "v3-rev20190501-1.28.0"
  lazy val googleOAuthClient = "com.google.oauth-client" % "google-oauth-client-jetty" % "1.28.0"
}
```

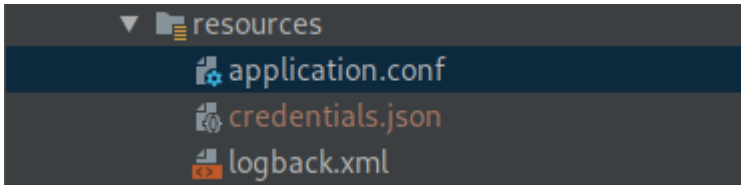
y plugins.sbt contiene dos plugins(uno para formateo de código con solo propósitos de organización y el otro para habilitar la construcción de la aplicación de forma nativa en Docker).

```
addSbtPlugin( dependency = "com.geirsson" % "sbt-scalafmt" % "1.6.0-RC3")
addSbtPlugin( dependency = "com.typesafe.sbt" % "sbt-native-packager" % "1.3.23")
```

Ahora, todo el código fuente se encuentra sobre el directorio src como se muestra a continuación:

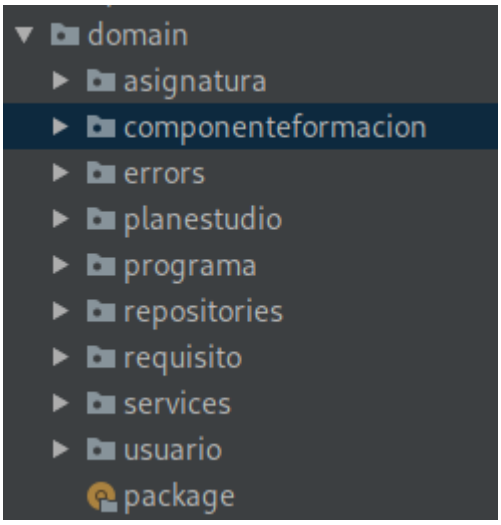


La carpeta resources tiene únicamente archivos de configuración:

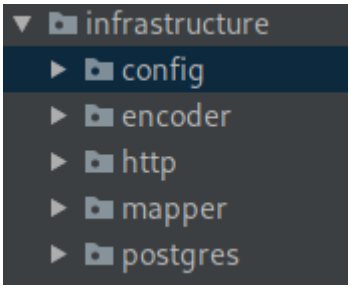


Sobre la carpeta scala se encuentra todo el código fuente Scala.

En la carpeta dominio se encuentra la parte lógica de la aplicación. En ella se encuentran las entidades de dominio, errores, repositorios y servicios. Todo lo presente en esta carpeta utiliza un lenguaje ubicuo y solo entiende reglas de negocio, quiere decir que nadie presente en esta carpeta entiende otro idioma que no sea de dominio.

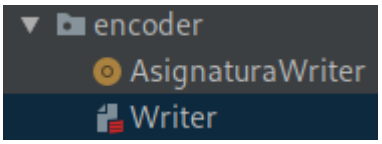


Para la conexión con el mundo exterior como bases de datos, puertos y demás está la carpeta infraestructura. Además de operaciones adicionales como transformadores de objetos y todo aquello que no tenga que ver con la parte lógica de la aplicación.

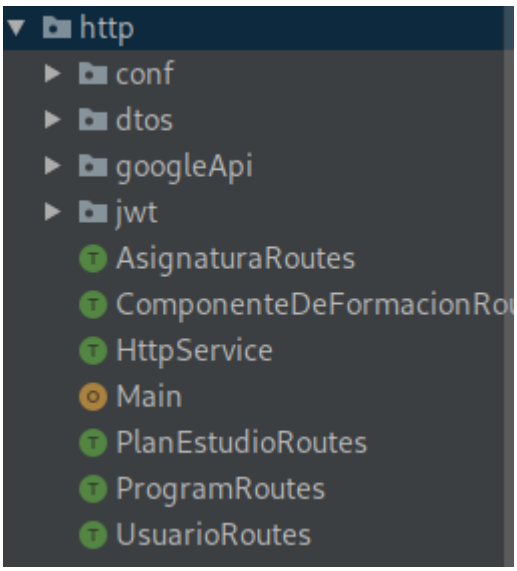


Se tiene una carpeta config que contiene un objeto que representa la configuración de credenciales para la conexión con OAuth.

Una carpeta encoder que contiene un Type Class genérico para la escritura de archivos csv y sus instancias.

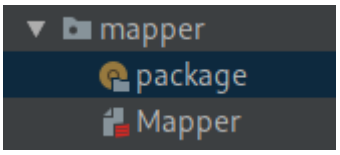


Un puerto http en el cual se encuentran las rutas http para conectar el mundo exterior con la aplicación, acá se definen todos los servicios REST.



Una carpeta mapper que contiene un Type Class para las transformaciones de tipos. Este con el fin de hacer transformaciones entre Records, entidades de dominio y DTOs. Esta se puede considerar como la capa anticorrupción de la aplicación. Un objeto JSON es recibido vía HTTP y es serializado en un objeto DTO gracias a circe, luego, este objeto es validado por una entidad de dominio, una vez validado se convierte ahora en una entidad que el negocio como tal entiende, se realizan operaciones sobre este y finalmente se debe responder al cliente. A la respuesta debemos nuevamente retornar un JSON, por lo cual este objeto de dominio no servirá, en este momento el puerto HTTP gracias a las instancias de nuestra TypeClass de transformación,

transforma el objeto de dominio en nuestro DTO de respuesta y finalmente circe se encargará de deserializar este objeto en JSON.

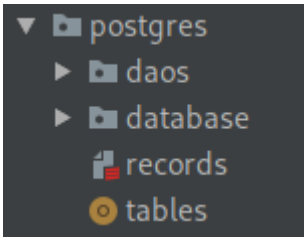


Un ejemplo de una instancia para estas transformaciones se muestra a continuación:

```
implicit def PlanDeEstudioToRespuesta
  : Mapper[PlanDeEstudio, PlanDeEstudioRespuesta] = Mapper(
  plan ⇒
    PlanDeEstudioRespuesta(
      id = plan.id.getOrElse(""),
      inp = plan.inp,
      creditos = plan.creditos,
      horasTeoricas = plan.horasTeoricas,
      horasLaboratorio = plan.horasLaboratorio,
      horasPracticas = plan.horasPracticas,
      programId = plan.programId,
      fechaDeRegistro = plan.fechaDeRegistro,
      fechaDeModificacion = plan.fechaDeModificacion
    )
)
```

Es una instancia implícita (implícita ya que al ser importada nuestro TypeClass busca las instancias y no es necesario entregarlas explícitamente) que transforma la entidad de dominio PlanDeEstudio en el objeto DTO PlanDeEstudioRespuesta.

Finalmente, la carpeta postgres contiene DAOS (Representación de los datos de la base de datos usando Slick), records y un singleton con las tablas y el setup de estas para ser creadas desde el Main de la app.



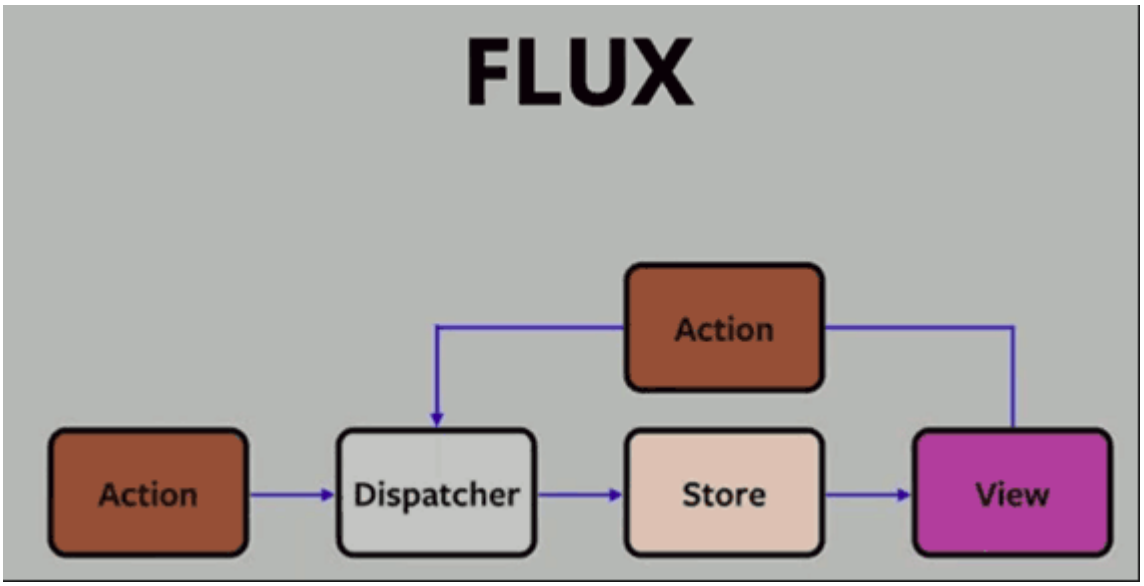
6.3.2 Front End

La Aplicación desde sus inicios se creó como un reto técnico a las habilidades adquiridas en la cátedra y la práctica, siendo la capa de presentación la más importante al momento de ofrecer una experiencia de usuario con altos índices de estética y flujos intuitivos que hicieron de la experiencia de usuario un factor diferenciador, a la hora de interactuar con la aplicación. Para esto usamos una librería creada por facebook llamada React, que como fin principal, tuvo separar las responsabilidades de los objetos de presentación, por medio de lo que llamaron "Componentes"; unidades de abstracción que permiten de una manera versátil, distribuir la vista de la aplicación, de manera que pudieran ser reutilizadas, tener sus propios estilos y en un momento determinado ser reubicadas o porqué no reemplazadas sin tener un impacto significativo en ninguno de los demás componentes de la misma. React fue creada para representar la capa de presentación (V) del modelo MVC, por lo tanto permite una gran cantidad de usos y facilidades al momento de ser usada.

En react se trabaja con DOM Virtual, el cual permite una sincronización con el DOM real en el que los estados de los componentes, el renderizado y los eventos se orquestan por medio del React DOM lo que hacer que la manera declarativa de los componentes de react sea posible.

La arquitectura que se usó para la implementación de la capa de presentación en React fue la arquitectura flux, arquitectura ideada también por Facebook para la implementación de su librería en el que el protagonista a facilitar era el flujo de datos y los estados de los componentes. Facebook idea la arquitectura flux, porque el modelo MVC se estaba quedando corto en grandes aplicaciones, al momento de manejar los datos de la misma y al momento de manejar errores, ya que utilizaba una comunicación bidireccional entre los modelos y los controladores. Con la arquitectura flux la comunicación es unidireccional, por lo cual los datos viajan desde la vista por medio de acciones y llegan a un store desde el cual se actualiza la vista de nuevo; de esta manera

en el store como única fuente de datos, es más fácil saber qué está pasando en la aplicación y controlar sus comportamientos.



Los actores de la arquitectura front son:

- Vista: Componentes WEB.
- Store: Se encarga de almacenar los datos o estados de la aplicación, se puede comparar de cierto modo con el modelo de la aplicación.
- Acción: Es una intención de hacer algo por medio de un objeto javascript.
- Dispatcher: Es el encargado de recibir la acción de la vista y propagarse hasta el store; funciona como un orquestador que desacopla la vista del store.

Para la implementación de la arquitectura flux manejamos MOBX como orquestador de estados de los componentes de la aplicación, el cual permite volver reactiva la arquitectura de los datos y al actualizarse el estado de los componentes se actualizará la vista por reacción al cambio al existir un componente observador y un store observado, el cual al cambiar el estado notifica a su componente observador.

6.4. Desarrollo

6.4.1 Back End

Para el desarrollo del back se basó la solución en un lenguaje de programación que brinda ayudas en tiempo de compilación, estas ayudas incluyen: seguridad de tipos y modularidad. Scala, no es solo basado en el paradigma funcional sino también en orientación a objetos. El desarrollo se hizo con excelentes prácticas y con un uso adecuado de acuerdo a los requerimientos del lenguaje. La inmutabilidad de datos es algo importante para resaltar, ya que en todo el código la declaración de variables y el manejo de objetos carecen de mutación de datos. Además, gracias al mismo lenguaje y algunas librerías que a continuación se describirán, la solución no es monohilo, las operaciones están desarrolladas y orquestadas para sacar provecho de la CPU y se están haciendo llamados paralelos para los servicios sobre la base de datos y autenticación de usuarios, lo cual incrementa el rendimiento y reduce la latencia de la aplicación.

A continuación la lista de librerías usadas:

postgresql: Proveedora de drivers para la conexión JDBC con bases de datos postgres.

scalaTest: Provee las herramientas necesarias para hacer ya sea pruebas unitarias o pruebas de integración del código.

circeCore, circeGeneric, circeParser, jwtCirce, httpCirce: Circe es una librería que provee instancias para analizar sintacticamente(del inglés parsing) objetos JSON. La forma en que los endpoints del servicio HTTP está recibiendo las peticiones es via JSON. A su vez, las peticiones de JSON Web Tokens.

cats: Contiene abstracciones para programación funcional. Su nombre es una abreviación de la palabra category.

akkaHttp: Contiene los recursos necesarios para exponer endpoints http.

httpCors: Límita los clientes que pueden acceder a los servicios http.

akkaHttpTestKit: Conjunto de utilidades para pruebas sobre akka http.

monix: Librería de alto desempeño para composiciones asíncronas en programas basados en eventos. Altamente importante para el código ya que todas nuestras operaciones sobre la base de datos usando slick retornan Futures, el cual no es transparente referencialmente y al ser compuesto produce desintegridad en los datos. Gracias a Task, un tipo de Monix, este problema es resuelto.

slick: Mapeador relacional funcional reactivo para Scala. Permite representar los datos y relaciones de forma funcional y realizar operaciones sobre los datos con funciones monádicas.

pureConfig: Cargador de archivos de configuración.

logBack: Pre requisito para scala logging

scalaLogging: Conveniente y rápida librería de logueo que envuelve slf4j.

slf4j (Simple Logging Facade for Java): Sirve como fachada o abstracción de varios marcos de trabajo de logueo.

googleApiClient: Api contenedora para las bibliotecas Java de Google

6.4.2 Front End

Finalmente, cada servicio (Front y Back) contiene los recursos necesarios para la construcción de sus respectivas imágenes Docker , después de ello y con la ayuda de docker compose, es directo el despliegue de la aplicación en cualquier máquina.

Sobre el proyecto front, hay un archivo Dockerfile que tiene la configuración para la construcción de la imagen. Obviando la instalación previa de docker y docker compose sobre la máquina que se quiere desplegar el proyecto, ejecutar docker build . -t <nombre de la imagen que se quiere construir(ej. pupe-ui)>.

Para el back, sobre el proyecto back ejecutar el comando sbt docker:publishLocal.

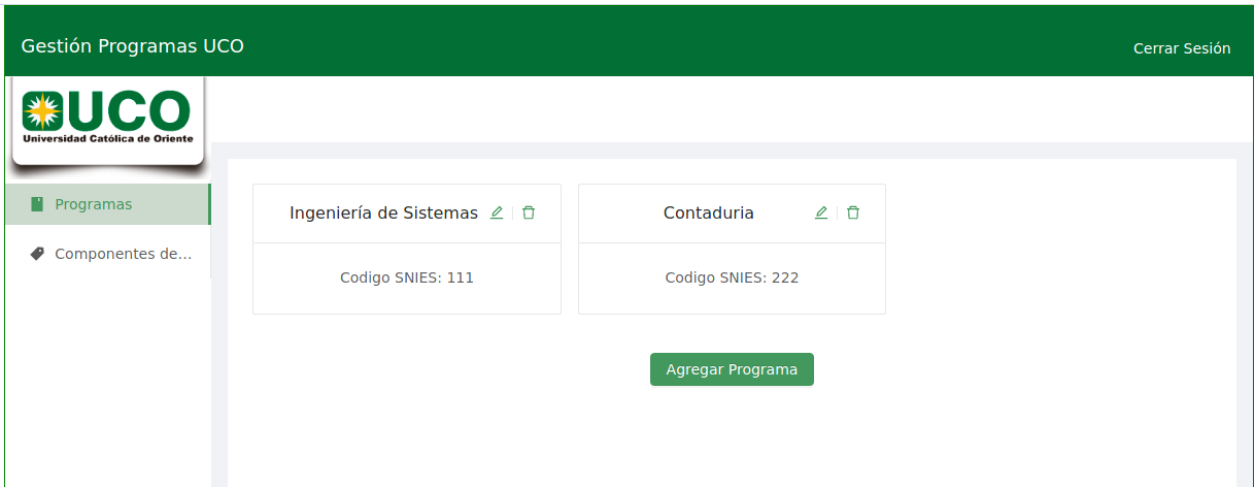
Después de este proceso las imágenes de ambos proyecto habrán sido creadas.

Para la creación de los contenedores, sobre el proyecto front hay un archivo docker-compose.yml que contiene la configuración de los servicios con sus dependencias y la imagen postgres. Sobre este proyecto ejecutar el comando docker-compose up y los 3 servicios serán levantados. El back sobre el puerto 8080, el front sobre el puerto 3000 y postgres sobre el puerto 5432. Todos ellos en localhost.

En un navegador visitar localhost:3000.

El portal de inicio de la aplicación cuenta con el logo de la Universidad y un botón de Inicio de Sesión con una cuenta de Gmail. Una vez ingresado con la cuenta de gmail se mostrará un menú como el siguiente:

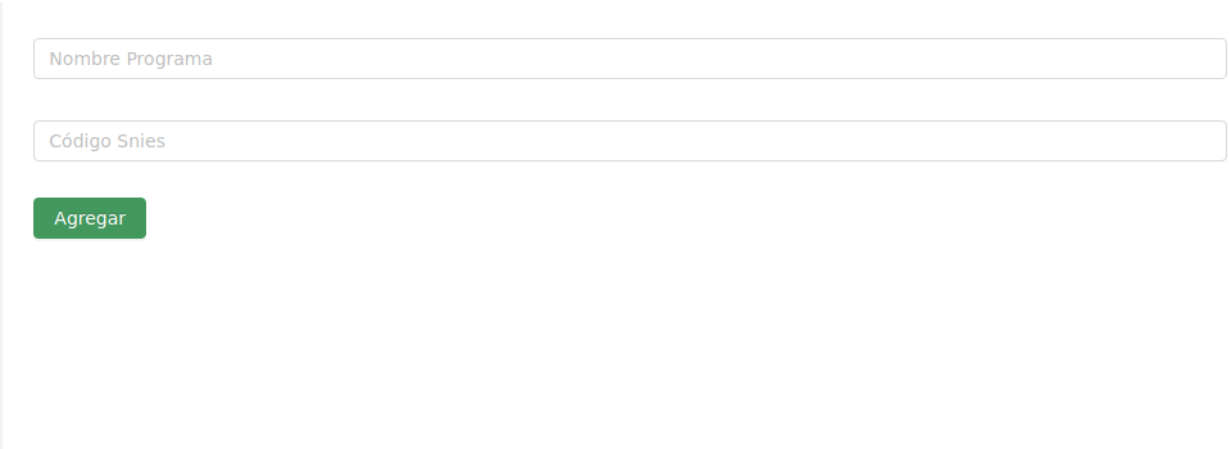
Al dar click sobre el botón de programas, la lista de programas será mostrada de la siguiente forma en el portal.



Las acciones que se pueden realizar sobre el menú de programas son las siguientes:

1. Agregar un programa
2. Editar un programa
3. Eliminar programa

Para agregar un programa se selecciona el botón ‘Agregar programa’ y la siguiente ventana con los datos requeridos para su creación saldrá:

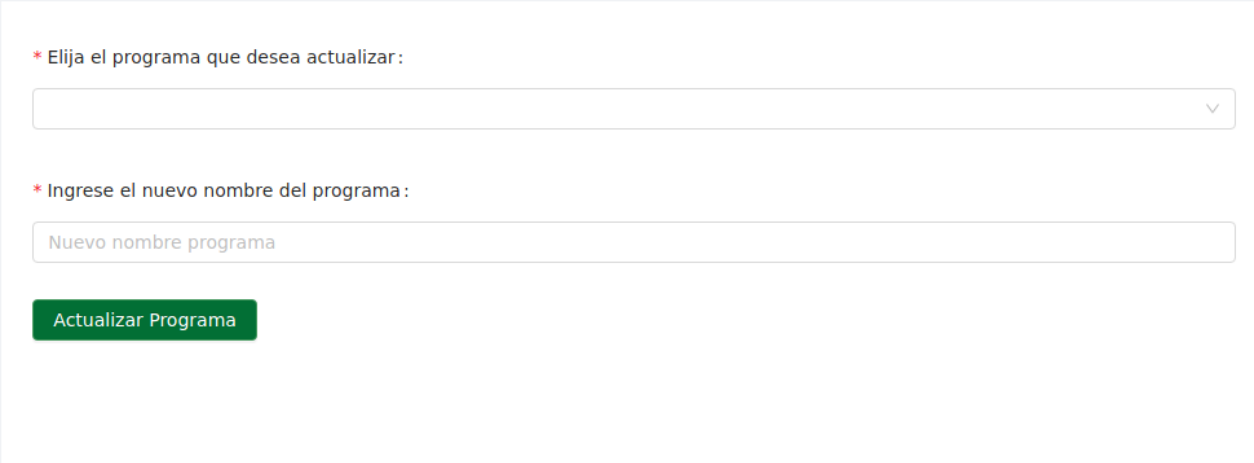


Una vez creado el programa el sistema lo almacenará y la lista de programas aparecerán con el nuevo programa.

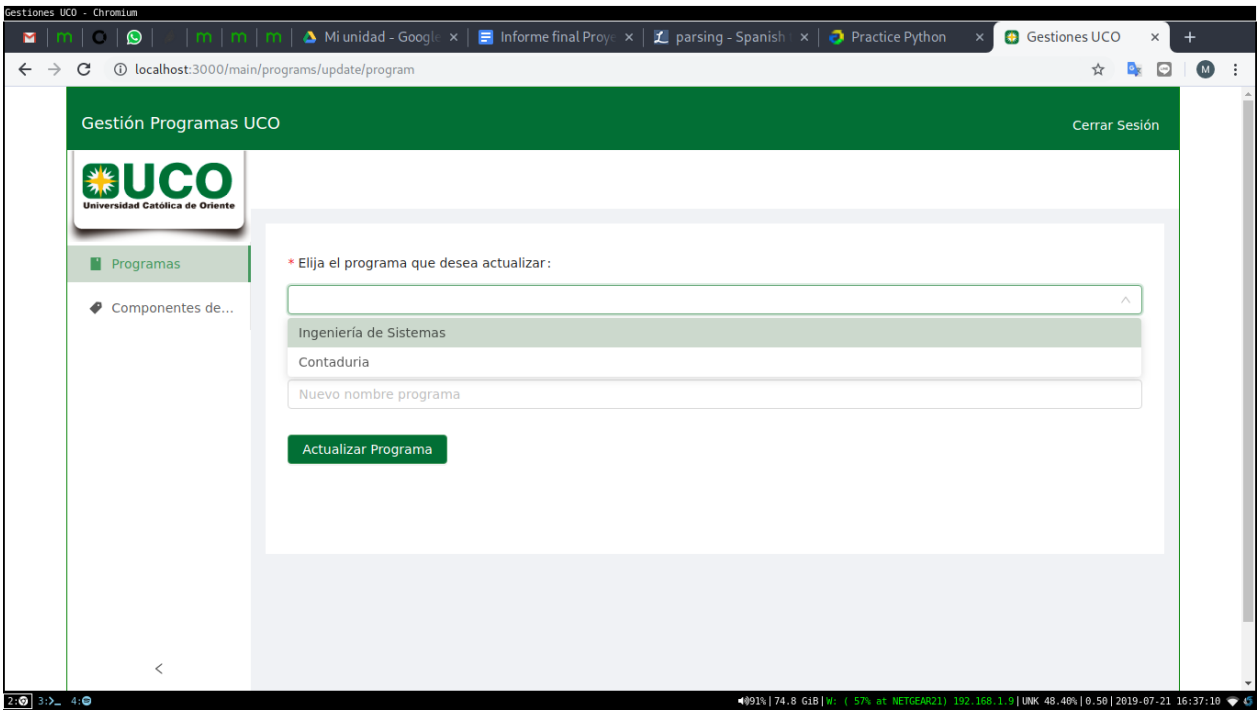
Cada programa listado tiene los siguientes dos botones en la parte superior:

Para editar o eliminar, respectivamente.

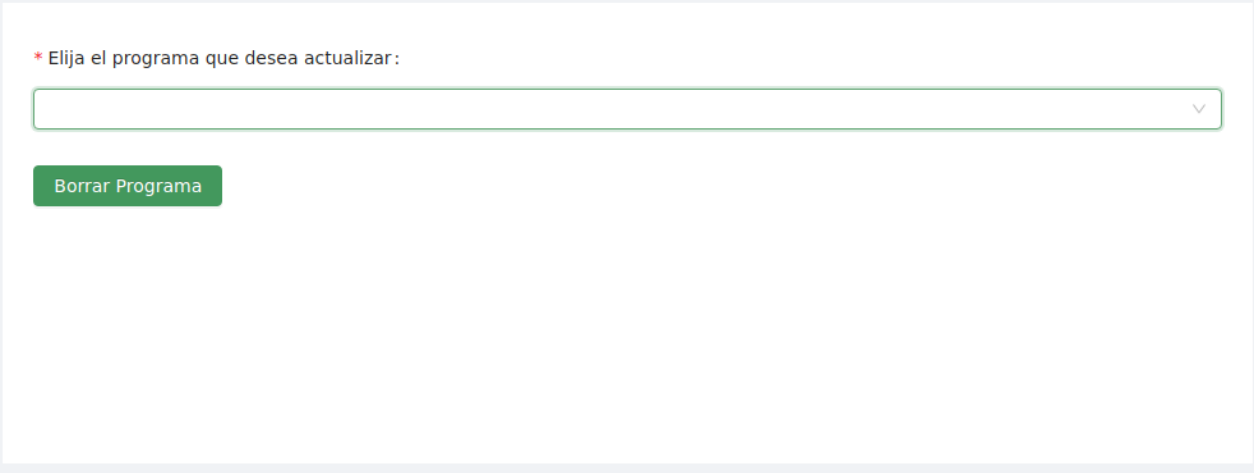
Para editar un programa se da click sobre  y una ventana preguntando por los datos que se quieren actualizar sobre el programa saldrá de la siguiente forma:



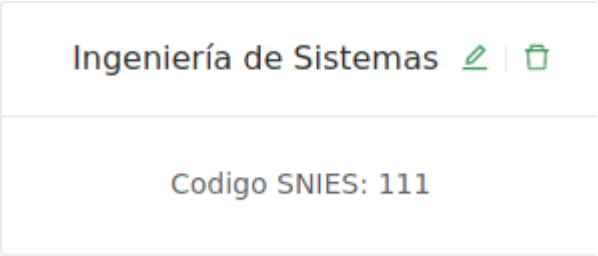
Para elegir el programa se despliega la lista de programas existentes.



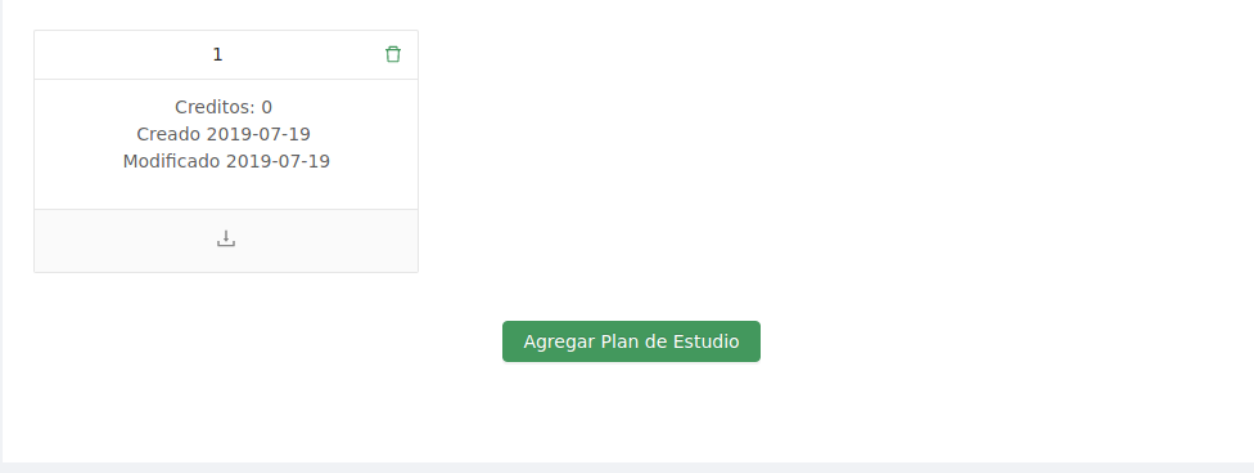
Para borrar el programa se selecciona el botón y una ventana se abre con la lista de programas existentes:



Si lo que se quiere es ver los planes de estudio de los programas entonces se presiona directamente sobre el programa

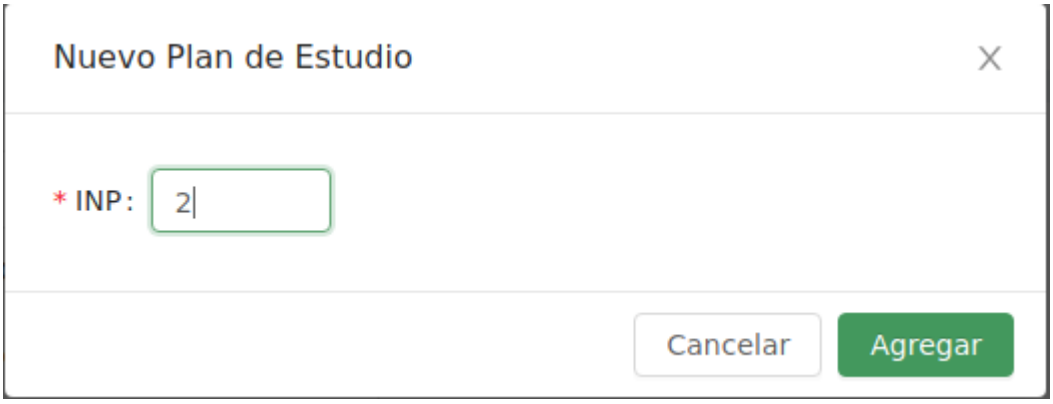


Y la lista de planes de estudio aparecerá con los planes de estudio de forma descendente.



Quiere decir que si se crea un nuevo plan de estudio ‘2’, este será el primero en aparecer sobre la lista así:

1. Se crea el plan de estudio seleccionando la opción de ‘Agregar Plan de Estudio’



Se da clic en Agregar y el plan de estudio se almacenará.

2. La lista de planes de estudio se mostrará de forma descendente así:

NOTA: El campo créditos de cada plan de estudio es dinámico, quiere decir que su valor incrementa al agregar asignaturas en cada uno de ellos como veremos más adelante. Al momento, entonces, ninguno de los dos planes de estudio tiene asignaturas y por ellos los créditos son 0.

Sobre cada plan de estudio se pueden hacer dos acciones:

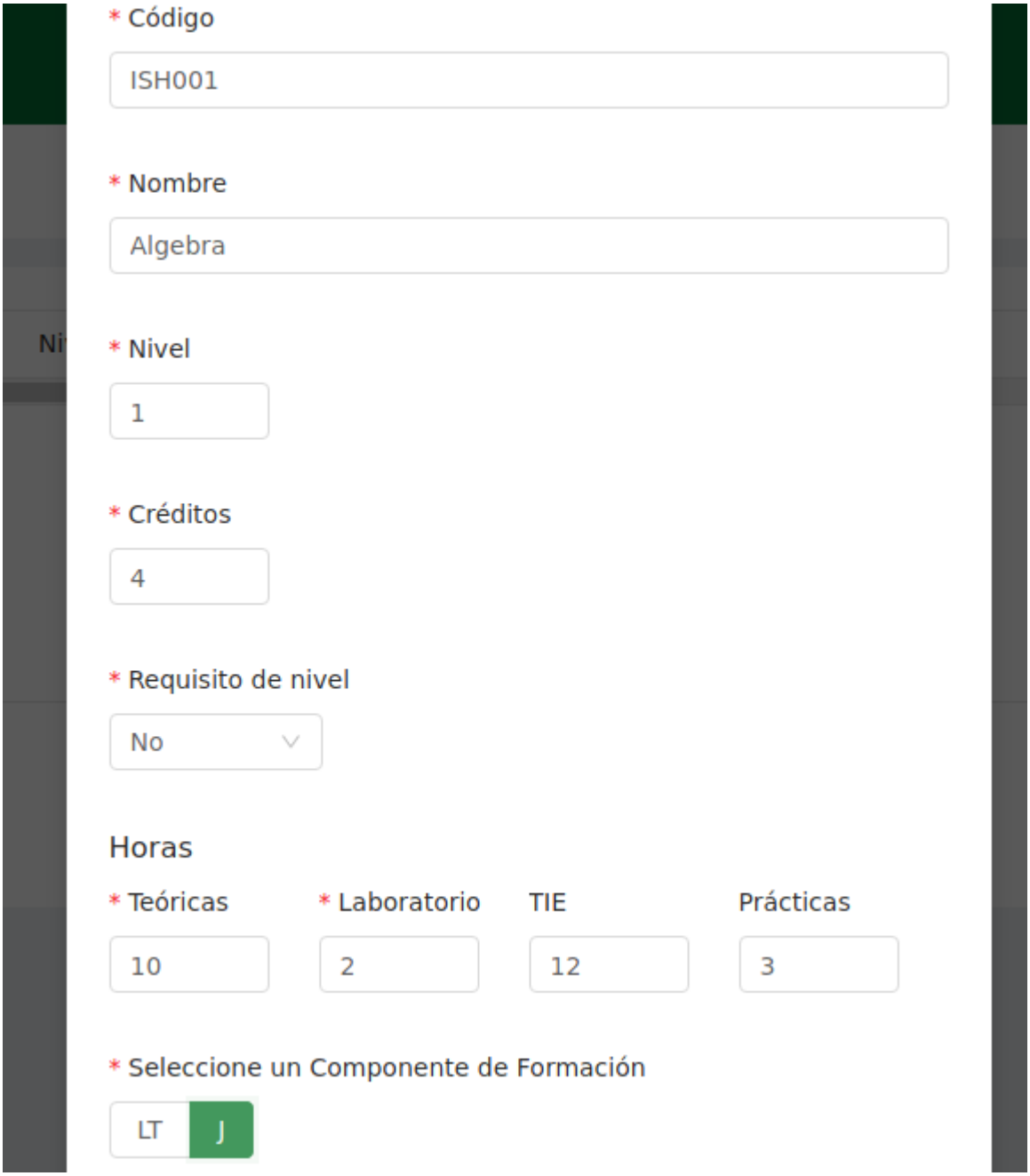
1. Eliminar plan de estudio, presionando el botón de la papelera en la parte superior derecha de la tarjeta de cada uno  .

2. Descargar el reporte de asignaturas presionando el botón de descarga que se encuentra debajo del INP. El reporte será descargado en formato xls

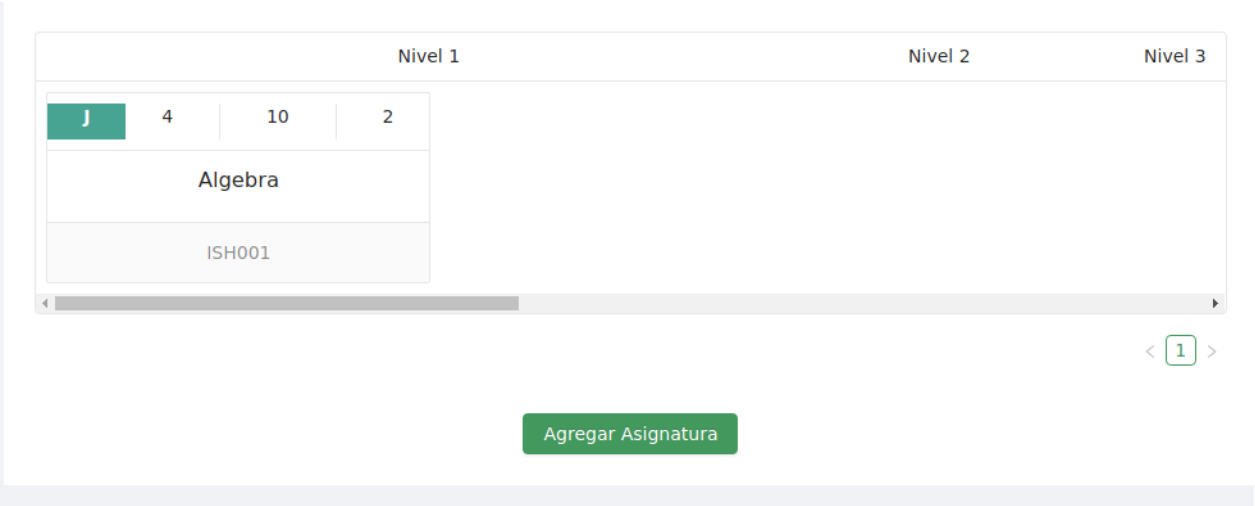
Para empezar a agregar asignaturas sobre los planes de estudio se selecciona el plan de estudio y una ventana como la siguiente se abrirá:

Es una matriz por nivel que se llena con tarjetas de asignatura a medida que se agregan como veremos a continuación.

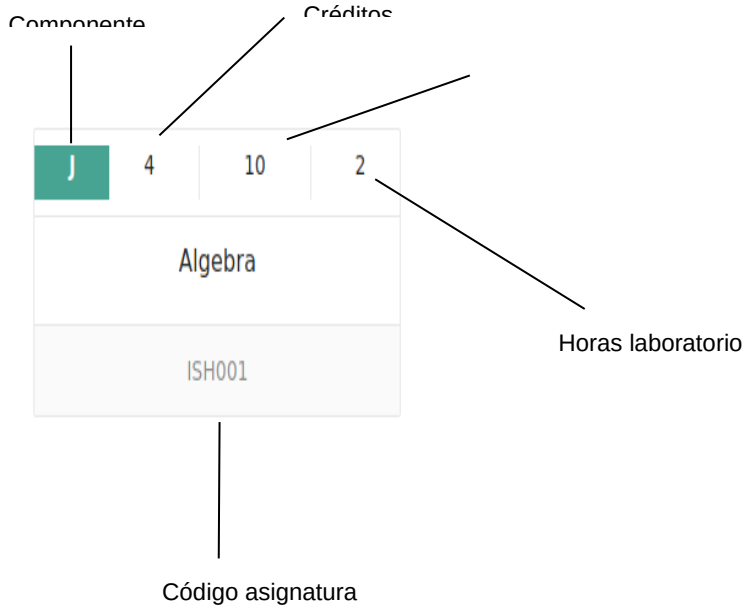
Se agrega una asignatura presionando el botón de Agregar Asignatura.



Una vez agregada la asignatura, la matriz se empieza a llenar dinámicamente de la siguiente forma:



Los elementos de la asignatura son los siguientes:



Presionando cada materia se puede visualizar sus campos:

Se observa el componente de formación en la parte superior izquierda con su respectivo color, que previamente debió haber sido creado. Seguido de este, vemos su código, nivel y créditos. Debajo de esto las horas respectivas y finalmente los requisitos que esta asignatura tiene.

Cada asignatura tiene 5 operaciones:

1. Revisión de la descripción de cambios
2. Subir un archivo con el plan de estudio de la asignatura o cualquier otro archivo para lo cual es obligatorio dejar una descripción.

3. Revisar plan de asignatura el cual muestra una ventana con los archivos que se han subido para dicha asignatura.

4. Actualizar asignatura, excepto su código.

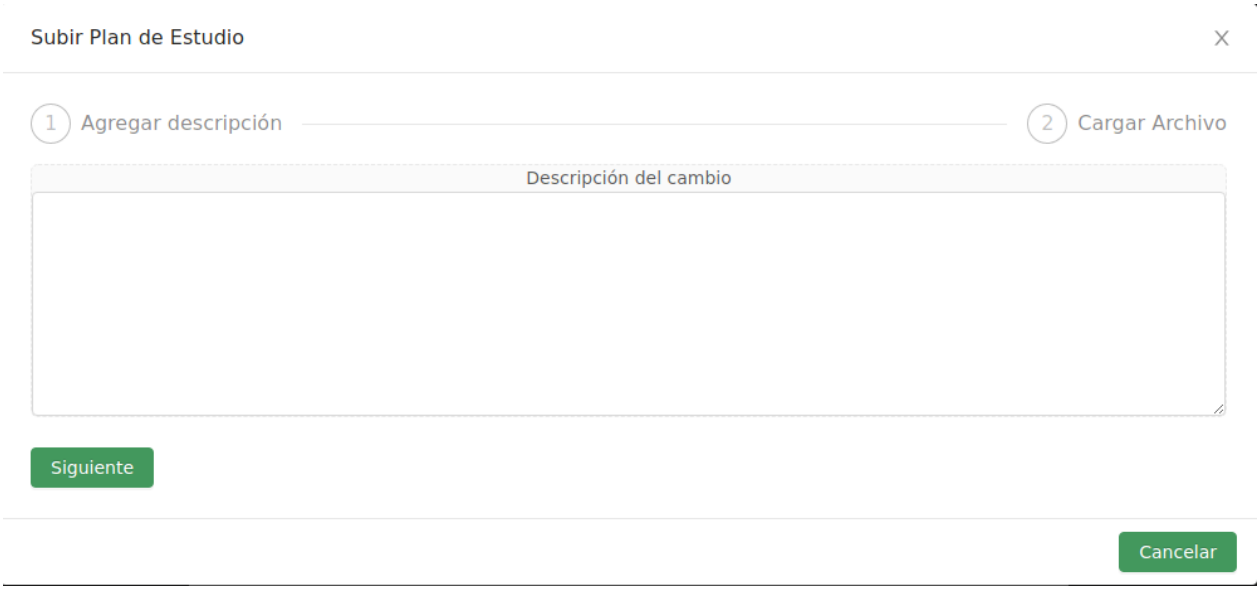
5. Eliminar asignatura

En el momento solo se ha realizado una modificación sobre la asignatura, se cambió de componente de formación, de Jurídica a Ciencia Exacta lo cual se puede evidenciar en las dos imágenes anteriores. Así se evidencian los cambios presionando sobre botón

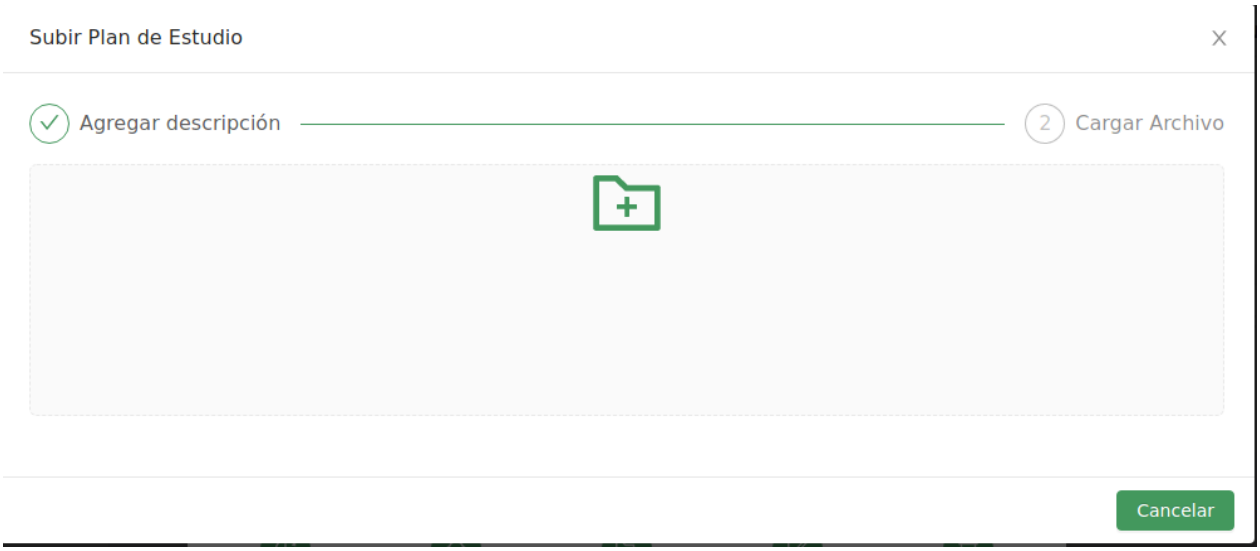
: 

Y así en una línea de tiempo con los cambios que se han ido realizando.

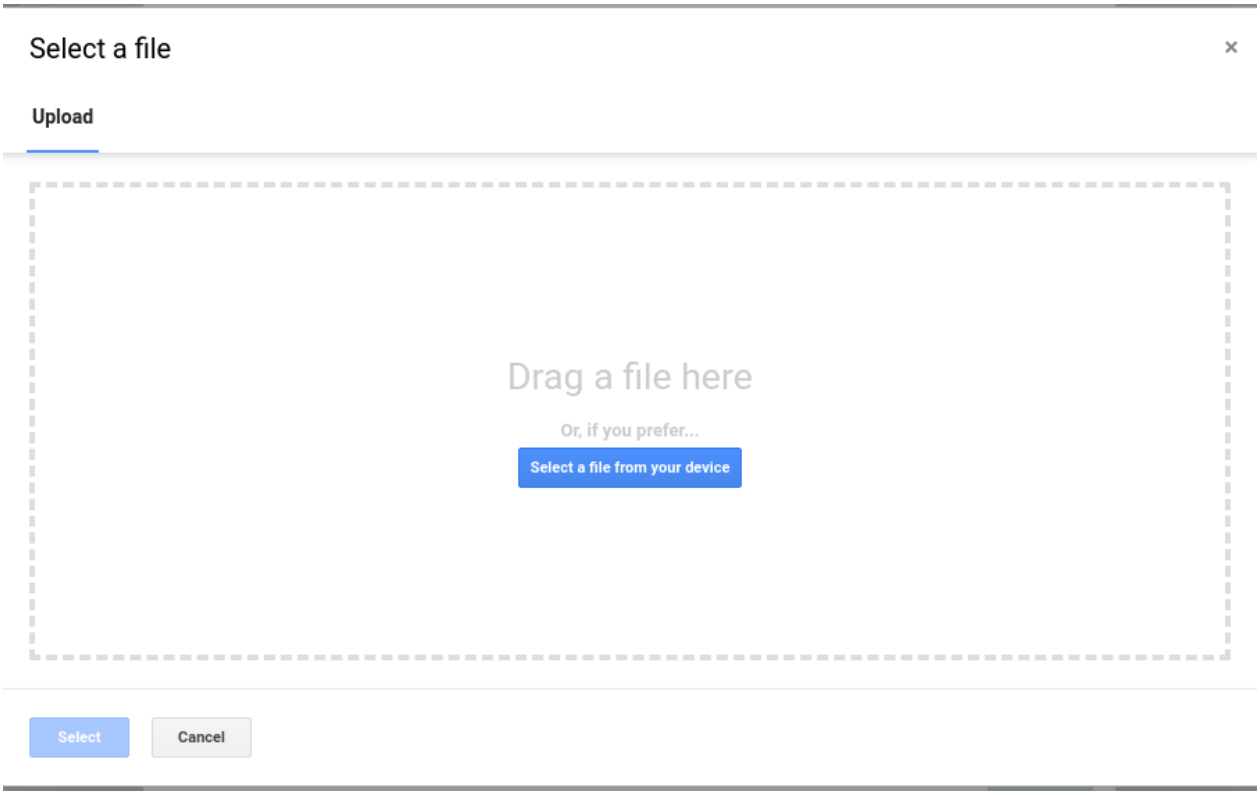
Ahora, si se sube un archivo presionando el botón:



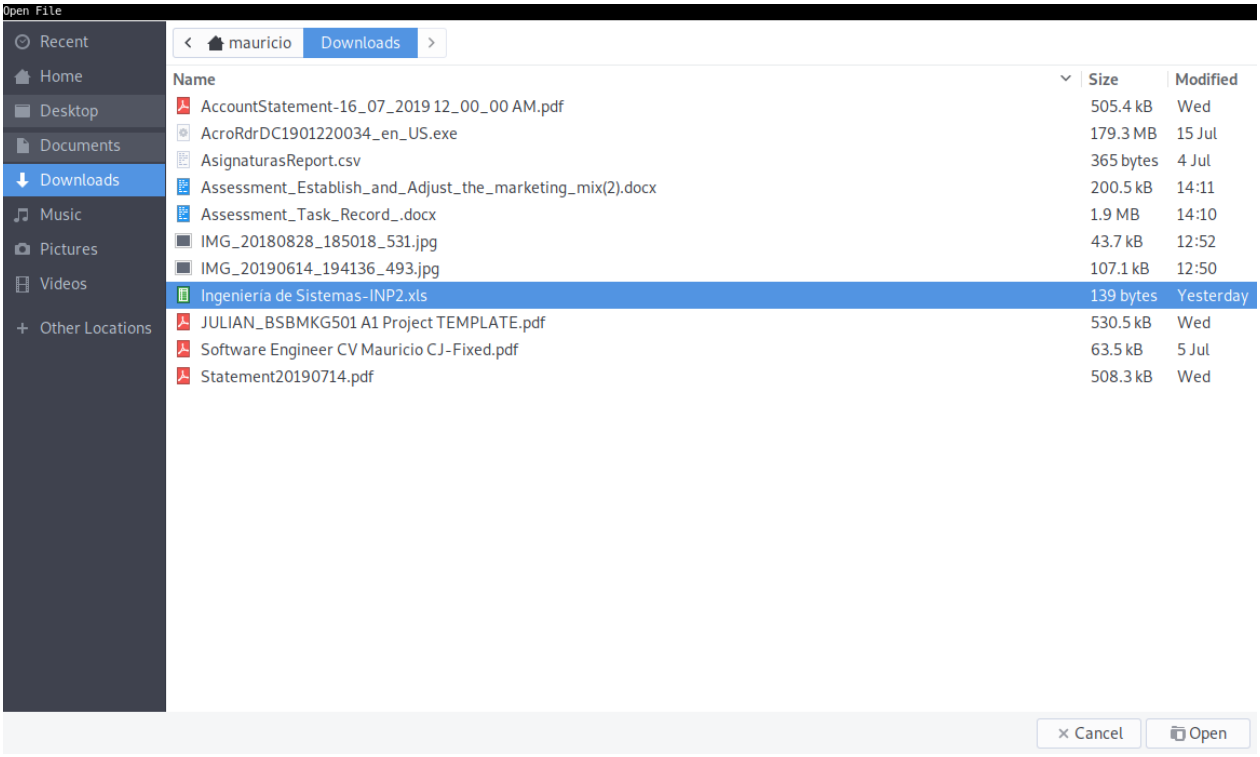
Primero se debe dejar una descripción. Suponga que se quiere subir un archivo con el plan de estudio para el semestre 2-2019 y su descripción será ‘Se adjunta archivo de plan de estudio para semestre 2-2019’.



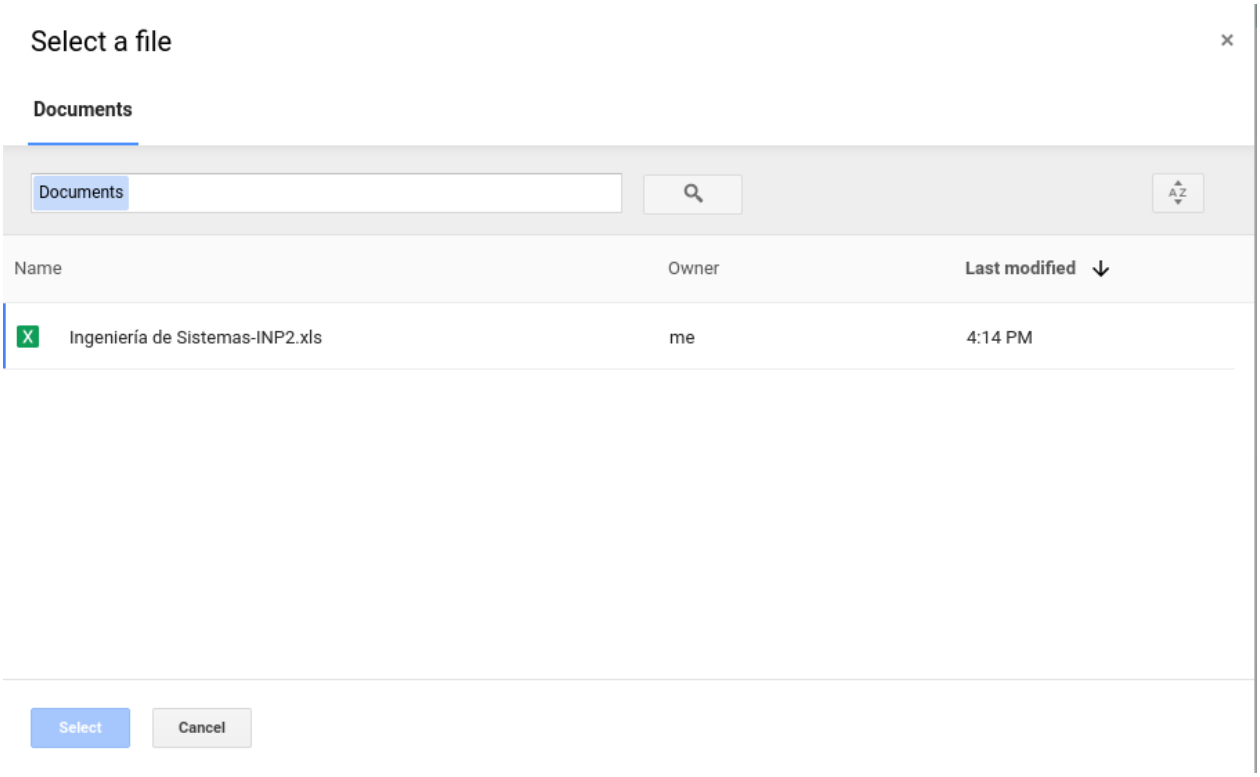
Luego se selecciona el archivo presionando la carpeta



Se mostrará la siguiente ventana sobre la cual se puede subir un archivo de su máquina presionando ‘Seleccionar archivo de tu dispositivo’.

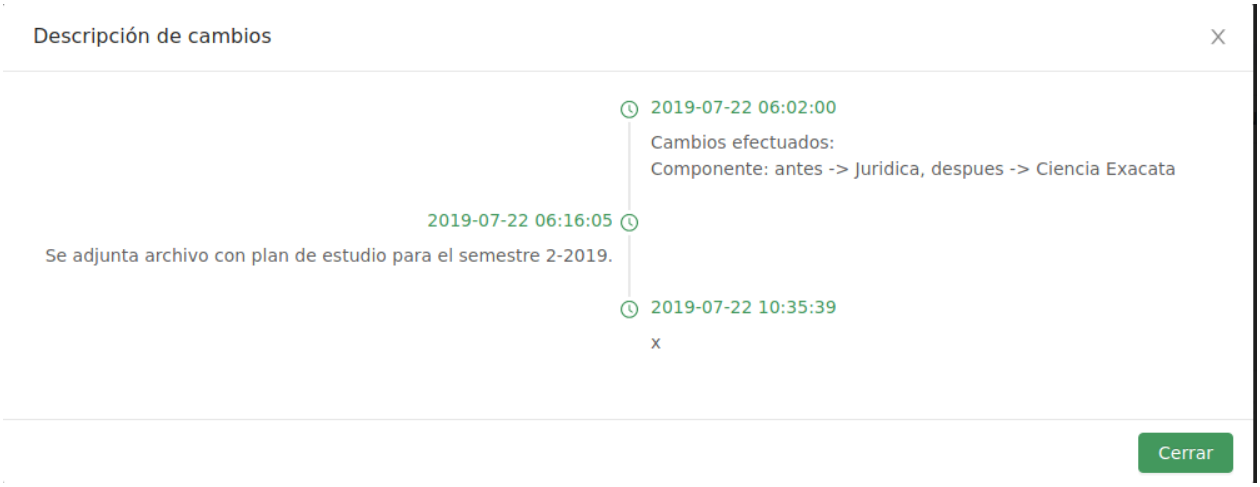


Se subirá el archivo ‘Ingeniería de Sistemas INP2.xls’ el cual podremos visualizar seleccionando el botón el cual muestra una ventana como la siguiente:



Si se selecciona el archivo, se será redireccionados a una nueva ventana donde podremos verlo e inclusive editarlo en caso de que sean documentos office como Word, Excel o PowerPoint.

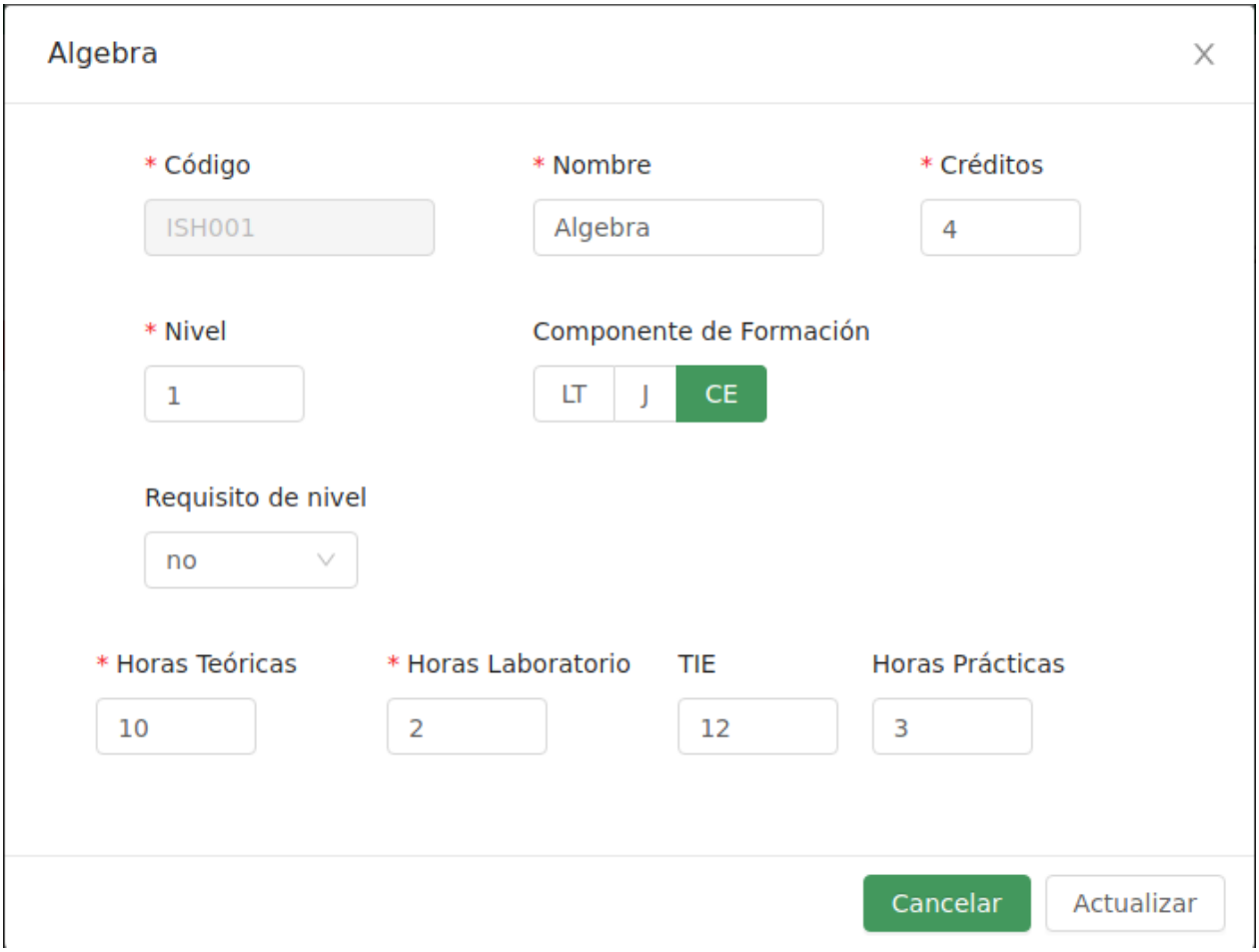
Ahora, si revisamos nuevamente la descripción de cambios:



Los cambios realizados se muestran de forma ascendente hacia abajo de acuerdo a las fechas de modificación, es decir, eñ último cambio será el último en aparecer leyendo de arriba

hacia abajo como se aprecia en la imagen. El último comentario ‘x’ fue el último cambio realizado.

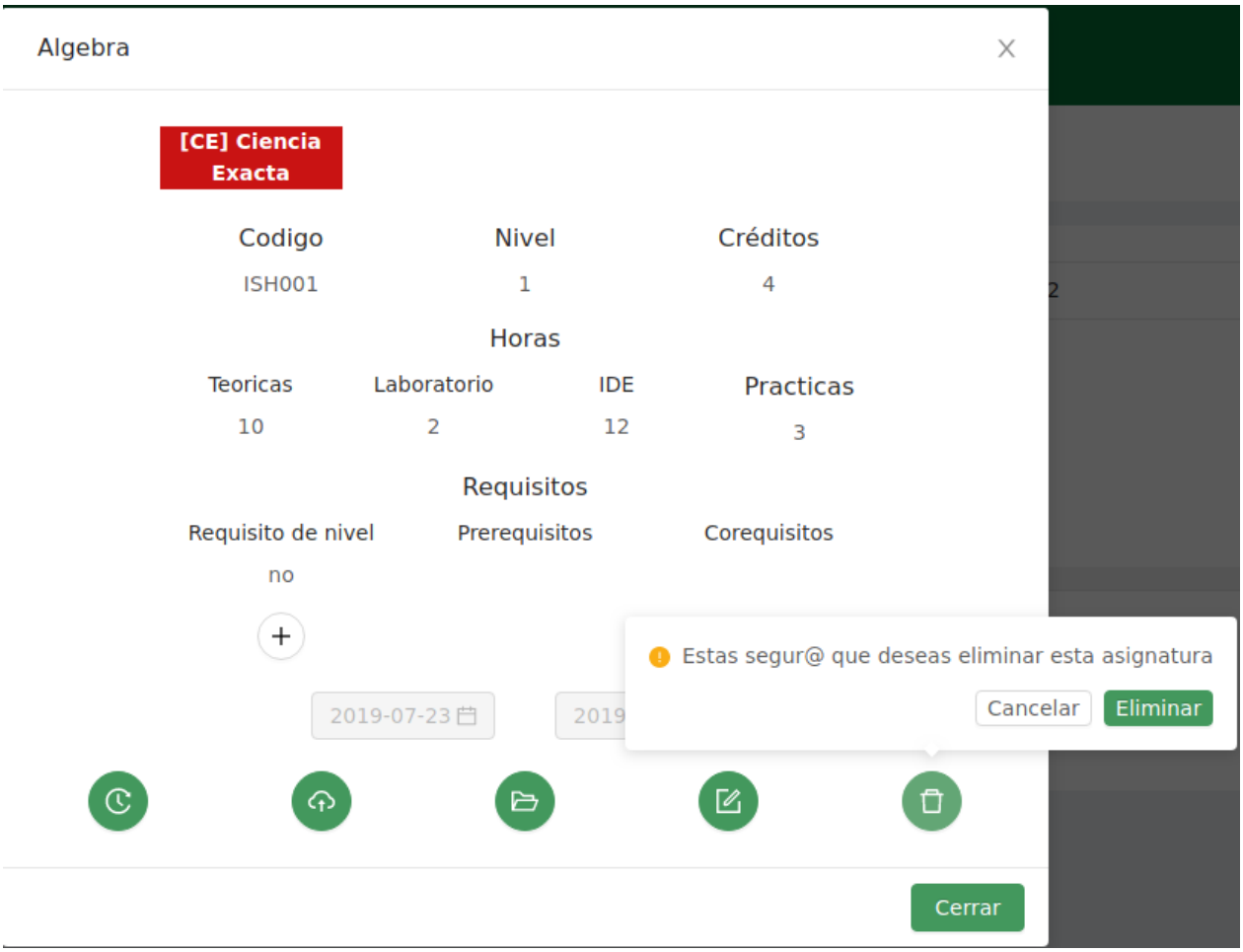
La siguiente propiedad de las asignaturas es su actualización, que se puede realizar presionando el botón



Acá se es posible modificar todos los campos de la asignatura excepto su código.

El trabajo Independiente del Estudiante es calculado al momento de la creación de la asignatura, sin embargo, es posible modificar su valor ya que para ciertas asignaturas no se cumple que este valor es la suma de las horas teóricas y las horas de laboratorio.

Por último, se puede eliminar asignaturas con el botón  , al ser presionado un aviso verificando su eliminación aparecerá encima de él así.



Después de confirmar, la asignatura será eliminada de la base de datos y no se mostrará más en la matriz.

NOTA: Es importante aclarar que ante cualquier procedimiento sobre cualquier componente de la aplicación siempre hay mensajes de lo que se acaba de realizar.

7. CONCLUSIONES

Gracias a lo aprendido durante la carrera, conocimientos adquiridos independientemente y estar rodeados de la industria día a día, se evidencia un gran entendimiento de lo que es hacer software en todo su ciclo de vida, desde el análisis y definición de requerimientos hasta la entrega de un producto realizado con tecnologías en vanguardia. La aplicación de metodologías ágiles para las iteraciones de la entrega de pequeñas partes funcionales y el resultado de un producto robusto desarrollado con tecnologías en vanguardia y patrones y diseños con fundamentos. Todo esto amplía nuestros conocimientos y cimienta los ya existentes. Con el desarrollo de este tipo de proyectos, se evidencia la inmensidad en la que la ingeniería de software se encuentra y la cantidad de retos a los que sus ingenieros deben enfrentarse día a día.

Se entrega acá, el programa de Gestión de programas para la Universidad Católica de Oriente con su objetivo principal, la trazabilidad de planes de estudio de asignaturas.

Además mediante la capacidad adquirida en la cátedra de investigación propuesta por la universidad, se logra una metodología de autoestudio efectiva, en la que el profesional no se vuelve dependiente de la tecnología, si no que aplica su capacidad de razonamiento para elegir los marcos de trabajo que mejor se adapten a la solución con criterios firmes y justificables, y emprende de ser necesario, el camino hacia un aprendizaje para el desarrollo de la misma bajo los mejores estándares, marcos de trabajo y las mejores prácticas; para que el trabajo técnico hable por sí solo del profesional que lo desarrolla y la visión de calidad con la que fue formado.

En otro aspecto, durante el desarrollo del proyecto, se logra consolidar la efectividad de los marcos de trabajo ágiles, en los que se compromete un alcance y se entraga de forma incremental, empezando por un mínimo producto viable(MPV) y luego dirigiendo los requerimientos hacía la visión última del cliente, que ve en su producto opciones de mejora y al tener una versión inicial, amplía el espectro de funcionamiento que se puede lograr mediante el

mismo; todo esto sin sobrepasar el alcance, lo que permite como equipo de desarrollo, tener feedback a tiempo y atención al mismo de manera efectiva y con un impacto mínimo en el desarrollo total de la aplicación, incrementando la satisfacción y la experiencia de atención y compromiso con el cliente.

Resaltamos el trabajo en equipo, como parte fundamental en el proceso; ya que permite hacer uso de la división de responsabilidades y las diferentes capacidades de los integrantes; facilitando la entrega final, al permitirnos, ser partícipes del proyecto con nuestras mejores cualidades y tener diferentes puntos de vista a la hora de de definir puntos críticos del producto, un producto no solamente funcional sino estético y con altos índices de calidad técnica.

REFERENCIAS BIBLIOGRÁFICAS

Blanco, I. y Quesada, V. (s.f.) La gestión académica: criterio clave de la calidad de gestión de las instituciones de educación superior. Recuperado de: http://www.ucv.ve/fileadmin/user_upload/vrac/documentos/Curricular_Documentos/Evento/Ponencias_1/ Blanco_y_Quesada.pdf

Colombia. Ministerio de educación nacional. (s.f.) Plan de estudios. Recuperado de: https://www.mineducacion.gov.co/1621/article-79419.html

García,A.M. (2015). ¿Qué es docker? ¿Para qué se utiliza? Explicado de forma sencilla. Recuperado de http://www.javiergarzas.com/2015/07/que-es-docker-sencillo.html Martin, D. (2018). Tipos de software de gestión. Recuperado de : https://velneo.es/tipos-software-gestion/

Jenifer, K. (2016), Scala Tutorial: Create CRUD with Slick and MySQL. Recuperado de https://medium.com/@kennajenifer1234/scala-tutorial-create-crud-with-slick-and-mysql-1b0a5092899f

Mishra, A. (2017) Start Building RESTful Microservices using Akka HTTP with Scala, Independently published

Monix developers. Monix, (s.f.). Recuperado de https://monix.io/

Obe R., Hsu L., (2017). PostgreSQL: Up and Running. Sebastopol, California: O'Reilly Media, Inc.

Página oficial de RedHat, ¿Qué es Kubernetes?, Recuperado de: https://www.redhat.com/es/topics/containers/what-is-kubernetes

Rico, A.D. (2016) La gestión educativa: Hacia la optimización de la formación docente en la educación superior en Colombia. Sophia 12(1): 55-70. Recuperado de: http://www.scielo.org.co/pdf/sph/v12n1/v12n1a04.pdf

Roche, E. (2013) Que es Git/GitHub?. Recuperado de: https://barradevblog.wordpress.com/2013/01/21/que-es-gitgithub/

Rouse , M .(2017). What is Docker?. Recuperado de: https://searchitoperations.techtarget.com/definition/Docker.

Sobers, R. (2018) What is OAuth? Definition and how it works. Recuperado de https://www.varonis.com/blog/what-is-oauth/

Ugarte, J.J. , Santelices, V , Catalán, X , García, F , Burgui, J , Bonilla, E , Cabrera, K , Del Mastro, C , Martínez, N , Medrano, J , Ojeda , U , Pérez, Y , Rodríguez, M , Saavedra, R

Velázquez, R., Pérez, M. y Ortega, L. (2009) . Perfeccionamiento de los procesos universitarios. Un acercamiento a las herramientas de gestión. Revista académica semestral.

Welsh M., Gurnell D. (2016). Advanced Scala with Cats. Birghton, UK. Underscore.

Zegarra, J , Muga, A (2016). Calidad de la formación Universitaria Información para la toma de decisiones.