ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

DESARROLLO DE UNA APLICACIÓN WEB DE PLANIFICACIÓN ACADÉMICA PARA LA FACULTAD DE INGENIERÍA DE SISTEMAS

Manejo de datos ingresados por el usuario y validación/aprobación de horarios generados

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO/A EN SOFTWARE

DIANA STEFANÍA LÓPEZ PRIKHDOKO

diana.lopez03@epn.edu.ec

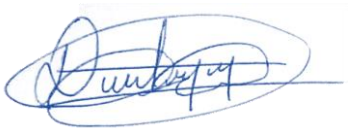
DIRECTOR: CARLOS EDUARDO ANCHUNDIA VALENCIA

carlos.anchundia@epn.edu.ec

DMQ, septiembre 2022

CERTIFICACIONES

Yo, Diana Stefanía López Prikhodko declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.



Diana Stefanía López Prikhodko

Certifico que el presente trabajo de integración curricular fue desarrollado por Diana Stefanía López Prikhodko, bajo mi supervisión.

Carlos Eduardo Anchundia Valencia
DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

Diana Stefanía López Prikhodko

Carlos Eduardo Anchundia Valencia

Mahatma Israel Quijano Zanipattini

Luis Alejandro Llanganate Valencia

DEDICATORIA

Dedico el presente trabajo de integración curricular a los futuros estudiantes de la Facultad de Ingeniería de Sistemas de la Escuela Politécnica Nacional, los cuales van a beneficiarse viviendo un proceso de matriculación ágil y efectivo.

AGRADECIMIENTO

Agradezco a mis padres Lidia Nikolaevna Prikhodko y Luis Eduardo López Vinueza por su amor y apoyo incondicional durante toda mi formación académica y personal.

Agradezco a Mahatma Israel Quijano Zanipattini y Luis Alejandro Llanganate Valencia por formar parte del equipo de desarrollo y siempre estar dispuestos a ayudar con todos sus conocimientos y experiencia.

Agradezco a Carlos Eduardo Anchundia Valencia por haber sido un excelente guía a lo largo del desarrollo del presente trabajo.

Agradezco a las autoridades de la Facultad de Sistemas de la Escuela Politécnica Nacional, quienes entregaron un espacio de diálogo activo y recursos necesarios para poder ayudar al mejoramiento del proceso de planificación académica.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

La Facultad de Ingeniería de Sistemas de la Escuela Politécnica Nacional (FIS-EPN) ofrece a la comunidad conocimientos actualizados en tecnología e impulsan proyectos de innovación en diferentes ámbitos. Debido al aumento sustancial de estudiantes que ingresan a las diferentes carreras ofertadas, la planificación académica se vuelve un reto constante para las autoridades. A pesar de que actualmente se maneja un sistema de escritorio llamado FET, este no provee todas las facilidades para disminuir la complejidad del proceso. Como solución se propuso el desarrollo de un sistema web multiusuarios para la planificación académica basándose en lo provisto en el sistema de escritorio y el uso de su motor de generación de horarios. En el proyecto se aplicaron se aplicaron el marco de trabajo Scrum para la gestión del proyecto y el conjunto de buenas prácticas BDD para guiar el desarrollo, enfocándose en el valor que necesitan las autoridades. El proyecto se dividió en tres componentes enfocándose en diferentes aristas del proceso. Para el componente “Manejo de datos ingresados por el usuario y validación/aprobación de horarios generados” se enfocó en dos características: integridad de datos y visualización de horarios. Como resultado, la primera característica logró disminuir problemas de duplicación y perdida de datos; mientras que, la segunda característica entregó una pantalla intuitiva usando la librería Material Angular UI y permitiendo filtrar los horarios en base a las necesidades indicadas por los involucrados.

PALABRAS CLAVE: Planificación académica, horarios, BDD, FET, Cucumber, docentes, asignaturas, carreras.

ABSTRACT

The Facultad de Ingeniería de Sistemas de la Escuela Politécnica Nacional offers the community up-to-date knowledge in technology and promotes innovation projects in different fields. Due to the substantial increase in students entering the different careers offered, academic planning becomes a constant challenge for the authorities. Although the desktop FET schedule generation system is used but it does not provide all the facilities to reduce the complexity of the process. As a solution, we develop a multi-user web system for academic planning based on what was provided in the desktop system and the use of its timetable generation engine. In this project we apply Scrum framework for project management and the BDD good practices to guide the development process, focusing on the value needed by the authority. The project was divided into three components focusing on different edges of the process. For the "Management of data entered by the user and validation/approval of generated timetables" component, two features were focused: data integrity and display of schedules. As a result, the first feature managed to reduce data duplication and loss problems; while the second feature provided an intuitive screen using the Material Angular UI library and allowing the timetables to be filtered based on the needs indicated by those involved.

KEYWORDS: Academic planning, timetable, BDD, FET, Cucumber, teachers, subjects, careers.

1 INTRODUCCIÓN

1.1 Descripción del proyecto

Según el Estatuto de la Escuela Politécnica Nacional, el subdecano de cada Unidad Académica es responsable de coordinar las actividades de docencia. Una de sus actividades principales es la creación de la planificación académica de cada una de las carreras vigentes. Esta planificación debe ser aprobada por parte de los consejos de los departamentos adscritos; sirviendo como un insumo para el proceso de matriculación [1].

Según [2], un sistema de planificación académica es un aplicativo que permite el ingreso de datos relacionados a las asignaturas, docentes, niveles, espacios físicos, carreras y restricciones. Esto permite generar los horarios para cada uno de los niveles académicos de las carreras vigentes. Actualmente, el proceso de planificación académica de la Facultad de Ingeniería de Sistemas se lo hace por medio del software “FET”, es un generador automático de horarios que usa un algoritmo eficiente y veloz [3]. No obstante, FET dificulta la colaboración entre las varias personas que gestionan información académica.

Siendo así, en el trabajo realizado en [2] se buscó mitigar este problema desarrollando un aplicativo web. Sin embargo, la solución tiene varios problemas relacionados a la complejidad de su uso (solo funciona para casos específicos e ideales), produciendo que los involucrados en el proceso de planificación académica no utilicen dicho sistema (Anexo III).

Para ello, se propuso desarrollar un sistema web basándose en el proceso de planificación académica actual. El mismo se encuentra dividido en 3 componentes. El componente respectivo del presente trabajo de integración curricular se encuentra marcado dentro de la Figura 1.

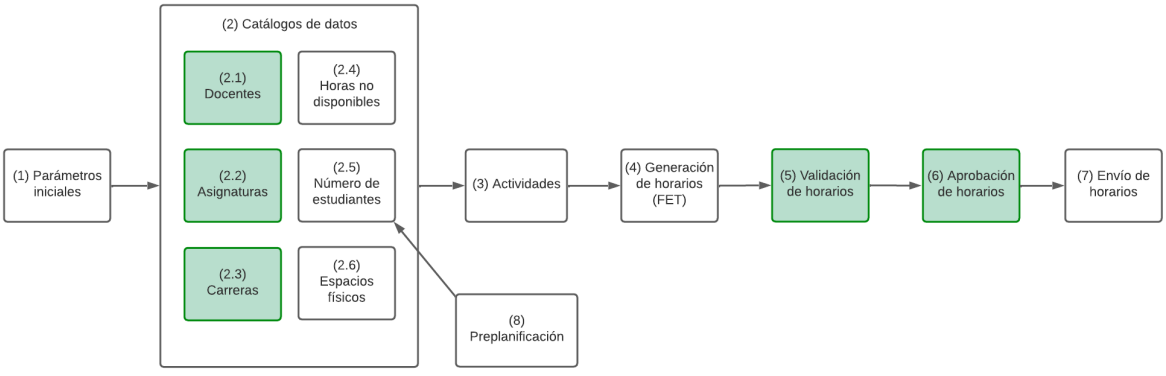


Figura 1. Diagrama del proceso del proyecto general.

1.2 Descripción del componente

El componente “Manejo de datos ingresados por el usuario y validación/aprobación de horarios generados” se relaciona a los pasos: (2.1) Docentes, (2.2) Asignaturas, (2.3) Carreras, (5) Validación de horarios y (6) Aprobación de horarios; mostrados en la Figura 1. Cada uno de estos pasos son necesarios dentro de la planificación académica, dado que se necesita tener un conjunto de datos a ser analizados y la validación/aprobación de los horarios resultantes.

La actividad de “manejo de datos ingresados por el usuario” involucra a los coordinadores de las carreras vigentes, puesto que son los encargados del ingreso de datos previo a la generación de horarios. Estos datos se relacionan a los docentes con contrato activo, las carreras y las asignaturas vigentes (explicado en la sección 4.2). Por ello, estos datos deben ser ingresados con mucho cuidado, puesto que pueden caer en duplicación, errores de tipeo o perdida de la integridad [4]. Siendo así, la solución propuesta contemplará la creación de un conjunto de interfaces (divididas en módulos) que permita el ingreso de los diferentes grupos de datos relacionados a los docentes, asignaturas y carreras.

En cuanto a la actividad de “validación de horarios generados” se busca entregar una pantalla que muestre de manera amigable toda la información de cada horario que fue generado por el motor FET con la información ingresada por parte de los coordinadores. No obstante, esta visualización de información debe enfocarse a las necesidades de los coordinadores y subdecano, con filtros relacionados a docentes y grupos, por nivel académico de cada carrera vigente. Para que los coordinadores y subdecano (bajo su experiencia) analicen las horas asignadas a cada docente y su distribución de asignaturas.

Por último, la actividad de “aprobación de los horarios generados” se hará por medio de culminación de la planificación del semestre actual. Esta aprobación la realizará el subdecano y producirá la inhabilitación de la opción de generación de horarios hasta que se inicie la siguiente planificación semestral. Además, esto permitirá controlar que los datos críticos (horas no disponibles y actividades) no sean alterados fuera de las fechas permitidas.

Por lo expuesto previamente, el trabajo presente tiene como objetivo desarrollar un sistema web que permita de forma intuitiva y fácil el ingreso de datos. Asimismo, poder tener un proceso de validación y/o aprobación de los horarios generados por el motor FET, previo al ingreso de estos en el sistema de matriculación usado en la Escuela Politécnica Nacional.

1.3 Objetivo general

Contar con un sistema web intuitivo que permita agilizar el proceso de planificación académica en la Facultad de Ingeniería de Sistemas para asegurar la integridad de los datos ingresados y facilitar la validación/aprobación de los horarios generados.

1.4 Objetivos específicos

1. Entregar un sistema web que evite problemas de inserción de datos duplicados o incorrectos en los módulos de profesores, asignaturas, niveles y carreras, de tal manera que los usuarios finales puedan generar horarios para los diferentes niveles de las carreras disponibles en la facultad de Ingeniería de Sistemas sin ningún problema de integridad de datos.
2. Validar y aprobar los horarios generados dentro del sistema web con la intervención de los diferentes usuarios que se encargan de dichos procesos (coordinadores de carrera y subdecano), asegurando así la precisión de la información que va a ser entregada a los estudiantes y docentes durante el proceso de matrículas.

1.5 Alcance

El proyecto tiene como alcance entregar un sistema web de planificación académica basado en las funcionalidades que entrega el sistema FET de escritorio y son utilizados actualmente por parte de los coordinadores y subdecano de la Facultad de Ingeniería de Sistemas. Por ello, se creará los diferentes módulos relacionados a los procesos presentados en la Figura 1.

Para el caso particular del presente componente, se buscará entregar módulos que faciliten el ingreso de los datos de docentes, asignaturas y carreras. En donde se va a considerar el control de datos duplicados, la integridad y un diseño bajo librerías de diseño de interfaces gráficas. De igual forma, se llevará a cabo la implementación de un módulo que permita visualizar los datos obtenidos del proceso de generación de horarios. Siendo prioridad permitir el filtrado en base a las funcionalidades actuales del sistema FET de escritorio.

Por otra parte, se debe considerar que la actividad de aprobación se lo sugirió para completar una parte del ciclo de proceso de planificación académica. No obstante, esta podría no llegarse a realizar dependiendo de lo indicado en las futuras reuniones con los *stakeholders* o un cambio de enfoque de esta (posterior a un análisis con todos los involucrados). Asimismo, no se realizará la migración de los datos almacenados en los varios archivos generados en los últimos semestres y no se contempla la integración con herramientas que se encuentre utilizando la Facultad de Ingeniería de Sistemas.

Este proyecto se lo desplegará dentro de los servidores de la Facultad de Ingeniería de Sistemas (FIS), los cuales van a ser proporcionados durante el proceso de desarrollo del sistema web. Sin embargo, se lo realizaría al momento de finalizar la programación e integración con los tres componentes que la conforman.

2 MARCO TEÓRICO

En esta sección se detallan los temas investigados y utilizados durante el desarrollo del componente del sistema de planificación académica.

2.1 Planificación académica

El proceso de planificación académica dependiendo de la institución educativa varia su concepto y aplicación. Esto se debe a la cantidad de estudiantes y actividades que desempeña la universidad. En algunos casos, las instituciones llegan a abarcar múltiples procesos tales como [5]: inscripción de estudiantes, actualización de calificaciones, control de asistencia del personal y estudiantes, planificación de horarios, emisión de boletas de calificaciones, entre otros. Si bien se debe considerar todo lo anterior mencionado para poder llevar a cabo un ciclo de estudio para un conjunto de estudiantes, no siempre es lo más acertado.

Varias instituciones manejan algunos de sus procesos de manera digital en sistemas previamente desarrollados, siendo así innecesario considerar todos los procesos. Por otro lado, en los últimos años la sociedad promueve la reducción de desechos y el aprovechamiento de los recursos. Llevando a las instituciones de educación promover la reducción del consumo de papel, concursos de reciclaje y educación del tratamiento de residuos [6]. Además, el uso de procesos manuales lleva a la perdida de información, falta de manejo de historiales y cortos tiempos de respuesta [4]. Por esta razón, algunas instituciones deben migrar sus procesos a medios digitales, ya que es la solución óptima.

En cuanto a la calidad de los sistemas de planificación, según [4] la calidad debe enfocarse en tres atributos indispensables y obligatorios, los cuales son: funcionalidad, usabilidad y portabilidad. En el caso de los dos primeros atributos se debe considerar en casi cualquier sistema, puesto que si no cumple el objetivo para el cual fue hecho (ya sea funcionalidad o dificultad de uso), el sistema es un fracaso. Pero en cuanto a la portabilidad se pudo analizar que en el caso de [4] es importante, ya que los profesores y autoridades no solo van a manipular el sistema en computadoras de escritorio.

Finalmente, en la Facultad de Ingeniería en Sistemas no se aplican todos los procesos y aspectos de calidad que se pudo analizar de otros sistemas desarrollado. Esto se debe a que la Escuela Politécnica Nacional posee varios sistemas enfocados en varios de estos procesos (Anexo III). Además, el objetivo del sistema únicamente se va a centrar en la generación de horarios en un momento dado del periodo académico.

2.2 Planificación académica en la Facultad de Ingeniería de Sistemas (FIS)

La Facultad de Ingeniería de Sistemas de Escuela Politécnica Nacional actualmente define el proceso de planificación académica como la generación de horarios para las diferentes carreras de pregrado y posgrado que oferta. El proceso de planificación inicia al finalizar el primer bimestre del semestre en curso, puesto que los coordinadores analizan la cantidad de estudiantes que se encuentran cursando las diferentes asignaturas y realizan una aproximación para obtener la cantidad de grupos/paralelos que van a ser necesarios. Asimismo, realizan una encuesta a los estudiantes para ratificar los resultados obtenidos (Anexo III).

Con los valores obtenidos, los coordinadores comienzan el proceso de carga de datos al software FET de manera secuencial (un coordinador a la vez), llegando a generar decenas de versiones que son compartidas entre todos los involucrados. Los datos que se deben tomar en cuenta son:

- **Docentes con contrato vigente:** Personas con una formación académica de mínima de tercer nivel y con un contrato a tiempo completo u ocasional para impartir clases el siguiente semestre (ciclo de clases).
- **Carreras:** Se refiere al conjunto de asignaturas relacionadas que son impartidas de manera virtual, presencial o dual.
- **Asignaturas:** Se refiere a las materias que forman parte de cada uno de los planes de estudios de las carreras impartidas.
- **Espacios físicos:** Se refieren a las aulas, laboratorios o espacios asignados por la universidad para la impartición de clases u actividades educativas.
- **Número de estudiantes:** Se refiere al número de estudiantes que va a cursar una asignatura en específico.
- **Grupos:** Se refiere a la cantidad y nombres de paralelos que se debe generar para cada asignatura de una carrera específica y se lo obtiene en base al número de estudiantes.
- **Restricciones:** Se refieren a las limitaciones de espacios físicos (asignación a otras actividades académicas o de investigación), las horas de no disponibilidad de los docentes (actividades personales, académicas o de otra índole corroboradas por

jefatura de departamento) y asignaturas dictadas por otros departamentos o facultades).

No obstante, este proceso finaliza un mes después de la culminación del semestre, debido a la naturaleza cambiante de varios datos, tal como: restricciones de espacios físicos, horas de disponibilidad de los docentes y asignaturas dictadas por otros departamentos.

Como consecuencia, en varios periodos académicos se deben modificar los horarios de varios niveles durante el proceso de matrículas (dos semanas antes del inicio de clases). Esto produce inconformidad a los docentes y estudiantes, ya que deben esperar hasta los primeros días de clases para saber su horario de clases final para el semestre entrante.

2.3 FET

FET es un software libre de escritorio creado en el año 2003, el cual permite la generación de horarios para las diferentes instituciones de educación [7]. Esta aplicación permite el ingreso de datos como asignaturas, docentes, restricciones, espacios físicos, edificios, estudiantes, actividades, subactividades, restricciones, grupos, entre otros. Dependiendo de la necesidad de la institución, varias opciones no son utilizadas. Por ejemplo, la Facultad de Ingeniería de Sistemas no hace uso de subactividades debido a que no necesita una clasificación tan específica para los subgrupos que puedan generarse en una misma asignatura.

El software FET entrega una carpeta con los resultados de la generación de horarios, donde muestra en hojas HTML o archivos XML los horarios clasificados por docente y grupos (los archivos más relevantes). Si se lo visualiza dentro de la aplicación, se puede observar reportes con cantidad de horas asignadas a un docente o la carga horaria de un grupo en un semestre específico.

Como desventaja, se debe guardar regularmente todos los cambios que se vayan incorporando de forma manual, ya que la aplicación no lo hace automáticamente y se puede llegar a perder los resultados obtenidos [7]. Adicionalmente, las personas que desean utilizar el software FET deben leer y comprender toda la documentación que incluye, dado que entrega una gran cantidad de herramientas y opciones que en un inicio pueden ser abrumadoras. Asimismo, el trabajo colaborativo es imposible debido a la naturaleza del FET, pero tiene un motor que puede ser incluido en sistemas web (impulsando el desarrollo del proyecto presente).

2.4 Behavior-Driven Development (BDD)

BDD es un conjunto de buenas prácticas de ingeniería de software (basada en metodologías ágiles) que permite a los equipos de desarrollo poder entregar porciones de software valioso con un grado alto de calidad y en tiempos reducidos [8]. Esto se debe al uso de lenguaje común para generar oraciones simples que posteriormente se estructuran en requerimientos de alta prioridad y así facilita la comunicación de todas las partes interesadas [8].

Las actividades de BDD son posibles gracias al manejo de un lenguaje sencillo y al análisis previo de las metas del negocio y sus necesidades; permitiendo obtener una mejor conexión y entendimiento de los objetivos buscados por cada parte interesada. Posteriormente, se generan las características específicas que necesitan que el software cumpla para llegar a satisfacer sus objetivos. A pesar de tener una mejor comprensión del negocio, es necesario definir ejemplos de cada característica, es decir la explicación con datos reales y resultados esperados. El siguiente paso es generar el código necesario para poder ejecutar estas características y poder validar el cumplimiento de lo requerido [8]. Lo mencionado previamente, se lo conoce como documentación viva (o del inglés, *living documentation*), ya que nos provee de información actualizada y fácil de entender en cualquier momento del ciclo de desarrollo de software [13]. Finalmente, los desarrolladores pueden iniciar con la programación del software.

En la Figura 2 se muestra un resumen de las actividades y resultados que entrega la aplicación de BDD:

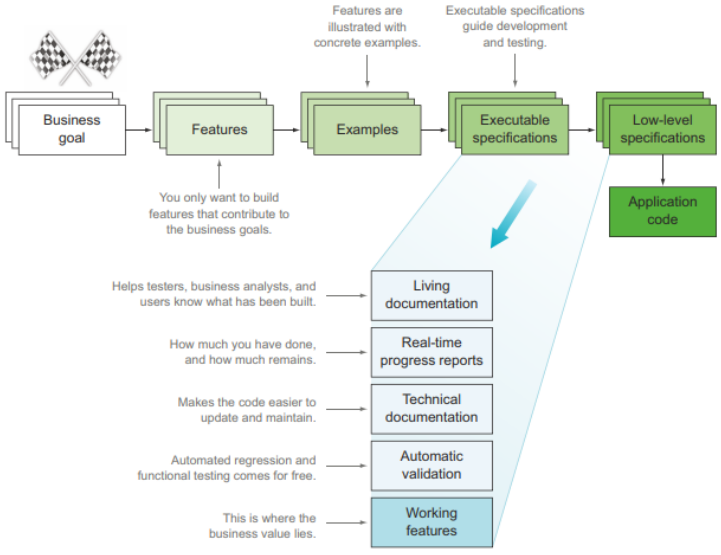


Figura 2. Principales actividades y resultados de BDD. Obtenido de *[8]*.

Con lo anterior visto, los desarrolladores tendrán en un inicio una prueba que se encuentra fallando; volviéndose su objetivo generar el código necesario para que esta pueda pasar (generando la característica de valor para el usuario). Al final, el desarrollador debe realizar una inspección en su código y mejorar/ordenar de tal manera que pueda ser comprendido por otro los otros miembros del equipo [8]. Pese a todo, puede existir cambios que afecten a las pruebas realizadas y por ello, este proceso se vuelve cíclico. En la Figura 3 se presenta de manera gráfica lo explicado previamente:

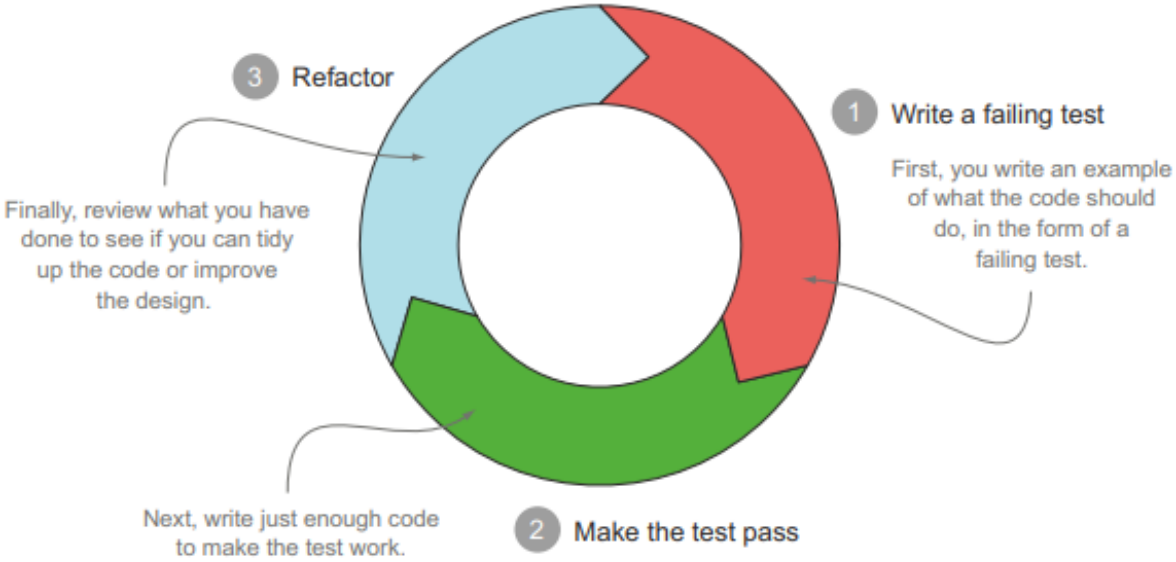


Figura 3. Ciclo de 3 fases de BDD. Obtenido de *[8]*.

2.5 Cucumber

Cucumber es una herramienta BDD para automatizar una característica ejecutable y se lo realiza escribiendo código de prueba de cada paso por medio de un DSL (Lenguaje Específico de Dominio). El DSL que *Cucumber* utiliza se lo conoce como *Gherkin*, el cual maneja una estructura generada por varios elementos basados en las características, comportamientos y acciones [8]. En consecuencia, la aplicación de este lenguaje produce que los desarrolladores puedan entender el negocio para el cual están creando una solución de manera efectiva.

Gherkin entrega elementos que permiten estructurar las pruebas, los cuales son [8]:

1. *Feature*: Representa la característica que se va a probar y se lo relaciona con las historias de usuario, ya que estas entregan el valor al usuario.

2. *Scenario*: Representa a la situación que se va a probar y se lo relaciona con los criterios de aceptación de una historia de usuario.

3. *Scenario Outline:* Permite probar un *scenario* con diferentes datos y salidas. Esto es bastante utilizado, ya que da la posibilidad de probar los diferentes caminos que se puede tomar en una característica.

4. *Example*: Es la notación para indicar los diferentes grupos de datos que van a ser usados dentro de un *Scenario Outline.*

5. *Given, When, Then, And*: Son los elementos que permiten narrar los pasos en lenguaje natural de cada una de las características y a estos elementos, se les atribuye código de prueba para realizar la automatización.

Es importante notar que el uso de *Gherkin* se ha ido expandiendo de tal manera que se puede utilizar en varios idiomas. Esto se lo hace colocando la abreviación de este como etiqueta al inicio del archivo. A continuación, la Figura 4 presenta un pequeño ejemplo aplicando los elementos de *Gherkin*:

Elementos Ghekin

```
# Created by diana at 10/6/2022
# language: es
  Característica: Transferencia bancaria
    Como titular de una cuenta
    deseo realizar una transferencia bancaria segura
    para mantener la confianza en el banco

  Esquema del escenario: Transferencia entre bancos locales
    Dado que Miguel tiene una cuenta de ahorros con $<saldo_origen> dólares de saldo
    Y que Gabriel tienen otra cuenta de ahorro con $<saldo_destino> dólares de saldo
    Cuando Miguel transfiere un monto de $<monto> dólares a Gabriel
    Entonces Miguel tendrá $<saldo_origen_nuevo> dólares de saldo
    Y Gabriel tendrá $<saldo_destino_nuevo> dólares de saldo
    Ejemplos:
      | saldo_origen |  saldo_destino | monto | saldo_origen_nuevo | saldo_destino_nuevo |
      |    100.00    |      10.00     | 50.00 |        50.00       |        60.00        |
      |    150.00    |      70.00     | 70.00 |        80.00       |       140.00        |
      |    10.00     |      10.00     | 10.00 |        0.00        |        20.00        |
```

Figura 4. Ejemplo de una característica usando Gherkin.

3 METODOLOGÍA

En esta sección se detallan las fases que se siguió durante el desarrollo del componente presente. Estas fases inician con la investigación y planificación del proyecto, para posteriormente enfocarse en el desarrollo y ejecución del componente asignado. Se decidió seleccionar el *framework Scrum*, dado que permite organizar todas las ideas anteriores dentro de diferentes ceremonias y conceptos que permiten llegar un mejor control de la gestión del desarrollo [9, 10]. A continuación, se detallará los puntos sobresalientes al aplicarlo en el proyecto:

1. Organizar el desarrollo en cuatro *sprints* de tres semanas, para poder obtener un incremento de valor para el usuario de manera continua. Además, los desarrolladores ejecutan las tareas de planteamiento, diseño, implementación y pruebas dentro de cada uno de los *sprints*.

2. Realizar reuniones cortas indicando las actividades realizadas, actividades a realizar e impedimentos por cada uno de los desarrolladores. Esto permite tener una mejor coordinación por parte del equipo de desarrollo.

Por otro lado, se va a utilizar Cucumber (herramienta basada en las prácticas BDD) para guiar el desarrollo de cada uno de los comportamientos de valor que los clientes esperan del sistema [8]. Esto facilitaría que los desarrolladores comprendan de forma más sencilla la problemática y como se visualizaría dentro del sistema a desarrollar. Adicionalmente, entrega una herramienta que permite automatizar las pruebas, manteniendo la trazabilidad con los comportamiento y ejemplos de estos.

3.1 Fase de investigación de la problemática

Durante esta fase se enfocó en comprender el proceso de planificación académica dentro de la Facultad de Ingeniería de Sistemas (explicando en la sección 2.1 y 2.2) y el funcionamiento del software FET (explicado en la sección 2.3). Para ello, se realizó un conjunto de actividades que van a ser detalladas a continuación:

Investigación de fuentes

En esta fase se realizó una revisión bibliográfica de varias fuentes buscadas en los diferentes repositorios de varias universidades, el cual se encuentra en el Anexo I y Anexo II. Se pudo comprender el gran alcance e impacto que tiene el proceso de planificación académica en las diferentes instituciones. Asimismo, algunas escuelas y colegios

implementan un sistema de planificación por razones sociales (cuidado ambiental) o económicas (evitar desprecios y pérdida de tiempo).

No obstante, al analizar la fuente [2] se pudo observar el significado de este proceso dentro de la Facultad de Ingeniería de Sistemas. Siendo este la necesidad de poder generar horarios de manera efectiva, dado que actualmente existe reprocesos y falta de agilidad en los procesos con el software FET. De la misma manera se analizó que este proyecto [2] no logró cumplir con las expectativas de los interesados, llevándolo a su desuso.

Entrevista a los interesados

A lo largo de la planificación del proyecto se pudo realizar tres entrevistas a los diferentes involucrados (Anexo III). La primera entrevista fue realizada por Carlos Eduardo Anchundia Valencia (director del proyecto) a los tres principales *stakeholders*: Julio César Sandobalín Guamán (coordinador de la carrera de software), Josafá de Jesús Aguiar Pontes (coordinador de la carrera de computación) y José Francisco Lucio Naranjo (subdecano de la facultad). En donde se buscó explicar el funcionamiento del software FET y las desventajas que este tiene. Por otro lado, se pudo visualizar los diferentes módulos que son utilizados en gran medida y los resultados obtenidos con la herramienta.

Con toda la información recolectada se pudo visualizar una mejor delimitación del resultado esperado del sistema. No obstante, se necesitó asegurar varios puntos que en la primera entrevista no fueron explicados detalladamente y definir los roles de cada uno de los involucrados. Para ello, se realizó una segunda entrevista donde los entrevistadores fueron los tres miembros del equipo de desarrollo hacia el subdecano José Francisco Lucio Naranjo. Al finalizar la reunión, se establecieron los siguientes roles en base a la experiencia y disponibilidad de cada interesado:

- *Product Owner:* José Francisco Lucio Naranjo
- *Scrum Master:* Luis Alejandro Llanganate Valencia
- *Scrum Development Team*: Diana Stefanía López Prikhodko y Mahatma Israel Quijano Zanipattini

Finalmente, la tercera entrevista fue realizada debido a la necesidad de comprender varios aspectos técnicos y de funcionalidad que se entregó en [2]. Los entrevistadores fueron los tres miembros del equipo hacia Alex Javier Ulloa Arévalo y se concluyó que el desarrollo del sistema debe plantearse desde cero. Esto se debe a que el producto entregado en [2] no tiene un funcionamiento correcto y se utilizaron tecnologías no adecuadas para su uso.

Sin embargo, se va a reutilizar el servicio FET que generó en la nube, dado que esto facilitaría en gran medida la generación del producto esperado.

Selección de herramientas

Para poder tener una mejor organización de la información recolectada y la documentación de las tareas de desarrollo del sistema, se va a usar el software *Notion*. Dicha herramienta permite gestionar proyectos de software sin definir un conjunto de reglas (obligadas por una metodología o marco de trabajo), ya que el equipo es el encargado de generar la mejor estructura de hojas para mostrar la información en base a la metodología escogida. El acceso a la misma se encuentra ubicado en el Anexo IV.

Por otro lado, para el manejo del desarrollo del proyecto se va a utilizar el lenguaje *NestJS* y la librería *TypeORM* para el desarrollo del *backend*. En cuanto a la documentación de la API se lo hará usando la herramienta llamada *Swagger*. Por último, para el *frontend* se usará el *framework Angular* (desarrollado en *Typescript*). Asimismo, se utilizará las librerías de *Angular Material UI* y *Swal* (mensajes en pantalla personalizados).

3.2 Fase de diseño

En esta fase se inició con el proceso de planificación y diseño del sistema en base a toda la información recolectada. Se inició con el planteamiento de la arquitectura base, seguido de la generación de las historias de usuario *(product backlog)*, la división del proyecto en componentes y la aplicación de BDD.

Diseño de la arquitectura base

En conjunto con el equipo de desarrollo se planteó utilizar un servidor *Ubuntu Server 20.0* para el despliegue del aplicativo. Este servidor tendrá un mínimo de memoria de 4GB y un espacio de al menos 50GB de disco *SSD* o 100GB en disco *HDD*. Por otro lado, se instalará un *WebServer Nginx_v1.21.4* para que actúe como proxy redireccionando el tráfico entre el *frontend*, la API, el *backend* y el cliente del motor FET. Además, se manejará una base de datos en la nube *PostgreSQL_12* del servicio de *Azure*. En la Figura 5, se muestra el diagrama de la arquitectura propuesta por el equipo de desarrollo.

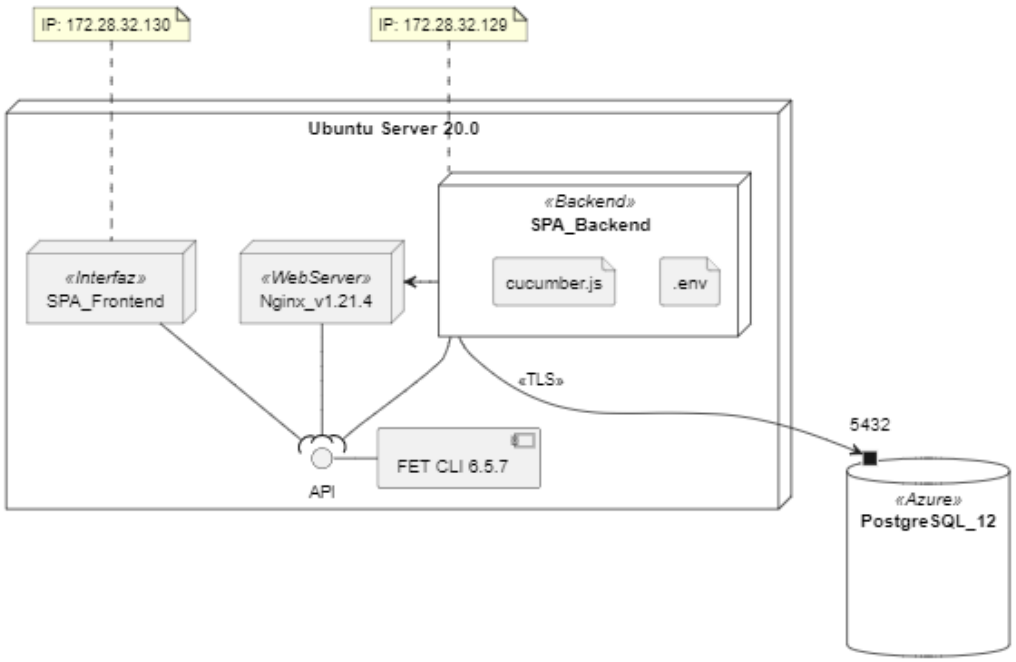


Figura 5. Arquitectura propuesta.

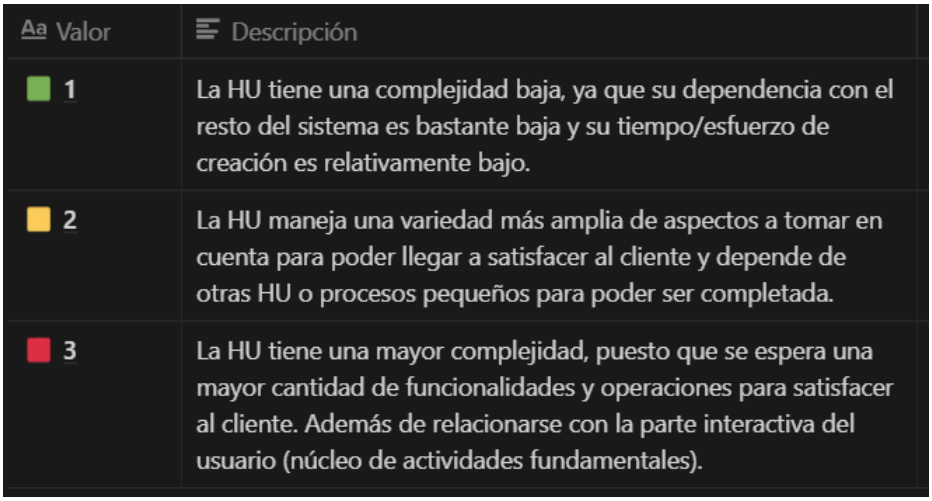
Análisis de requerimientos

1. Historias de usuario

Las entrevistas y la literatura permitieron recabar las necesidades de los interesados. Con esta información planteamos las posibles características que debe posee el sistema web a través de la especificación de un product backlog con historias de usuario y tareas que fueron gestionadas a través de *Notion*, disponible en Anexo IV. En este componente se establecieron las siguientes historias o características de valor: el ingreso de datos, la generación/visualización de horarios y la aprobación/difusión de horarios.

2. Estimación y priorización

La siguiente actividad realizada fue la estimación usando la técnica del *Planning Poker*, la cual permitió obtener la opinión de cada uno de los miembros del equipo de desarrollo y llegar a un acuerdo. Pero se realizó dos sesiones para esta actividad, dado que, en la primera reunión, las estimaciones no siguieron ningún rango o regla definida (obteniendo estimaciones que no pueden ser clasificadas). En la Figura 6 se presenta la tabla planteada con los valores de estimación y su descripción respectiva:



Valor	Descripción
1	La HU tiene una complejidad baja, ya que su dependencia con el resto del sistema es bastante baja y su tiempo/esfuerzo de creación es relativamente bajo.
2	La HU maneja una variedad más amplia de aspectos a tomar en cuenta para poder llegar a satisfacer al cliente y depende de otras HU o procesos pequeños para poder ser completada.
3	La HU tiene una mayor complejidad, puesto que se espera una mayor cantidad de funcionalidades y operaciones para satisfacer al cliente. Además de relacionarse con la parte interactiva del usuario (núcleo de actividades fundamentales).

Figura 6. Valores de estimación para las historias de usuario.

Finalizado el proceso de revisión, se generó el *product backlog* planteado para el proyecto y ubicado en el Anexo VII. En la siguiente sección, se va a explicar las historias asignadas para la realización del componente.

Planificación de las historias de usuario

En esta fase se realizó un análisis de complejidad de las historias de usuario para encontrar las tres principales y así ubicarlas en distintos componentes. Siendo así, se denotó que las historias 2, 8 y 12 cumplen este criterio. Luego se trató de realizar la mejor distribución en base a su estimación. Este proceso permitió llagar a la definición de los componentes y en la Tabla 1 se presenta el correspondiente a este trabajo:

Tabla 1. Distribución de historias de usuario en tres componentes.

	Componente 2
Historias de usuario	5, 6, 7, 8, 9, 10
Desarrollador	Diana López

Luego de la asignación de las historias, se procedió a rotular los títulos con la inicial del nombre cada uno y reiniciar la numeración. De esta forma, cada uno de los desarrolladores podrá diferenciar a simple vista sus historias. A continuación, se muestra el resultado obtenido en la Figura 7:

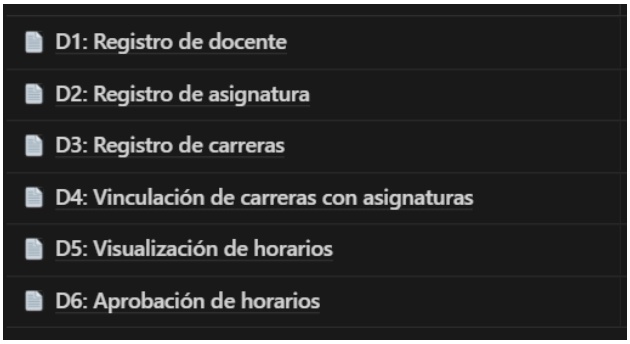


Figura 7. Historias de usuario asignadas al componente 2.

Seguidamente se inició la planificación de los *sprints* con las historias asignadas, en donde se consideró la necesidad de realizar inicialmente la D1, D2, D3 y D4. Eso debido a que son las primeras actividades del proceso denotado en la Figura 1. Obteniendo la planificación mostrada en la Figura 8:

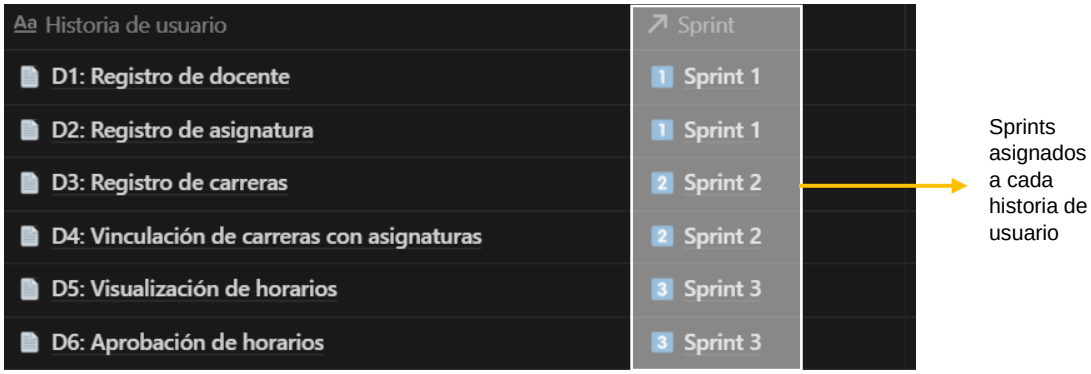


Figura 8. Planificación de *sprints* con las historias de usuario asignadas.

Especificación de escenarios

La siguiente actividad que se realizó es la definición de los escenarios en cada una de las historias. Para lo cual se utilizó la técnica dado, cuando y entonces. Esta es muy utilizada al momento de escribir los escenarios, pues se enfoca la descripción en el comportamiento que se busca obtener. Además, permite tener una sinergia con el proceso metodológico de BDD, ya que esto ayudará más adelante a la generación de los escenarios usando la herramienta *Cucumber*.

Se tuvieron que realizar varias versiones de los escenarios para cada historia, dado que no se lograba demostrar el valor que va a obtener el usuario. Además, no se seguía la estructura de manera correcta. A continuación, se va a mostrar la primera versión de un escenario para la historia D1:

> **Dado** que se tiene la información de un docente de la facultad, **cuando** se ingrese el nombre completo y correo electrónico en un formulario en el módulo de docentes, **entonces** se

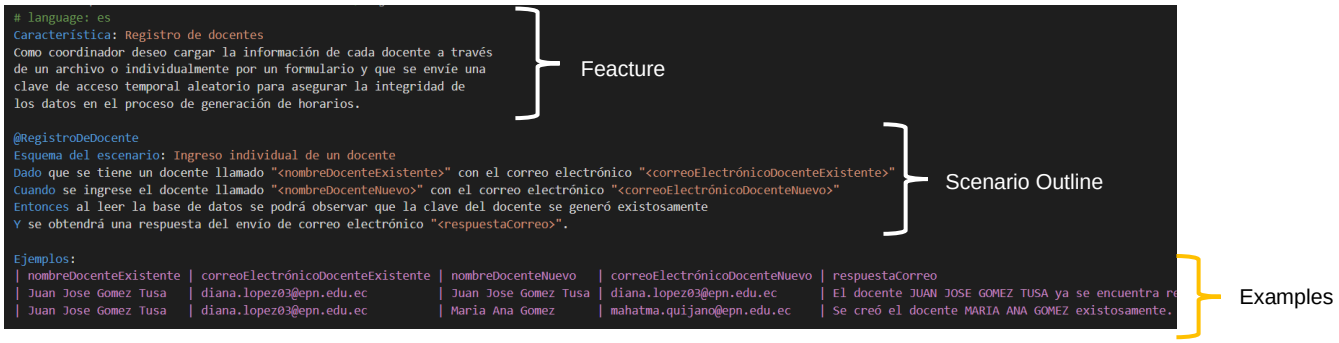
generará un código de acceso temporal (generada aleatoriamente); se añadirá en la base de datos estos 3 datos y se podrá visualizar dentro del módulo de docentes.

Como se puede observar, el ***dado*** habla de manera genérica del docente y que este no necesariamente se encuentra registrado en el sistema (preámbulo o precondición de la prueba). Asimismo, el ***cuando*** indica una precondición, lo cual es incorrecto. En esta sección se debería indicar la acción que va a ser ejecuta. Finalmente, el ***entonces*** no entrega valor al usuario (indicar sobre los registros de la base de datos no es algo relevante) y a su vez, la oración demuestra actividades que no pueden ser probadas con *Cucumber* (interfaz gráfica).

Ahora bien, con todas estas correcciones y observaciones que fueron sugeridas se obtuvo el siguiente escenario:

> **Dado** que se tiene un docente existente **cuando** se ingrese el docente con un nombre y correo electrónico **entonces** al leer la base de datos se podrá observar que la clave del docente se generó exitosamente **Y** se obtendrá una respuesta del envío de correo electrónico.

La nueva versión del escenario denota la existencia de un docente, los datos necesarios para crear uno nuevo y dos resultados de valor (la generación de su clave de acceso y envió por correo electrónico). Por último, se puede tener dos casos bien marcados: el docente existe previamente o el docente no existe. En estos casos se hace uso del escenario planteado y se lo transforma a un *Scenario Outline* con *Examples* (visto en la sección 2.5). Para el escenario analizado, se puede ver transformación en la Figura 9.




```
# language: es
Característica: Registro de docentes
Como coordinador deseo cargar la información de cada docente a través
de un archivo o individualmente por un formulario y que se envíe una
clave de acceso temporal aleatorio para asegurar la integridad de
los datos en el proceso de generación de horarios.

@RegistroDeDocente
Esquema del escenario: Ingreso individual de un docente
Dado que se tiene un docente llamado "<nombreDocenteExistente>" con el correo electrónico "<correoElectrónicoDocenteExistente>"
Cuando se ingrese el docente llamado "<nombreDocenteNuevo>" con el correo electrónico "<correoElectrónicoDocenteNuevo>"
Entonces al leer la base de datos se podrá observar que la clave del docente se generó existosamente
Y se obtendrá una respuesta del envío de correo electrónico "<respuestaCorreo>".

Ejemplos:
| nombreDocenteExistente | correoElectrónicoDocenteExistente | nombreDocenteNuevo   | correoElectrónicoDocenteNuevo | respuestaCorreo
| Juan Jose Gomez Tusa   | diana.lopez03@epn.edu.ec          | Juan Jose Gomez Tusa | diana.lopez03@epn.edu.ec      | El docente JUAN JOSE GOMEZ TUSA ya se encuentra re
| Juan Jose Gomez Tusa   | diana.lopez03@epn.edu.ec          | Maria Ana Gomez      | mahatma.quijano@epn.edu.ec    | Se creó el docente MARIA ANA GOMEZ existosamente.
```

Figura 9. Esquema de escenario con sus dos ejemplos.

Este proceso de corrección y generación del *Scenario Outline* se lo realizó para cada una de las historias planteadas para el componente. No obstante, en las historias D4 y D6 no se llegó hasta la colocación del escenario en los archivos de pruebas del código. Esto se debe a que la historia D4 sería conveniente actualizar el escenario plateado en la historia

D2, ya que añade nuevas características de mejora. Mientras que la historia D6 se encuentra en análisis por parte de todos los involucrados.

3.3 Fase de desarrollo

En esta fase se comenzó con el desarrollo del componente, en el que se inició con la programación de la prueba respectiva a la característica o historia de usuario planteada. Posteriormente, se generó el código para pasar la prueba y luego la implementación de la interfaz gráfica. A pesar de que se planificó realizar pequeños entregables de valor en cada *sprint*, no se pudo cumplir esto durante el *sprint* 1 y 2. Esto se debió a la gran carga académica que se tenía y al tiempo de aprendizaje de las herramientas y/o tecnologías usadas para la implementación del componente.

Se realizó en equipo una nueva planificación de las historias que van a ser entregadas en los dos *sprints* restantes. Para el presente componente se concluyó lo siguiente:

- La historia D4 es una mejora que se pensó para facilitar el ingreso de datos en la pantalla de actividades. Sin embargo, no es algo crítico y se puede mantener la forma actual que lo manejan en el programa de escritorio (evitando desperdicio de tiempo en aprendizaje y alterando en gran manera el proceso actual de planificación académica).
- La historia D6 es una mejora para poder tener un fin formal de la planificación académica, ya que actualmente los involucrados dan por concluida cuando los horarios logran satisfacer las necesidades del semestre entrante. Así que al definirlo dentro de un intervalo de tiempo puede producir el desuso de esta funcionalidad, puesto que existe planificaciones semestrales que culminan en los primeros días de clases.

Con todo lo anterior, la nueva planificación de los *sprints* se puede visualizar en la Figura 10.

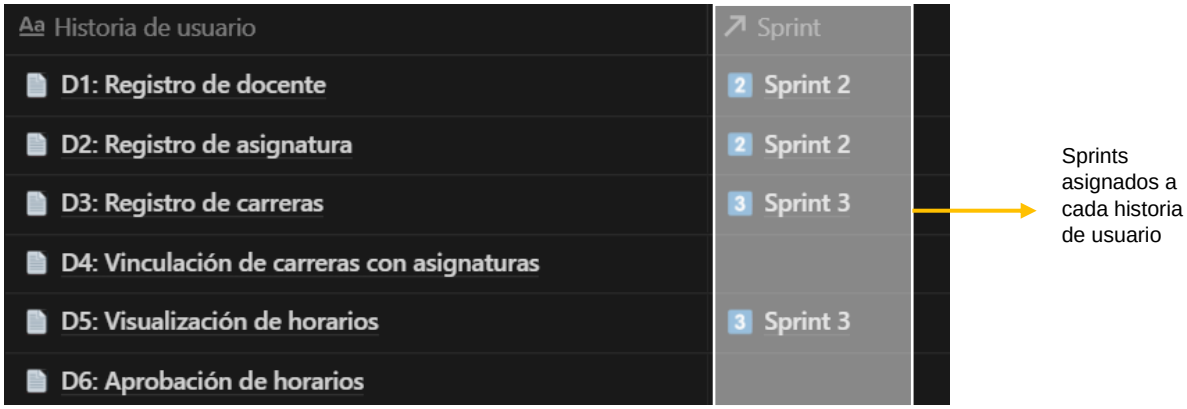


Figura 10. Nueva planificación *de sprints* con las historias de usuario asignadas.

Programación de la prueba

Para ejecutar los escenarios planteados para cada historia de usuario o característica se lo realiza por medio del comando:

```
npm test .\features\nombreCaracteristica.feature.
```

Sin embargo, se debe considerar el lenguaje de programación utilizado, ya que puede generar problemas si no se incorpora algún archivo específico. En este proyecto se planteó utilizar Nestjs, obligando a generar un archivo conocido como: ***beforeStep_afetStep.ts***. Este archivo activará el puerto de la aplicación para que durante la ejecución de pruebas se pueda acceder a API y al finalizar, se apagué la aplicación. En la Figura 11, se puede ver de manera breve la aplicación del *beforeStep* y *afterStep*:

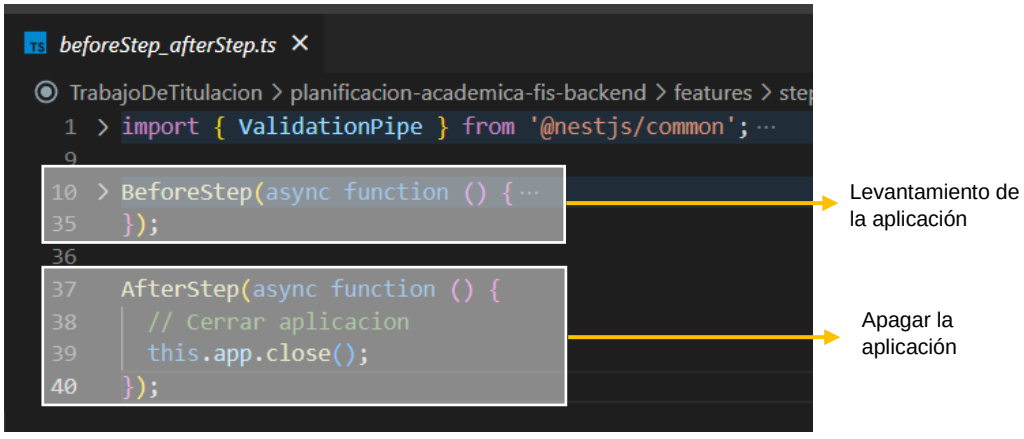


Figura 11. Archivo beforeStep-afterStep para el arranque de las pruebas.

Ahora bien, se debe ejecutar el comando mencionado anteriormente y en la consola nos va a mostrar que se debe generar un archivo de definición de pasos. Este debe tener el mismo nombre que el archivo de su característica (únicamente se cambia la extensión).

Por otro lado, se debe importar los elementos del lenguaje de *Gherkin* explicado en la sección 2.5. De la misma forma, se recomienda la estructuración de las carpetas como se presenta en la Figura 12.

Figura 12. Estructura de archivos de prueba.

Posteriormente, muestra la estructura que se debe generar en relación con los pasos redactados en los escenarios (usando los elementos *Given*, *When*, *Then* y *After*). En donde se reemplaza el retorno pendiente con el código que permitirá la automatización de las pruebas. Además, se debe considerar modificar estas funciones entregadas como funciones asíncronas, dado que el lenguaje usado maneja consultas con Promesas.

Otro aspecto importante es la colocación de un *timeout* mayor a 25000. Muchas veces, el acceso a la base de datos pude tomar un mayor rango de tiempo, produciendo que la prueba falle por el valor predeterminado de 5000 que maneja *Cucumber*. En la Figura 13, se muestra un ejemplo de un paso del escenario de docentes:

Figura 13. Ejemplo de un elemento "When" implementado.

En este punto, al ejecutar las pruebas todas deben fallar (como se explicó en la sección 2.4), dando paso a la fase de desarrollo del componente.

Implementación de cada historia de usuario

En esta sección se va a explicar los objetivos planteado en cada una de las historias de usuarios y el proceso que se llevó hasta su culminación.

D1: Registro de docente

Los datos del docente son fundamentales a la hora de generar horarios. Por ello, se planteó dos escenarios que optimice el tiempo de registro dependiendo de la situación. El primer escenario se llama “Ingreso individual de un docente” y se busca realizar el registro del nombre y correo electrónico de un docente. Además, se envía un correo electrónico con una clave de acceso al sistema si este docente no se encuentra registrado previamente (Figura 14). Esto se debe a que el docente podrá solicitar que no se le asigne horas de clases en un horario que no se encuentre disponible.

Por otro lado, el segundo escenario se llama “Ingreso por archivo de docentes” y hace uso de un archivo *csv* que contenga a todos los docentes que se desea registrar. Esto permite optimizar el tiempo de registro en general. De la misma manera, el sistema informará si algún docente se encuentra registrado.

Estos dos escenarios fueron implementados como esquemas de escenarios en el lenguaje de *Gherkin*, dado así la posibilidad de probar los diferentes casos en base a los preámbulos que se decida indicar. Para ambos casos se consideraron:

- Uno o varios docentes no existen en el sistema y se trata de registrarlos. Se obtiene un mensaje de retroalimentación informado del registro exitoso. Asimismo, se envía un correo electrónico con la clave generada para acceder al sistema.
- Uno o varios docentes existen en el sistema y se trata de volveros a ingresar. Se obtiene un mensaje de retroalimentación indicando este error y, por ende, no se envía el correo electrónico a los docentes.

En base a la prueba, se pudo ver la necesidad de implementar una entidad docente y usuario, donde la segunda debe manejar un control de acceso con roles y encriptación de la clave generada. Por otro lado, se necesitó generar un algoritmo que permita obtener una contraseña que tenga las siguientes características:

- Una longitud de 16 dígitos.

- Al menos un carácter en mayúsculas, un número y un digito especial.

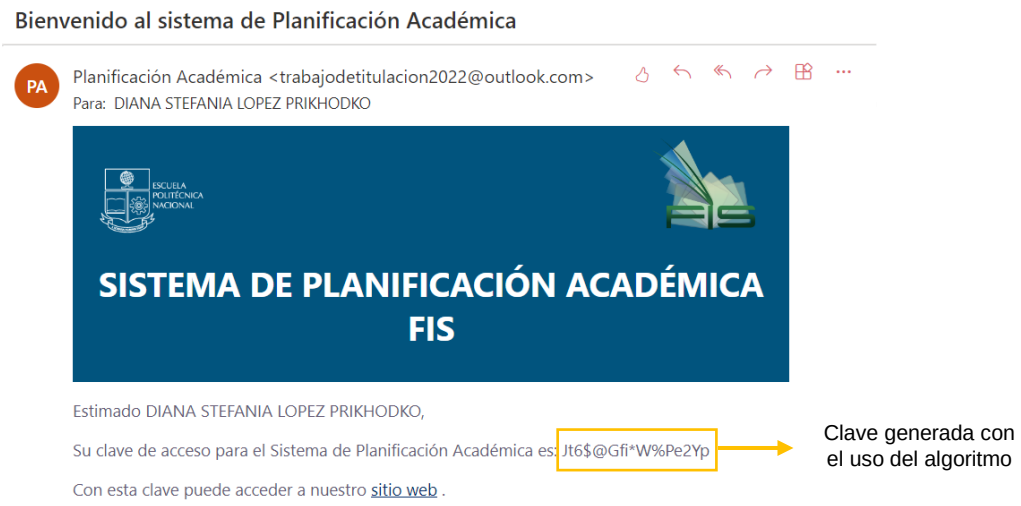


Figura 14. Correo enviado con la clave de acceso.

Por otro lado, se generó los diferentes métodos para la validación y envió de datos a la base de datos por medio de la API. Es importante denotar durante el proceso de generación de los métodos indicados, se documentó la API por medio de la herramienta *Swagger*.

A continuación, en la Figura 15 y 16 se presenta el resultado de la prueba una vez programado todo el código necesario:

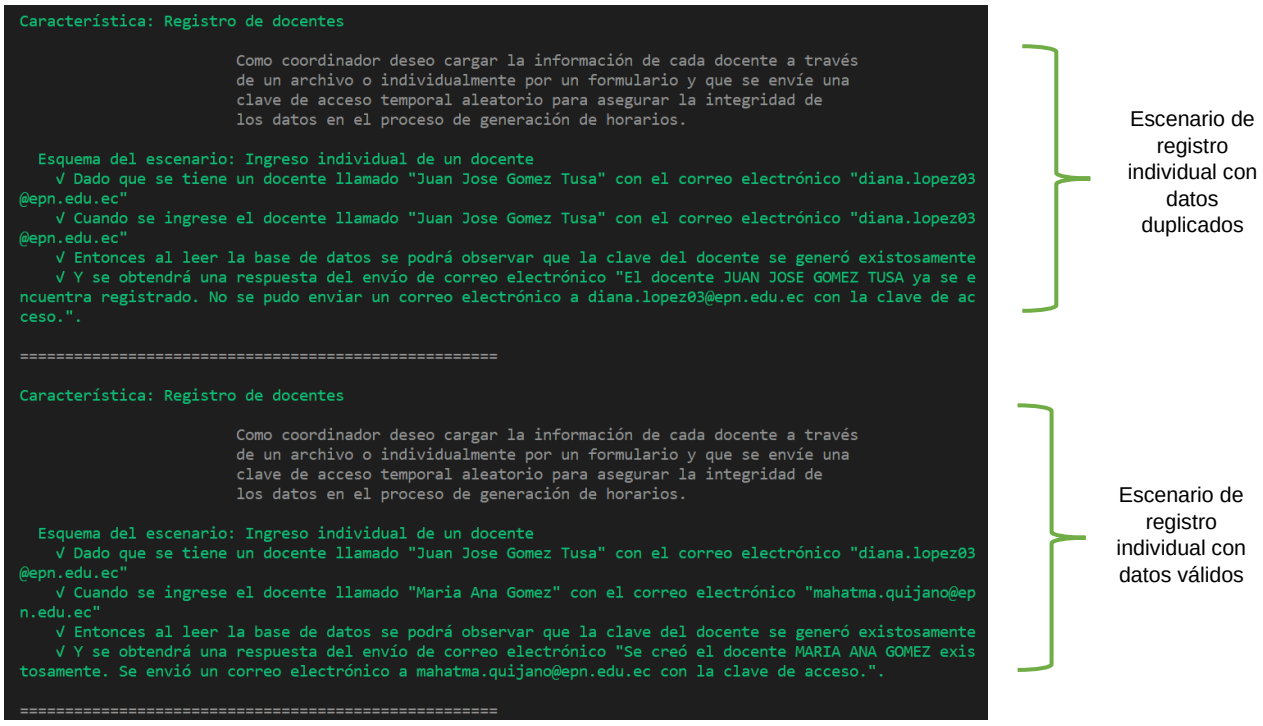


Figura 15. Resultados del escenario "Ingreso individual de un docente".

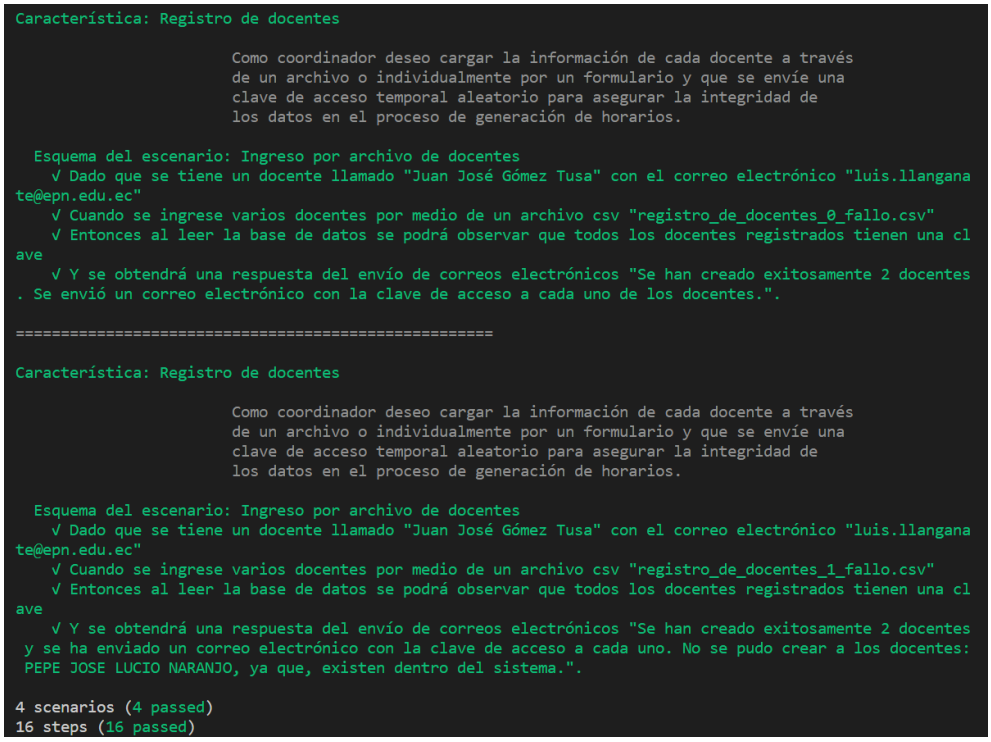


Figura 16. Resultados del escenario "Ingreso por archivo de docentes".

Una vez pasada las pruebas, se prosiguió con la implementación de la interfaz gráfica correspondiente al módulo de docentes, en donde se consideró las diferentes operaciones sobre los datos. Se tomó como referencia el prototipo mostrando en la Figura 17 y con el uso de los componentes de *Angular Material UI* se pudo implementar la interfaz mostrada en la Figura 18.

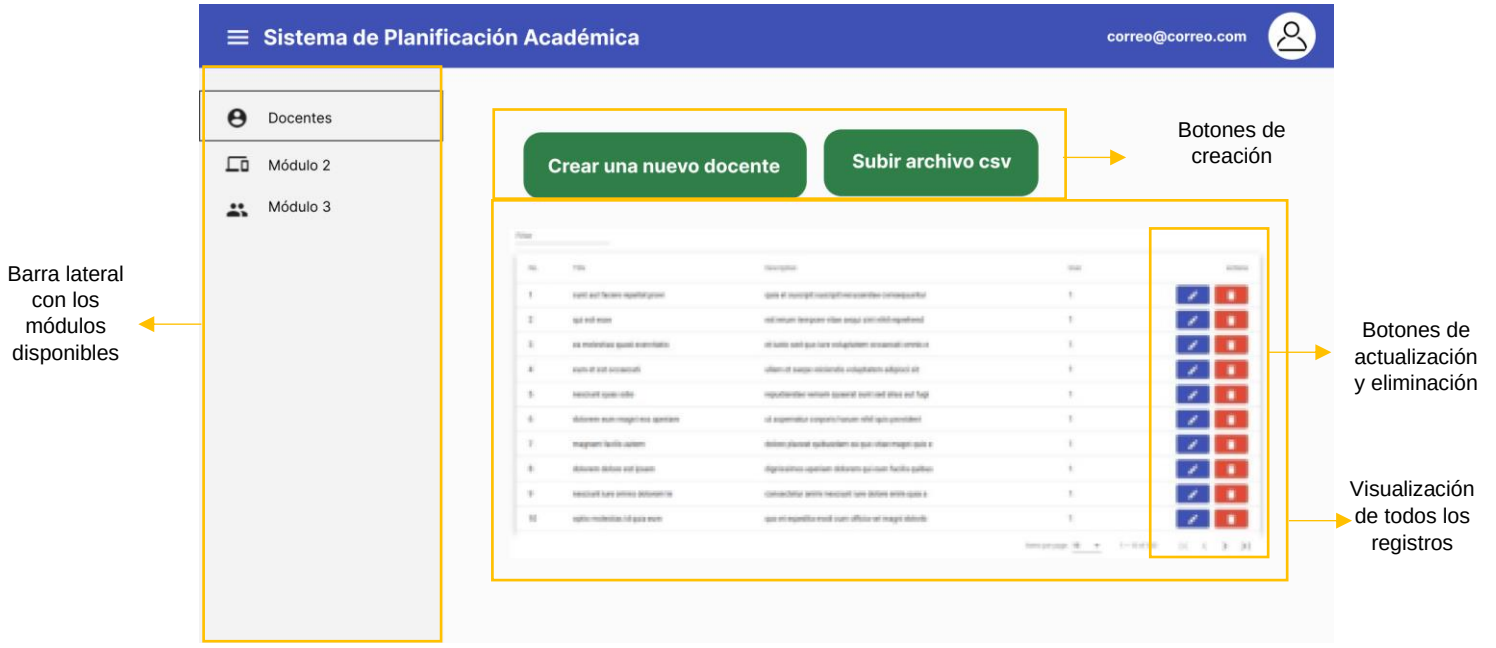


Figura 17. Prototipo del módulo de Docentes.

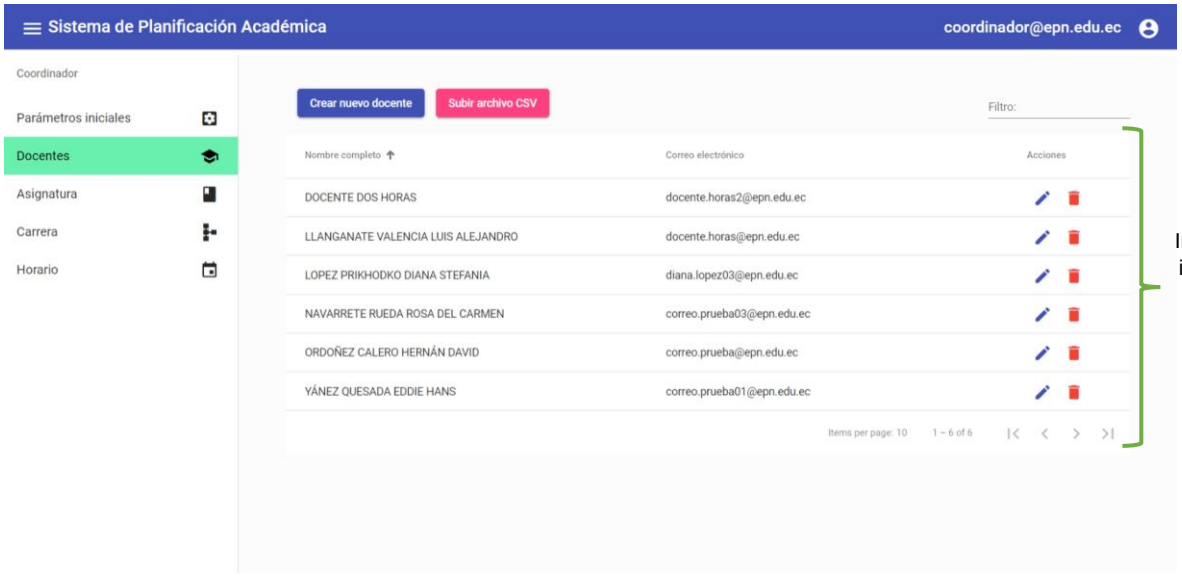


Figura 18. Interfaz del módulo de docentes.

D2: Registro de asignatura

Los datos de las asignaturas son igual de importantes que los docentes, ya que con su combinación se puede tener un bosquejo de un horario. En este caso se planteó dos escenarios que permitan mantener la integridad de los datos y optimicen el tiempo del proceso de registro. El primer escenario se llama “Ingreso individual de una asignatura” y representa el proceso de incluir los datos, tales como: código, nombre y créditos de una asignatura dentro de la planificación académica. En cuanto al segundo escenario “Ingreso por archivo de asignaturas” se busca algo similar al segundo escenario de la historia D1.

De la misma forma, los escenarios fueron implementados usando el lenguaje de *Gherkin* pero como esquemas de escenarios. Esto permite analizar varios casos:

- Una o varias asignaturas no existen en el sistema y se trata de registrarlas. Esto entrega un mensaje de registro exitoso.
- Una o varias asignaturas existen en el sistema y se trata de registrarlas. Esto entrega un mensaje indicando el error por duplicación.

Con esto se puede concluir la necesidad de implementar una entidad asignatura indicando las reglas de cada uno de los campos. Consecuentemente, se generaron los métodos y se documentó la API. A continuación, en la Figura 19 se presenta el resultado de la ejecución de las pruebas una vez generado el código:

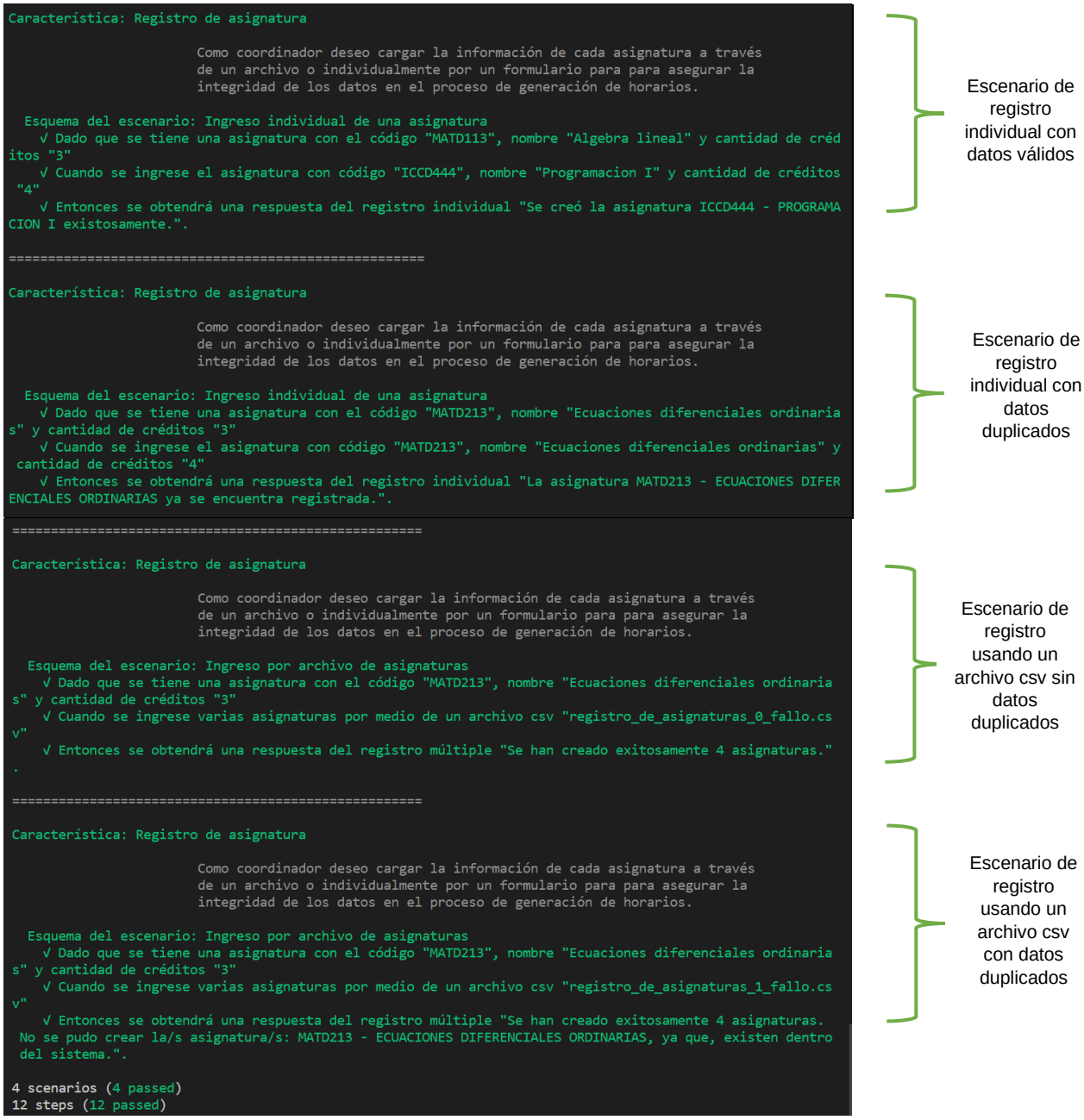


Figura 19. Resultados de los escenarios "Ingreso individual de una asignatura" e "Ingreso por archivo de asignaturas".

Para completar la historia de usuario, se inició el proceso de implementación de la interfaz gráfica correspondiente al módulo de asignaturas, en donde se consideraron operaciones similares al módulo de docentes. Se tomó en cuenta el prototipo mostrado en la Figura 20 y los componentes entregados por *Angular Material UI* para implementar la interfaz mostrada en la Figura 21.

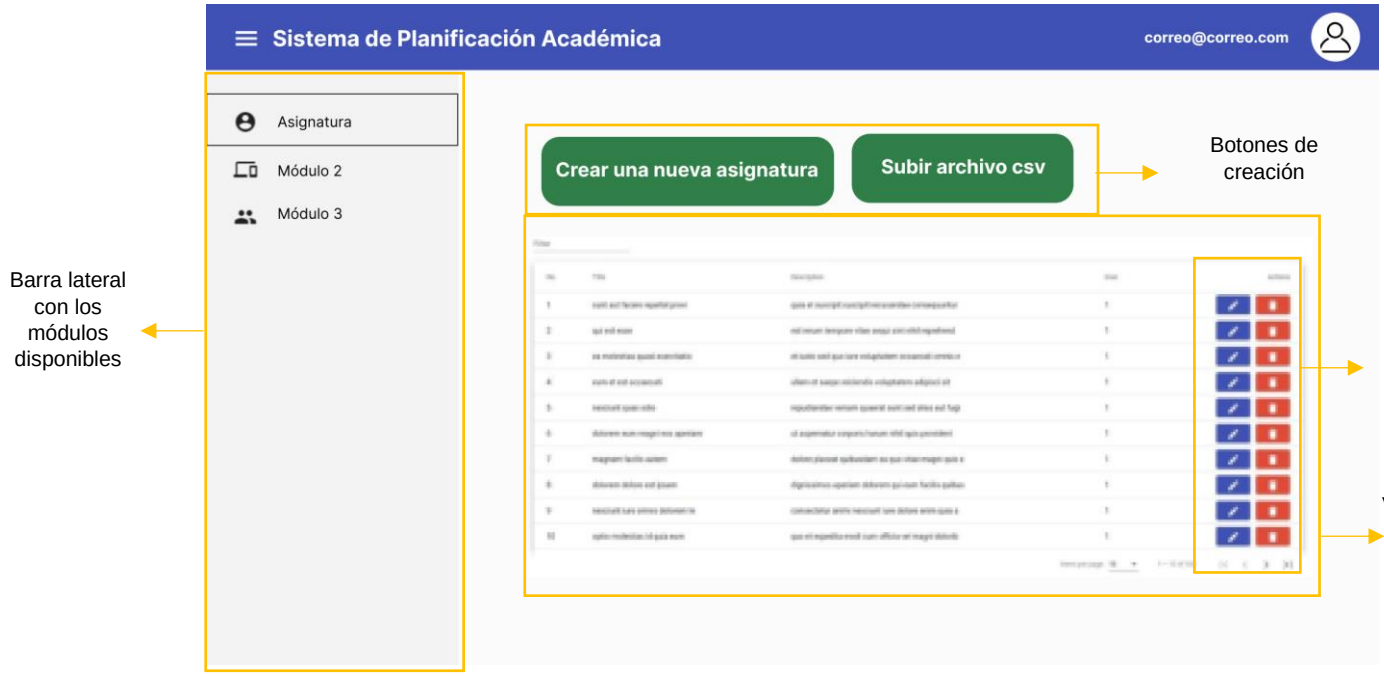


Figura 20. Prototipo del módulo de Asignatura.

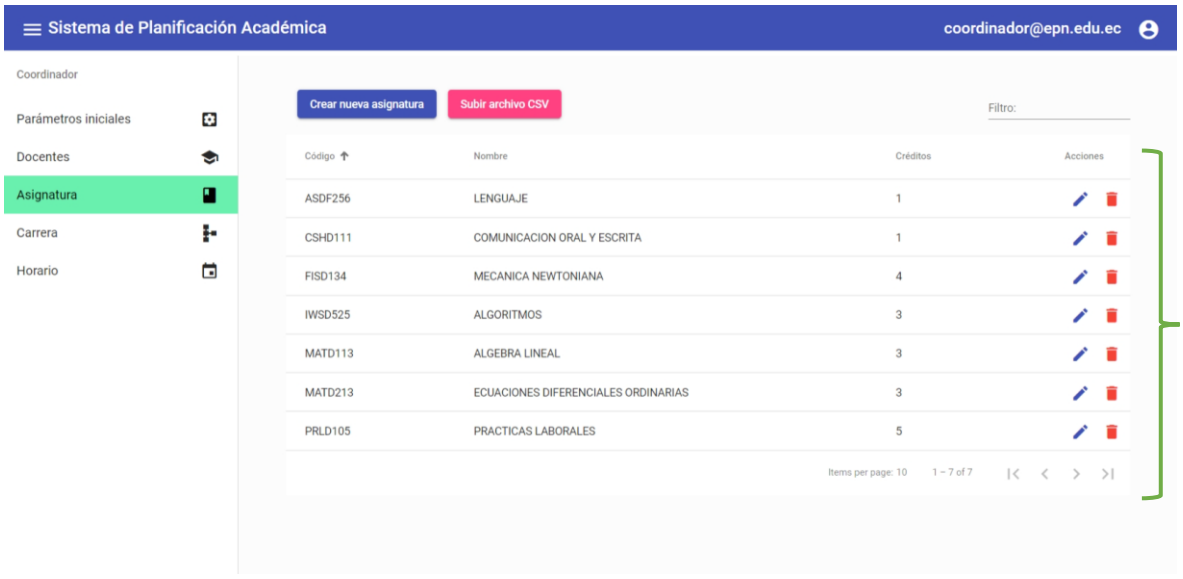


Figura 21. Interfaz del módulo de Asignatura.

D3: Registro de carreras

De manera similar a las dos historias anteriores, el proceso de análisis de esta historia llego a plantear un único escenario. Esto se debe a que la cantidad de datos que se ingresan son reducidos y no se necesitaría ampliar esta funcionalidad (como los módulos anteriores). Pero se debe tomar en cuenta realizar un esquema de escenario, puesto que

se debe controlar la integridad de los datos y evitar su duplicación. Por ello se planteó los siguientes casos:

- Una carrera que no existe previamente en el sistema y se desea ingresar. Se obtendrá un mensaje de registro exitoso.
- Una carrera existente en el sistema y se trata de volver a ingresar. Se obtendrá un mensaje de error por duplicación de registros.

En base a esto se necesitó implementar una entidad de carrera y sus diferentes métodos en la API. Finalizado lo anterior, se puede observar en la Figura 22 el resultado de la ejecución de la prueba con sus dos casos:

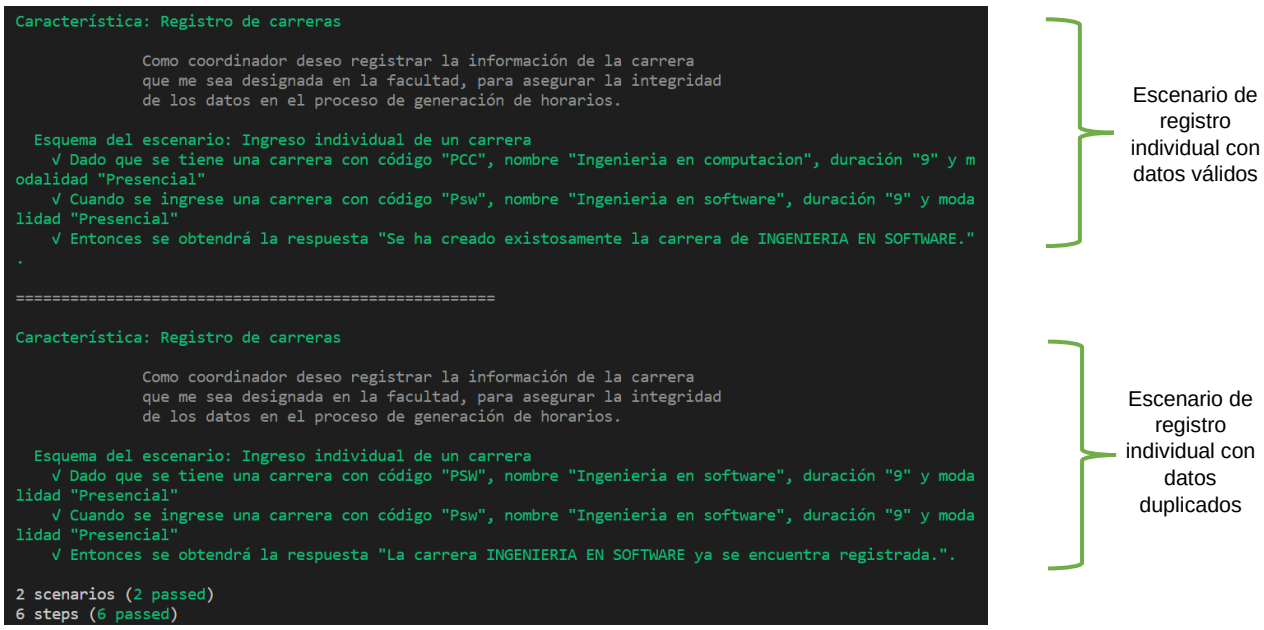


Figura 22. Resultado del escenario "Ingreso individual de carreras".

Posteriormente, se inició el proceso de implementación de la interfaz gráfica correspondiente al módulo de carrera, en donde se consideraron operaciones similares a los módulos anteriores. Además, se implementó la interfaz mostrada en la Figura 24 en base al prototipo planteado en la Figura 23.

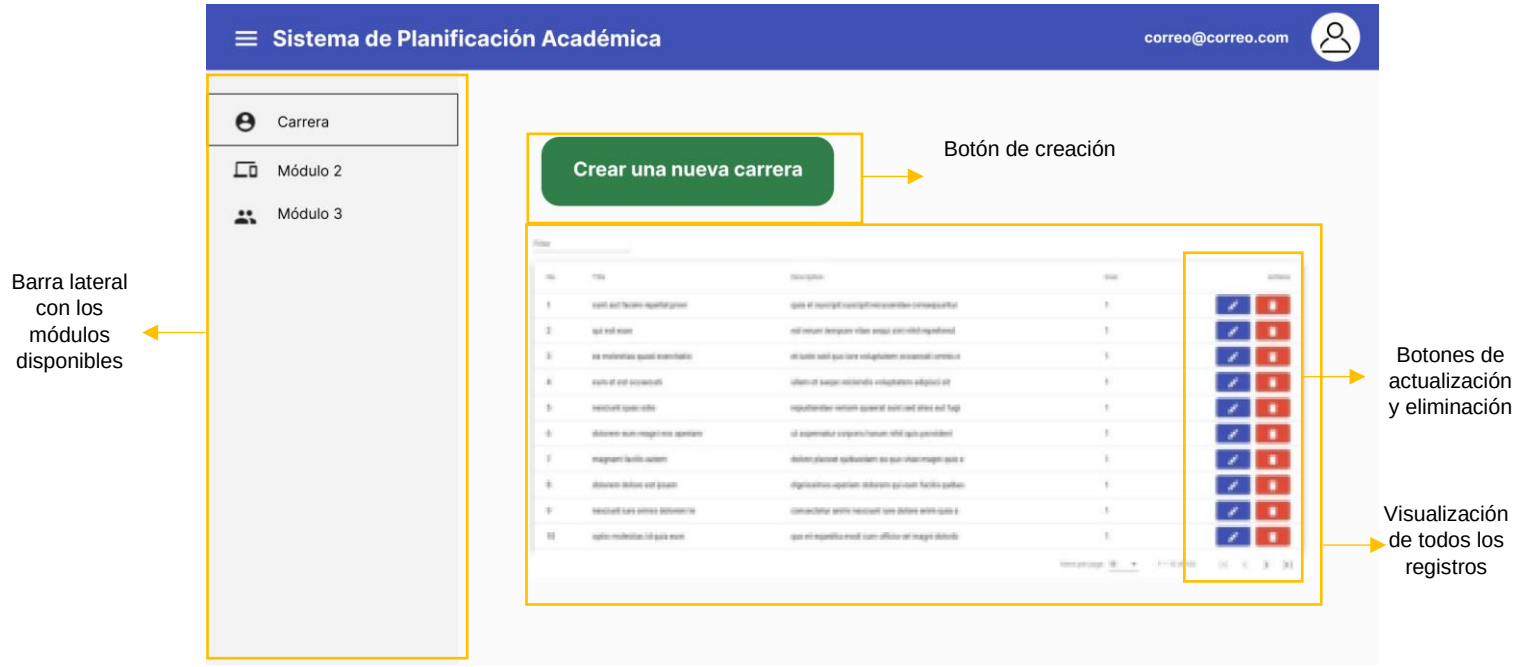


Figura 23. Prototipo del módulo de Carrera.

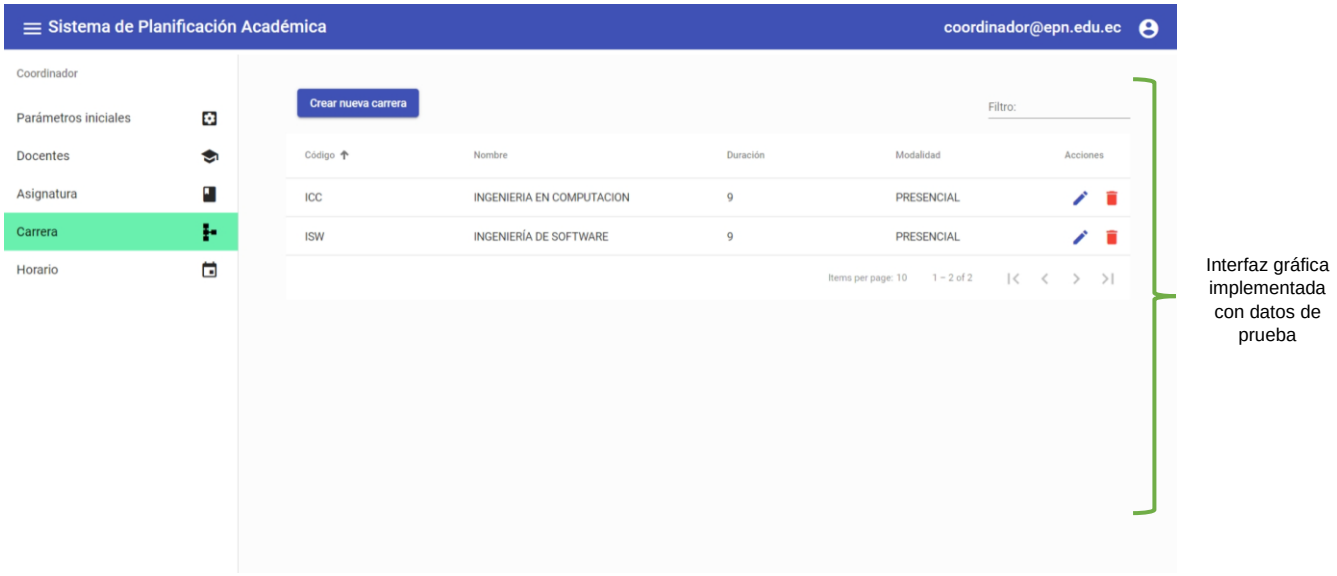


Figura 24. Interfaz del módulo de Carrera.

D5: Visualización de horarios

Como se pudo apreciar en la Figura 1, la visualización de horarios necesita como entrada los diferentes horarios generados con el software FET. Sin embargo, se definió previamente el formato en el que debe entregar los horarios (archivo tipo .json) y se generó un par de ejemplos para poder realizar las pruebas con datos más reales. Para esta historia

se planteó realizar dos escenarios en donde se deba probar varios algoritmos de filtrado, los cuales fueron mencionados en la primera entrevista.

El primer escenario se llama" Visualización de horarios usando filtro por docente" y se enfoca en entregar de las horas correspondiente a un docente. La hora se encuentra conformada por la asignatura, el día, la hora, el tipo de espacio físico y un espacio físico asignado (opcional). Por otro lado, el segundo escenario se llama "Visualización de horarios usando filtro por grupo", en donde se considera un grupo a la combinación del semestre, carrera y aula. A diferencia del anterior, la hora almacena también el nombre del docente. Estos dos escenarios no fueron implementados como esquemas de escenarios en el lenguaje de *Gherkin*, puesto que no se tenía una gran variedad de horarios generados.

En base a la prueba, se pudo ver la necesidad de implementar nuevas entidades tales como: horario, jornada y grupo. A la vez, se generó los diferentes métodos para la obtención y filtrado de los horarios. De igual forma, se documentó la API por medio de la herramienta *Swagger*. En la Figura 25 se presenta el resultado de la prueba una vez programado todo el código necesario:

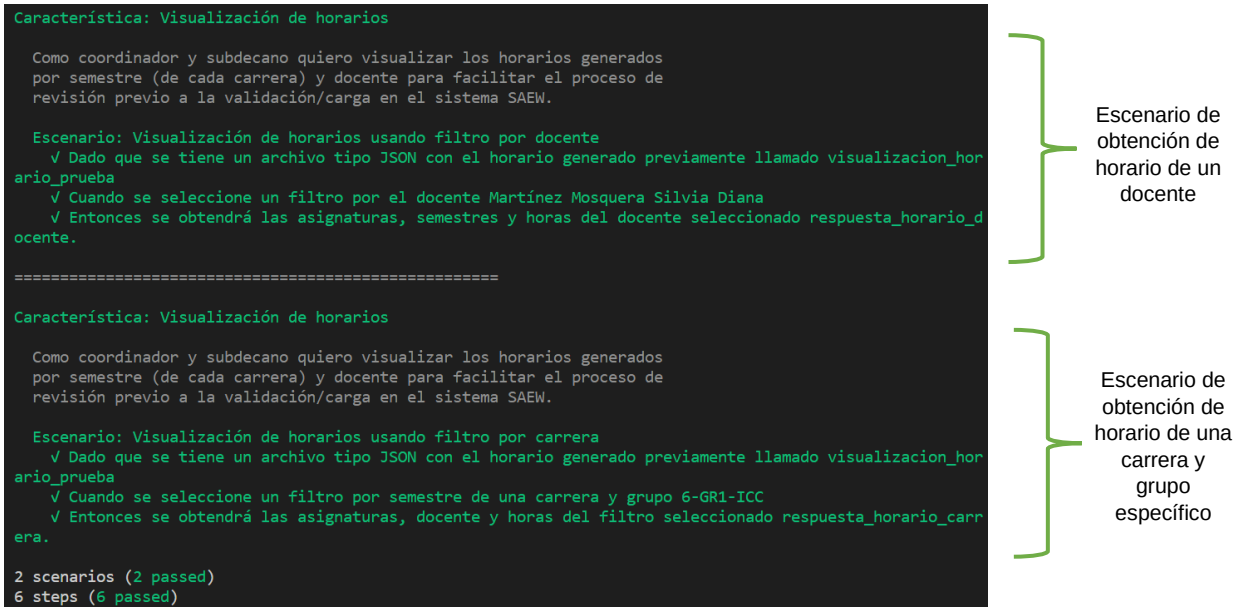


Figura 25. Resultados de los escenarios "Visualización de horarios usando filtro por docente y por carrera".

Para el desarrollo del *frontend* de esta historia se consideraron dos interfaces importantes dentro del proceso de validación. La primera interfaz va a mostrar una lista de horarios generados previamente (Figura 26), indicando una descripción (ingresada por el coordinador o subdecano cuando generen el horario), la fecha/hora de creación y el correo

electrónico. Mientras que, la segunda interfaz se basó en el prototipo indicando en la Figura 27 y se encarga de mostrar un horario en base al filtro seleccionado (Figura 28).

Figura 26. Interfaz del módulo de Horario.

Figura 27. Prototipo de la pantalla de visualización de horarios generados por filtros.

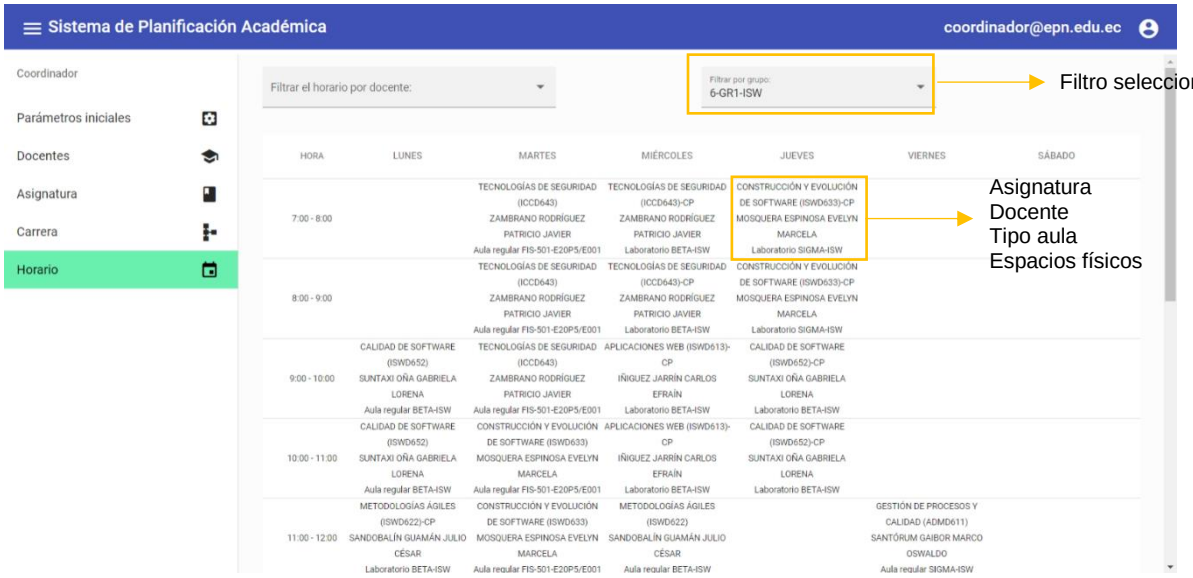


Figura 28. Interfaz de la pantalla de visualización de horarios generados por filtros.

4 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

4.1 Resultados

Para el desarrollo del componente “Manejo de datos ingresados por el usuario y validación/aprobación de horarios generados” se basó en las buenas prácticas y actividades definidas por BDD, la cual se explica en la sección 2.4. Por otro lado, el marco de trabajo para la gestión de proyectos Scrum fue aplicado para la división, organización de requerimientos y definición de roles dentro del equipo. Esto permitió enfocar el esfuerzo en el desarrollo a los escenarios de mayor valor para el proceso de planificación académica.

La fase de investigación (sección 3.1) fue primordial para comprender el funcionamiento de la planificación académica dentro de la Facultad de Ingeniería en Sistemas y a su vez, permitió definir los objetivos y comportamientos deseados por los diferentes involucrados. Asimismo, la fase de diseño ayudó a la transformación de necesidades en comportamientos programables en el lenguaje *Gherkin* (usando la herramienta *Cucumber*). Esto mostró de manera fácil a los desarrolladores el valor que se busca en cada una de las historias de usuario planteadas.

Como se mencionó en la sección 3.3, hubo varios problemas que retrasaron el desarrollo del proyecto. Por ello, se realizó una revaloración de cada historia de usuario y se definió no realizar las historias que únicamente son mejoras al proceso actual. La primera historia descartada fue la D4, dado que mejora la integración de los datos de asignaturas, carreras y semestres (entidad que debía ser creada). Esto alargaría el proceso de ingreso de datos al sistema, pero reduciría el esfuerzo al generar las actividades (relación de todos los datos que analiza el FET para generar los horarios).

De igual forma, la historia D6 completaría el proceso actual de planificación académica, ya que actualmente no existe un proceso formal de cierre. Sin embargo, la naturaleza cambiante de la cantidad de cupos por asignatura (conocido en el negocio como proceso de preplanificación), no permite definir adecuadamente un rango de tiempo fijo para cada actividad. Además, durante el proceso de matrículas varios estudiantes modifican o deciden tomar asignaturas diferentes a las esperadas y esto da necesidad de eliminar o aumentar grupos en asignaturas puntuales hasta los primeros días de clases.

A pesar de todo esto, se logró cumplir con los objetivos del componente, gracias a la aplicación de BDD (concentrarnos en el comportamiento de valor a entregar). Se puede comprobar que los datos de docentes, carreras y asignaturas se encuentran estructurado

de tal manera que se evita la duplicación y mantiene la integridad de estos. Igualmente, las interfaces gráficas de visualización de horarios son sencilla e intuitiva gracias al manejo de la librería *Angular Material UI*, usada en muchos sitios web. Esto permite que el proceso de validación realizado por los coordinadores y subdecano se lo pueda ejecutar de manera paralela y óptima en base a los filtros entregados.

En la Figura 29 se presenta el desarrollo de las historias planificadas en un inicio durante cada uno de los *sprints*. Se debe considera que las faltantes se refieren a las descartadas en el proceso de desarrollo, ya que representan mejoras no obligatorias al proceso actual de planificación académica:

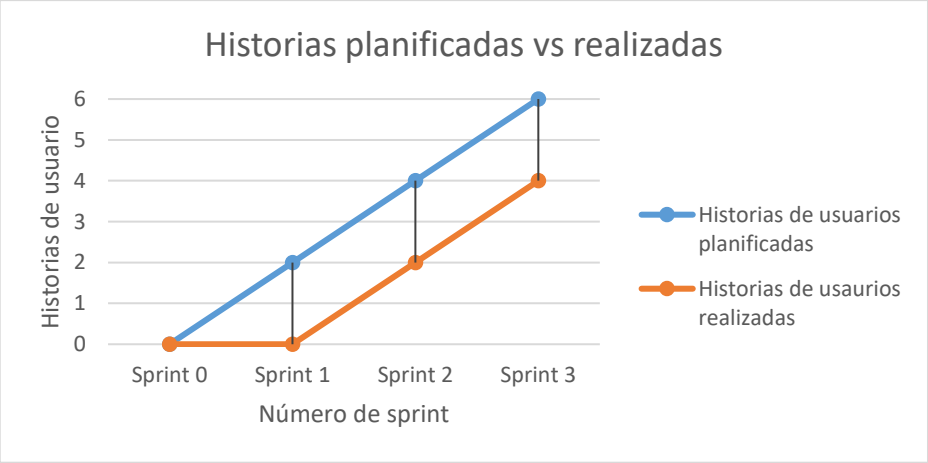


Figura 29. Resumen de historias planificadas y realizadas.

4.2 Conclusiones

El Sistema de Planificación Académica de la Facultad de Ingeniería de Sistemas se encuentra subdivido en 3 componentes que abarcan los diferentes procesos mostrados en la Figura 1. El componente “Manejo de datos ingresados por el usuario y validación/aprobación de horarios generados” es un conjunto de módulos que se encuentran diseñados para facilitar el uso y asegurar la integridad de los datos ingresados.

Como se explicó en la sección 3.1, se realizó un proceso que permitió generar valor a los coordinadores y subdecano al ingresar datos de docentes, asignaturas y carreras; en donde se realizó validaciones a nivel de *frontend* y *backend*. Para el *backend* se usó *TypeORM*, el cual permite controlar los datos bajo reglas específicas previo a la inserción en la base de datos. Por otro lado, en el frontend se usó el componente *FormControl* (interno de *Angular*) que permitió controlar los valores que son ingresados en los campos de texto.

Para agilizar el proceso de análisis y validación de los horarios entregados por el motor *FET*, se consideró la visualización general e individual de los horarios entregados. La interfaz gráfica de esta funcionalidad fue planteada en base a la necesidad del análisis por medio de filtros de docentes y grupos. Asimismo, se consideró de manera constante la experiencia de los coordinadores y subdecano al momento de implementar esta funcionalidad.

Adicionalmente, la integración del proyecto se lo hizo de manera continua entre los tres componentes, obteniendo así versiones estables constantemente y asegurando la compatibilidad entre módulos. Esto se lo hizo por medio del manejo de una rama de desarrollo llamada “develop”, en donde se colocan únicamente las versiones estables de cada uno de los miembros del equipo. No obstante, debido a la que el primer componente fue aplazado para entregar el semestre 2022B, no se puede observar el flujo completo del proceso de planificación académica.

4.3 Recomendaciones

Durante el desarrollo del componente “Manejo de datos ingresados por el usuario y validación/aprobación de horarios generados” se evidenciaron varias dificultades durante el desarrollo, puesto que al manejar un nuevo lenguaje de programación (N*estJS*) y la herramienta *Cucumber* implican un tiempo de capacitación que no fue considerado al momento de planificar el componente. Por ello se recomienda que se analice el impacto de aplicar una herramienta o lenguaje nuevo. Esto permitiendo poder evitar retrasos durante el desarrollo.

Si bien el uso de *Cucumber* permite tener un mejor control en lo que realmente el cliente espera del producto, se debe entender que no se deben generar varios escenarios donde se pruebe individualmente cada operación de la característica específica (operaciones). Mas bien, muchas de estas operaciones se encuentran intrínsecamente dentro de un escenario de manera completa. Esto con el fin de evitar la generación de excesivas pruebas que no son de valor para el usuario.

El código fuente de la aplicación cuenta con problemas en la consistencia del manejo de comentarios y el nombre de variables/métodos, dado que cada desarrollador tiene su propio estilo y no se definió un estándar adecuado. Con esto se aconseja generar un acuerdo entre todo el equipo para mantener un código ordenado y mejor estructurado. Esto con el fin de facilitar a las personas que deseen continuar con la mejora del sistema o bien entender el funcionamiento.

A lo largo del desarrollo se tuvieron varios inconvenientes por varios sucesos sociales y políticos, los cuales influenciaron en la comunicación entre los interesados y el equipo de desarrollo. Por eso se recomienda que se maneje varios canales de comunicación, tales como: videollamadas, reuniones presenciales, correos electrónicos o mensajería instantánea. Además, se recomienda establecer un homólogo o suplente en caso de que los interesados principales no se encuentren disponibles.

5 REFERENCIAS BIBLIOGRÁFICAS

[1] Consejo Politécnico de la Escuela Politécnica Nacional , «Estatuto de la Escuela Politécnica Nacional (Codificación). Resolución RCP-315-2019,» 8 Agosto 2019. [En línea]. Available: https://cei.epn.edu.ec/Documentos/EPN/EstatutoEPN_Agosto_2019.pdf.

[2] A. J. Ulloa Arévalo, «Desarrollo de una aplicación web de planificación académica para el Subdecanato de la Facultad de Ingeniería de Sistemas,» Quito, 2021.

[3] G. L. Lalescu y V. Dirr, «FET Free Timetabling Software,» 18 Agosto 2022. [En línea]. Available: https://lalescu.ro/liviu/fet/.

[4] W. D. Leonardini Aparicio, «Sistema de Planificación y Seguimiento académico. Caso: Unidad Educativa RVDO. P. Walter Strub,» La Paz, 2009.

[5] F. Quispe Maquera, «Sistema de Gestión Académica para la Unidad Educativa Daniel Sanchez Bustamante II SISGESA,» La Paz, 2014.

[6] I. G. Pineda Arias, «Sistema Inteligente de Soporte en la Generación de Horarios Académicos para la Carrera de Ingeniería de Sistemas de la Universidad Politécnica Salesiana,» Quito, 2010.

[7] G. L. Lalescu y V. Dirr, «FET Manual,» 17 Agosto 2018. [En línea]. Available: https://www.timetabling.de/manual/FET-manual.en.html#id_2.

[8] J. Ferguson Smart, BDD IN ACTION. Behavior-Driven Development for, Shelter Island: Manning Publications , 2015.

[9] K. Mendes Calo, E. Estevez y P. Fillottrani, «Un Framework para Evaluación de Metodologías Ágiles,» Bahía Blanca.

[10] E. Herrera Uribe y L. E. Valencia Ayala, «Del manifiesto ágil sus valores y principios,» Pereira, 2007, pp. 381-386.

[11] J. C. Sandobalín Guamán, J. d. J. Aguiar Pontes y J. F. Lucio Naranjo, Interviewees, *Requerimientos del Sistema de Planificación Académica para la Facultad de Ingeniería de Sistemas.* [Entrevista]. 19 Noviembre 2021.

[12] C. McKenzie, «What is living documentation?,» Diciembre 2014. [En línea]. Available: https://www.techtarget.com/searchsoftwarequality/definition/living-documentation.

6 ANEXOS

ANEXO I. Extracción de información

Disponible en:

https://bit.ly/3cQy94V

ANEXO II. Revisión sistemática

Disponible en:

https://bit.ly/3qhj67j

ANEXO III. Entrevistas y reuniones

Entrevista a los *Stakeholders* realizada por Carlos Eduardo Anchundia Valencia:

https://youtu.be/c6QVJtO-f4A

Entrevista a Alex Ulloa (autor de la tesis previa) por el equipo de desarrollo:

https://youtu.be/r8gOMTsNGzQ

Entrevista a José Francisco Lucio Naranjo (subdecano de la FIS) por el equipo de desarrollo:

https://youtu.be/8Y60S9U4XAg

ANEXO IV. Herramienta de gestión del proyecto - Notion

Disponible en:

https://bit.ly/3cNxZLH

ANEXO V. Repositorio del backend

Disponible en:

https://github.com/Trabajo-de-Titulacion/planificacion-academica-fis-backend.git

ANEXO VI. Repositorio del frontend

Disponible en:

https://github.com/Trabajo-de-Titulacion/planificacion-academica-fis-frontend.git

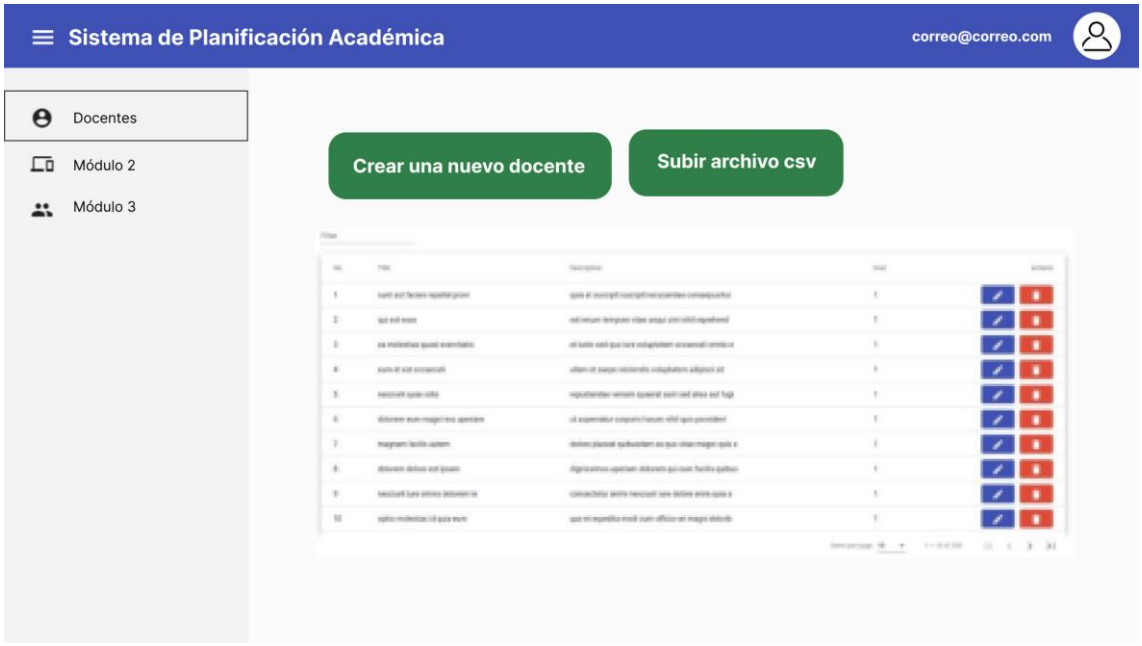
ANEXO VII. Backlog del proyecto

N°	Título	Descripción	Estimación
1	Parámetros iniciales	Como coordinador deseo establecer los parámetros iniciales como horas disponibles, los días hábiles, edificios y tipos de aulas, para definir los límites en la generación de horario y que cumplan con las limitaciones de la institución.	2
2	Generación de horarios con el software FET	El sistema necesita enviar los datos de entrada (docentes, asignaturas, aulas, carreras, paralelos y sus restricciones) al software FET vía TCP/IP para obtener como salida los horarios de cada nivel académico.	3
3	Difusión de horarios	Como subdecano deseo difundir los horarios aprobados a los respectivos involucrados para comunicar los horarios a los docentes, jefa de departamento, y secretaria para que se registren en el sistema académico de la institución con el fin de ser considerados en el proceso de matriculación de los estudiantes.	3
4	Creación de actividades	Como coordinador deseo poder crear una actividad antes de generar el horario seleccionando un profesor, la asignatura, aulas y sus restricciones para definir los horarios por nivel.	2
5	Registro de docente	Como coordinador deseo cargar la información de cada docente a través de un archivo o individualmente por un formulario y que se envíe una clave de acceso aleatorio para asegurar la integridad de los datos en el proceso de generación de horarios.	3
6	Registro de asignatura	Como coordinador deseo cargar la información de cada asignatura a través de un archivo o individualmente por un formulario para para asegurar la integridad de los datos en el proceso de generación de horarios.	2
7	Registro de carreras	Como coordinador deseo registrar la información de la carrera que me sea designada en la facultad, para asegurar la integridad de los datos en el proceso de generación de horarios.	2
8	Visualización de horarios	Como coordinador y subdecano quiero visualizar los horarios generados por semestre (de cada carrera) y docente para facilitar el proceso de	3

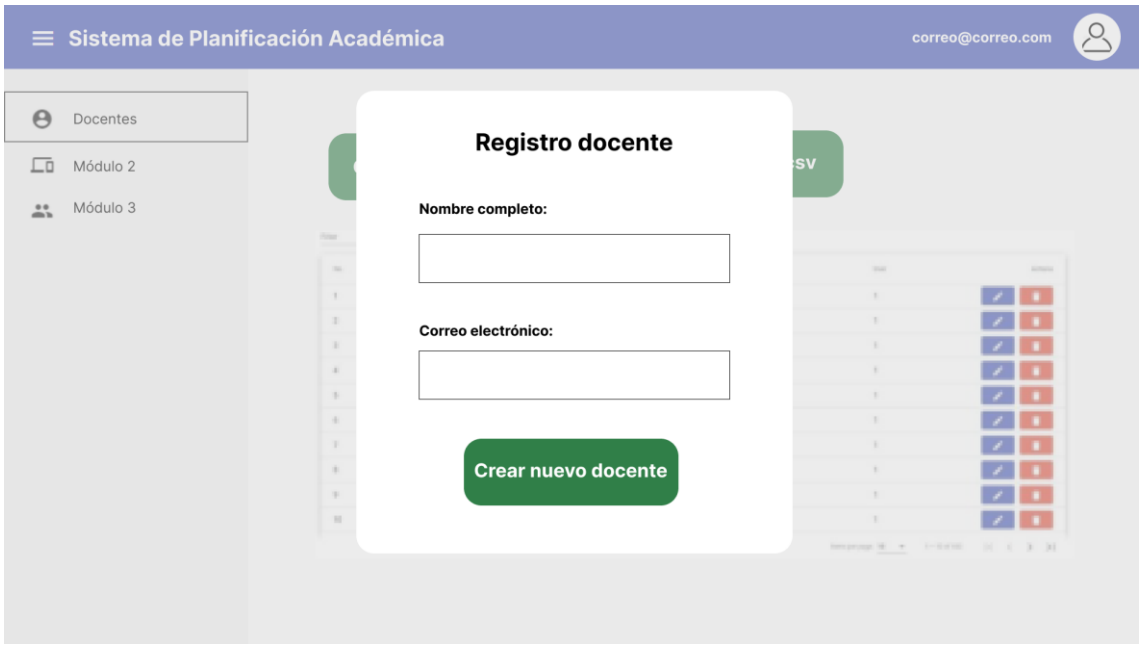
		revisión previo a la validación/carga en el sistema SAEW.	
9	Vinculación de carreras con asignaturas	Cómo coordinador deseo vincular los semestres de cada carrera existente con las asignaturas correspondientes a los mismos, para asegurar que la generación de horarios se enfoque en evitar cruces entre asignaturas del mismo nivel y carrera.	2
10	Aprobación de horarios	Como subdecano deseo aprobar los horarios generados por el software FET para evitar errores, reclamos y modificaciones no autorizadas con respecto a los horarios antes del inicio del semestre subsecuente.	2
11	Registro de espacios físicos	Como gestor de espacios físicos deseo registrar las diferentes aulas y laboratorios, ya sea de forma masiva a partir de un archivo o individualmente a través de un formulario, para distribuir los cursos de manera adecuada así evitando problemas por falta de infraestructura y evitar aglomeraciones.	2
12	Registro de horas no disponibles del docente	Como docente deseo solicitar al jefe de departamento la consideración de mis horas no disponibles, seleccionadas en una matriz, por cada día de la semana para que el horario del periodo subsecuente que me sea asignado pueda acoplarse a mis actividades académicas, de docencia y de investigación.	3
13	Registro de alumnos por asignatura	Como coordinador deseo ingresar a través de un formulario el número de estudiantes que cursarán cada una de las asignaturas del semestre actual, para poder planificar los horarios en base a los cupos que deberán estar disponibles para las asignaturas de dicho semestre.	3
14	Pre-planificación académica de horarios	Como coordinador deseo calcular el número de estudiantes que puedan matricularse en cada asignatura del semestre subsecuente a partir de la información de los estudiantes del semestre actual para contar con datos fiables que ayuden a disminuir la incertidumbre de los cupos que deberán estar disponibles el semestre subsecuente y determinar un número más certero para las asignaturas del siguiente nivel académico.	2

ANEXO VIII. Prototipos de los módulos

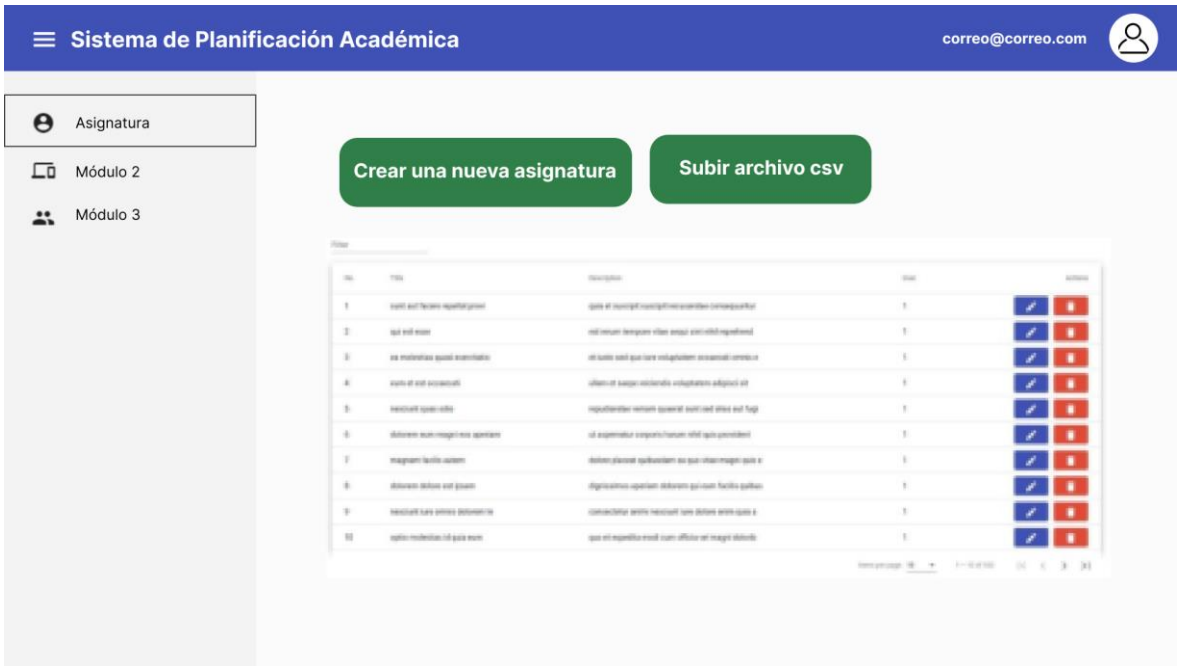
Prototipo 1: Visualización de docentes



Prototipo 2: Creación individual de docente

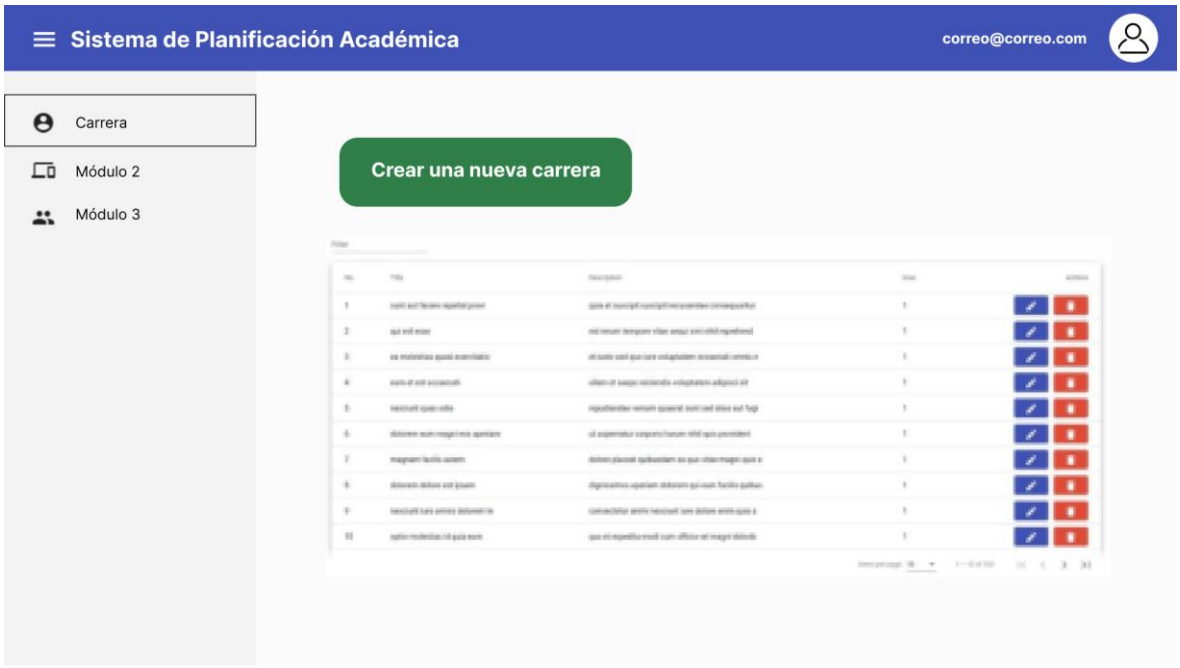


Prototipo 3: Visualización de asignaturas



Prototipo 4: Creación de asignatura

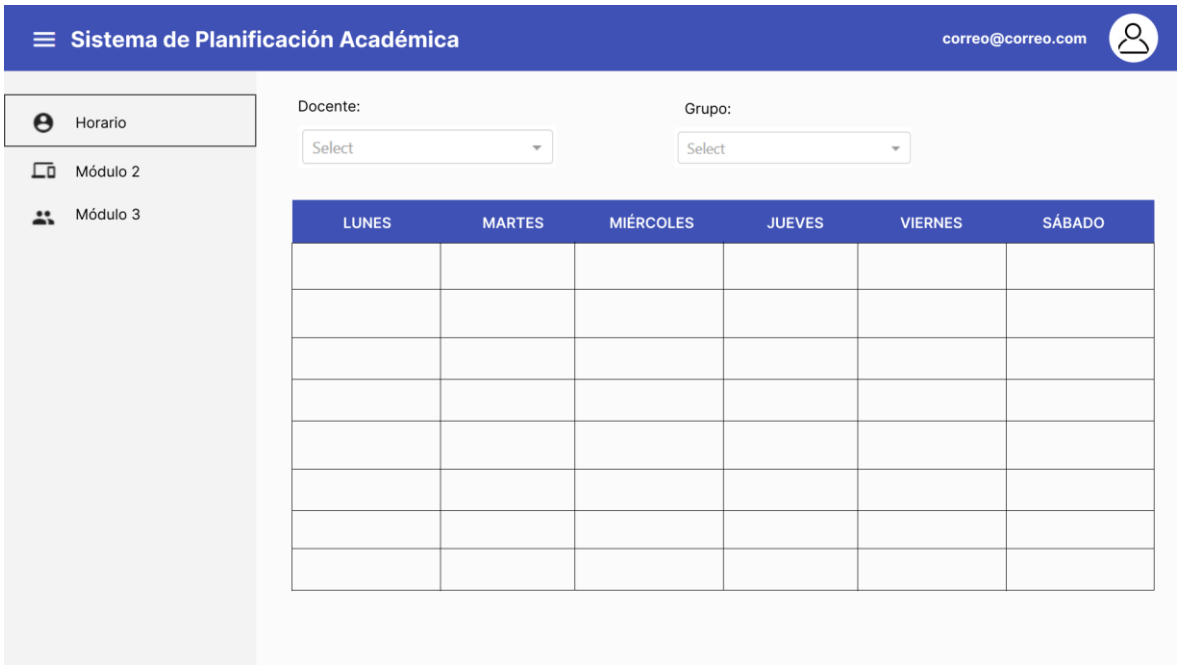
Prototipo 5: Visualización de carreras



Prototipo 6: Creación de una carrera

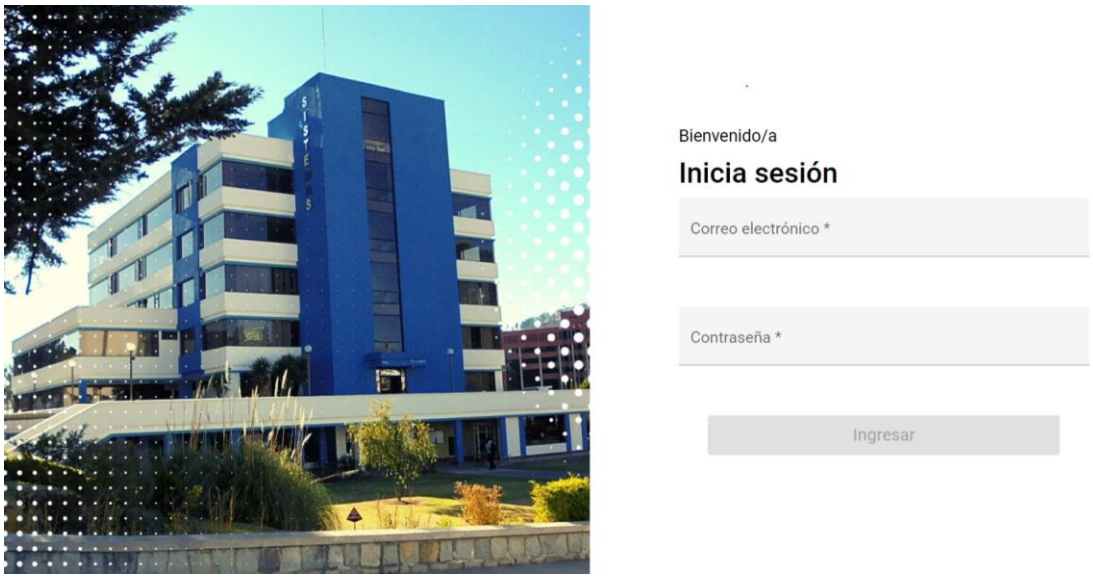
Prototipo 7: Visualización de horarios generados

Prototipo 8: Visualización de un horario específico en base a filtros

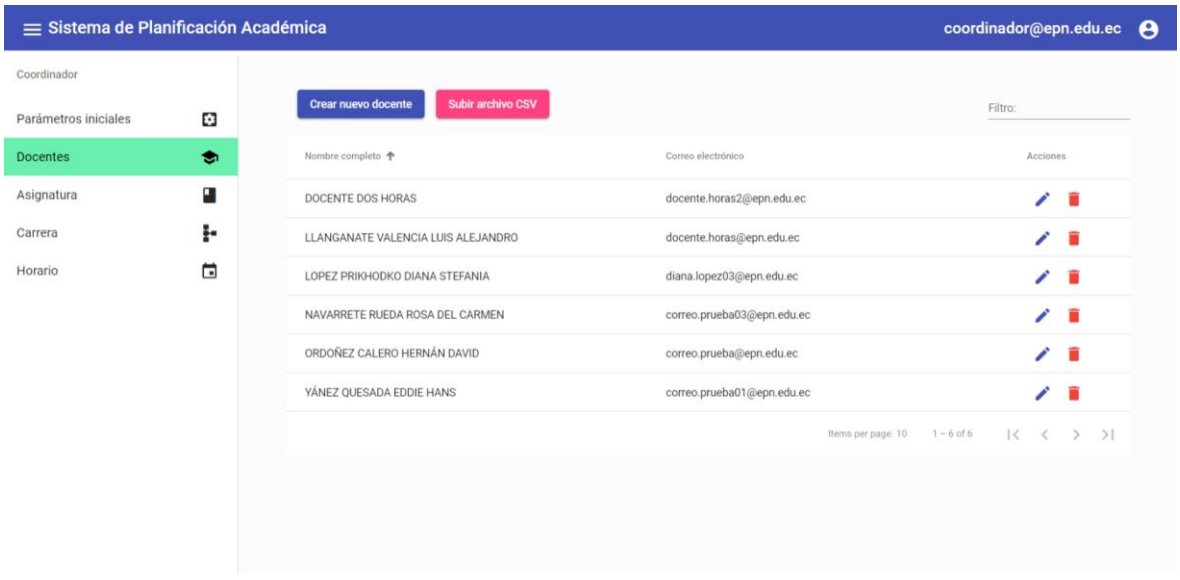


ANEXO IX. Interfaces gráficas del componente

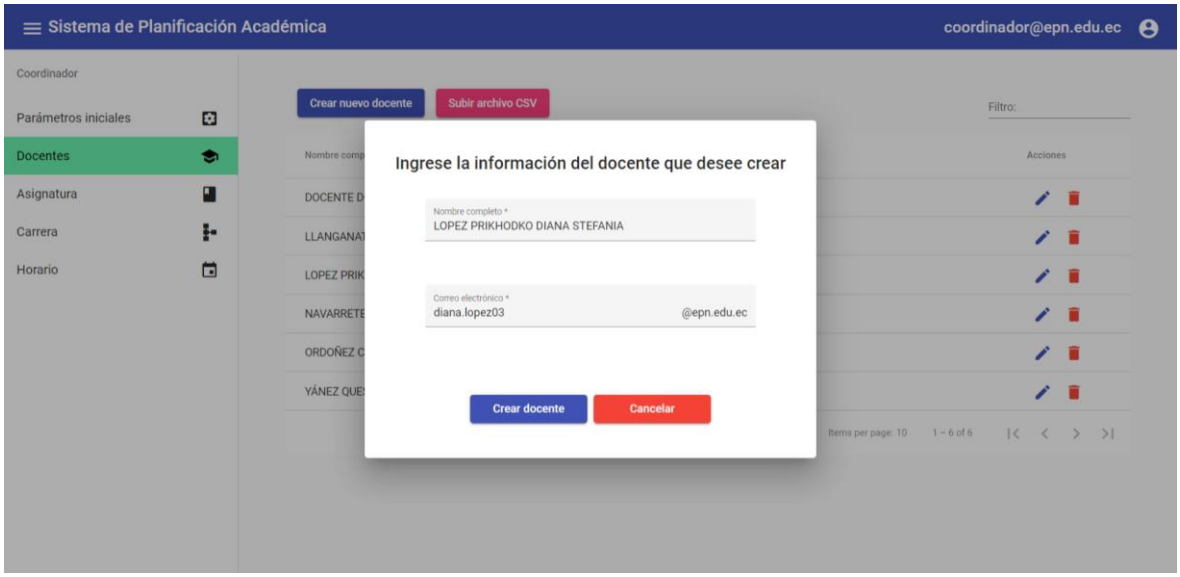
Pantalla 1: Inicio de sesión



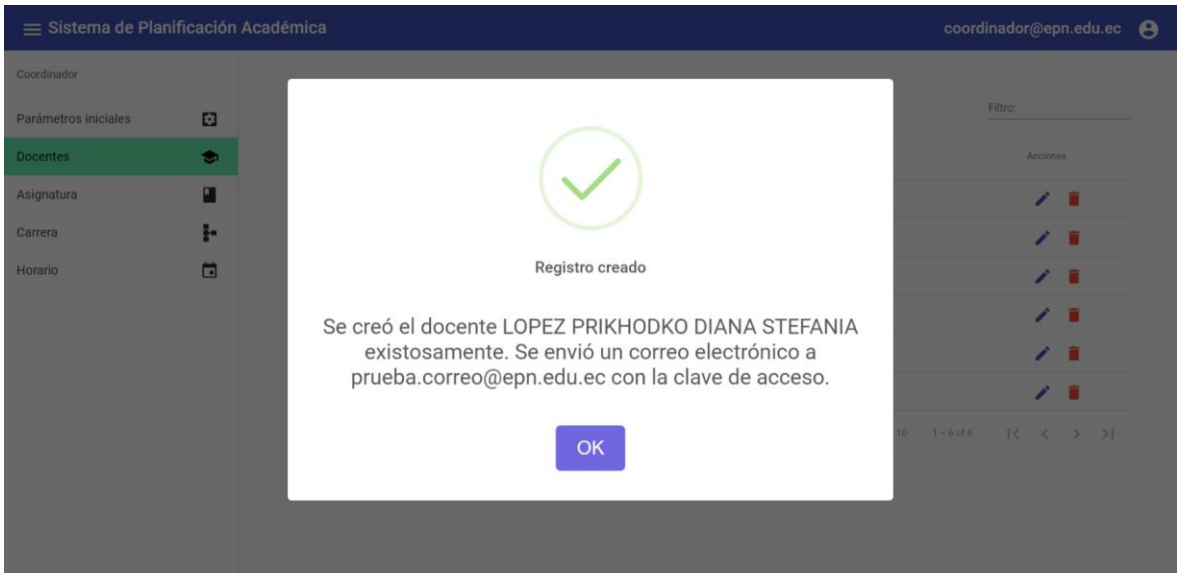
Pantalla 2: Visualización de docentes



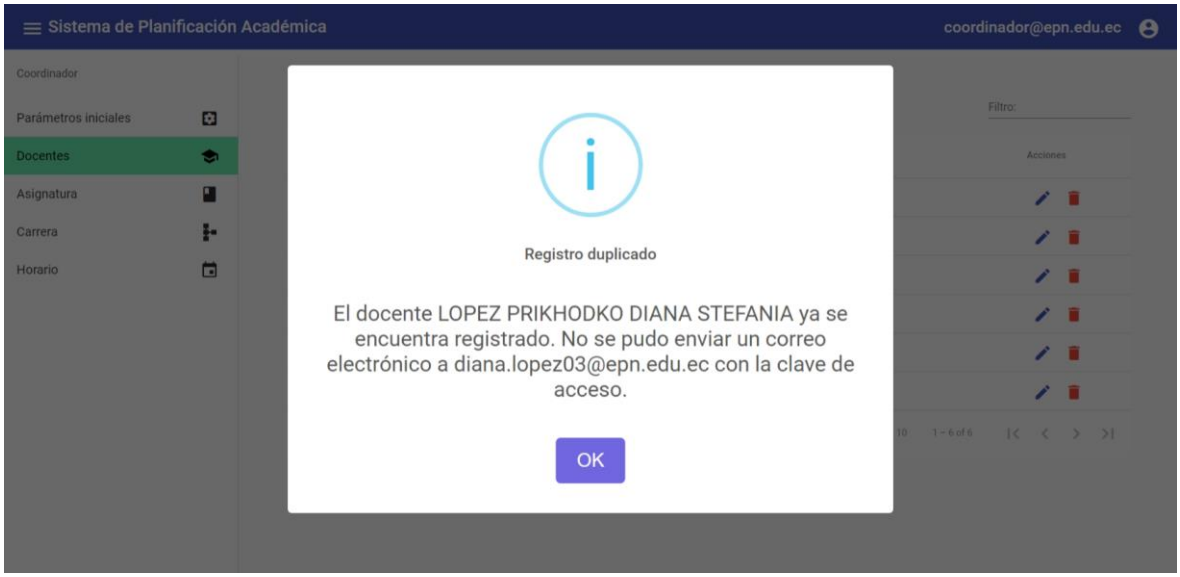
Pantalla 3: Crear un docente



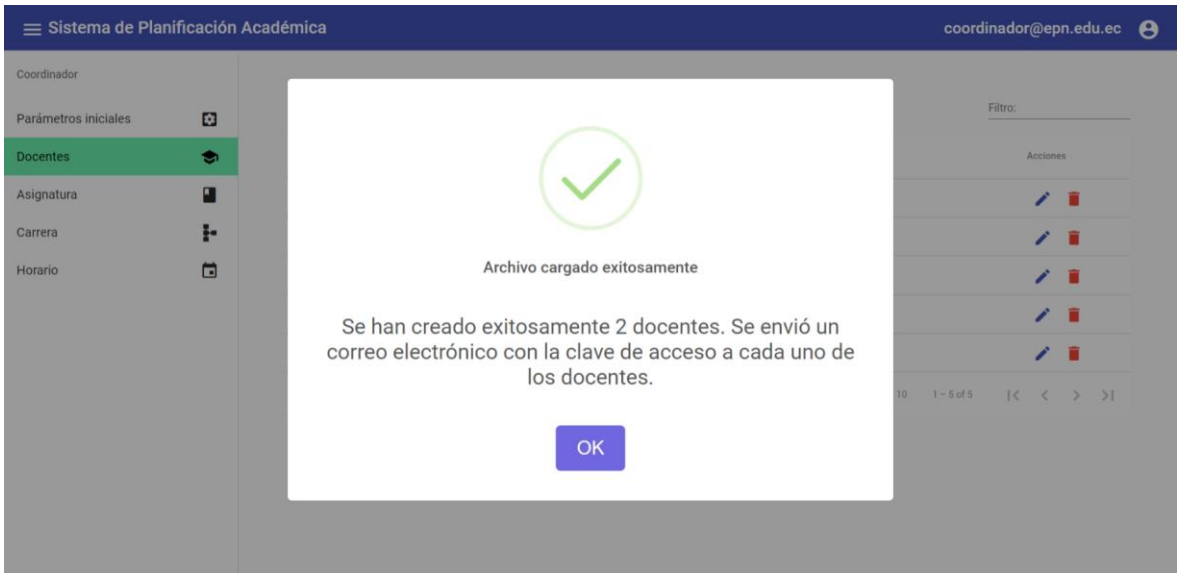
Pantalla 4: Creación exitosa de un docente



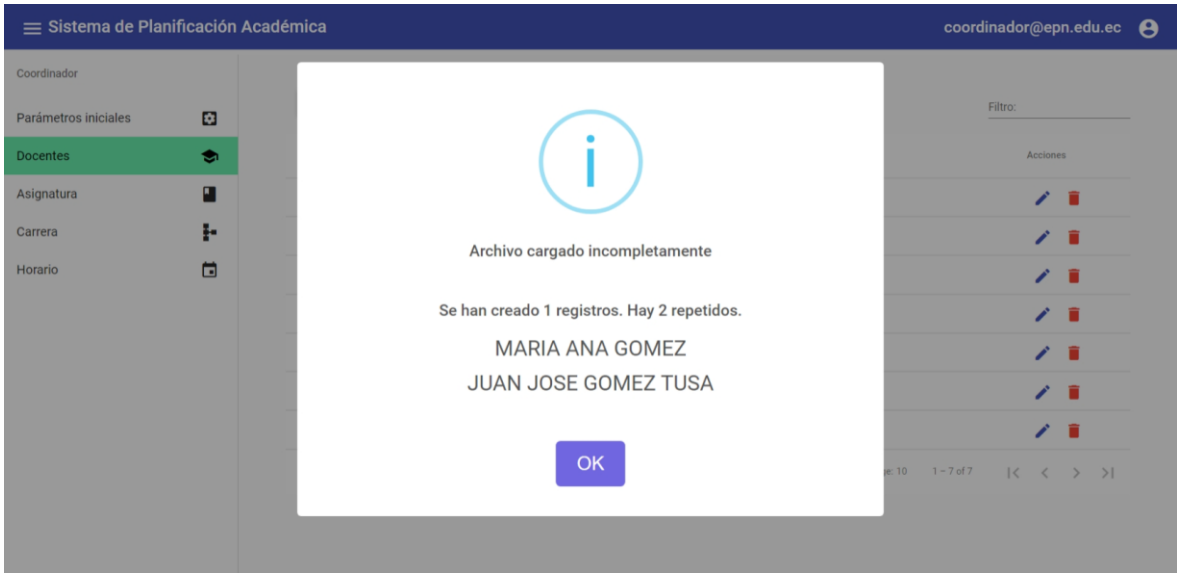
Pantalla 5: Creación fallida de un docente



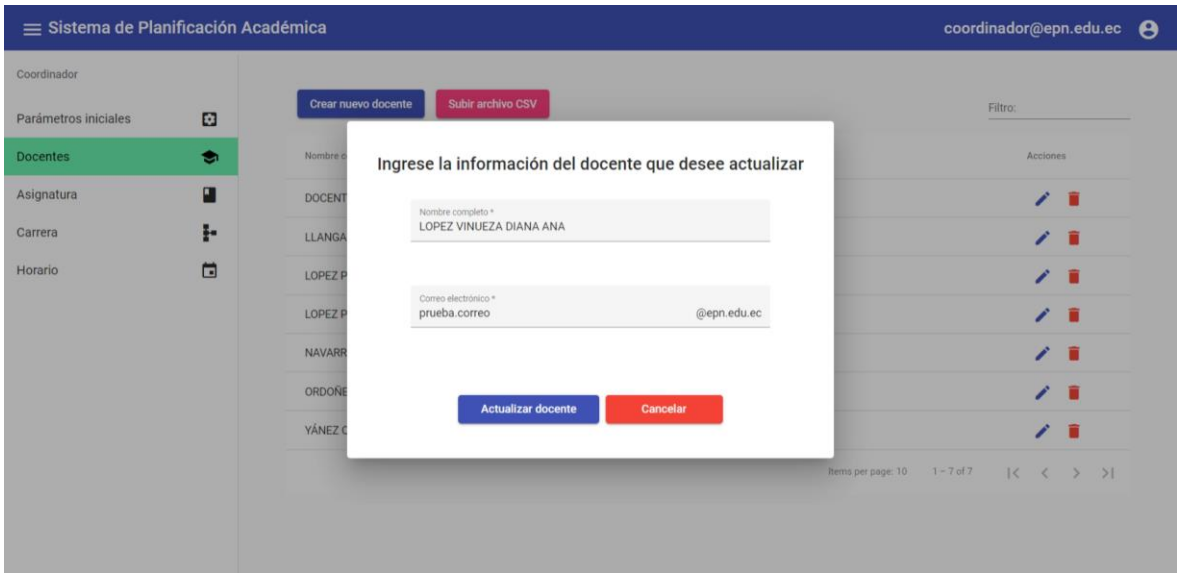
Pantalla 6: Creación exitosa de varios docentes por medio de un archivo *csv*



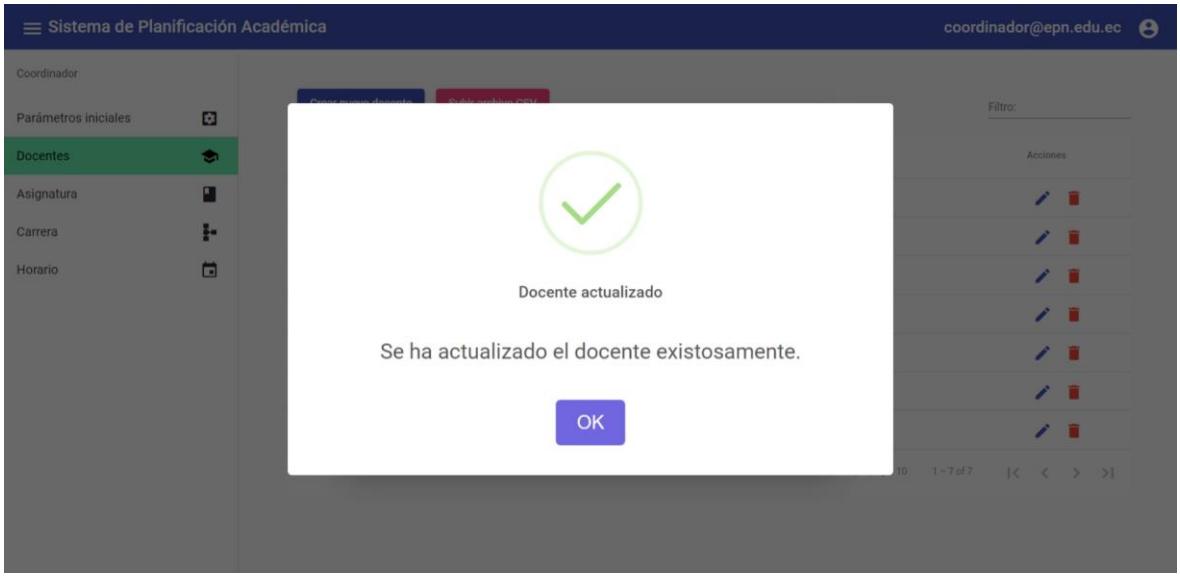
Pantalla 7: Creación fallida de varios docentes por medio de un archivo *csv*



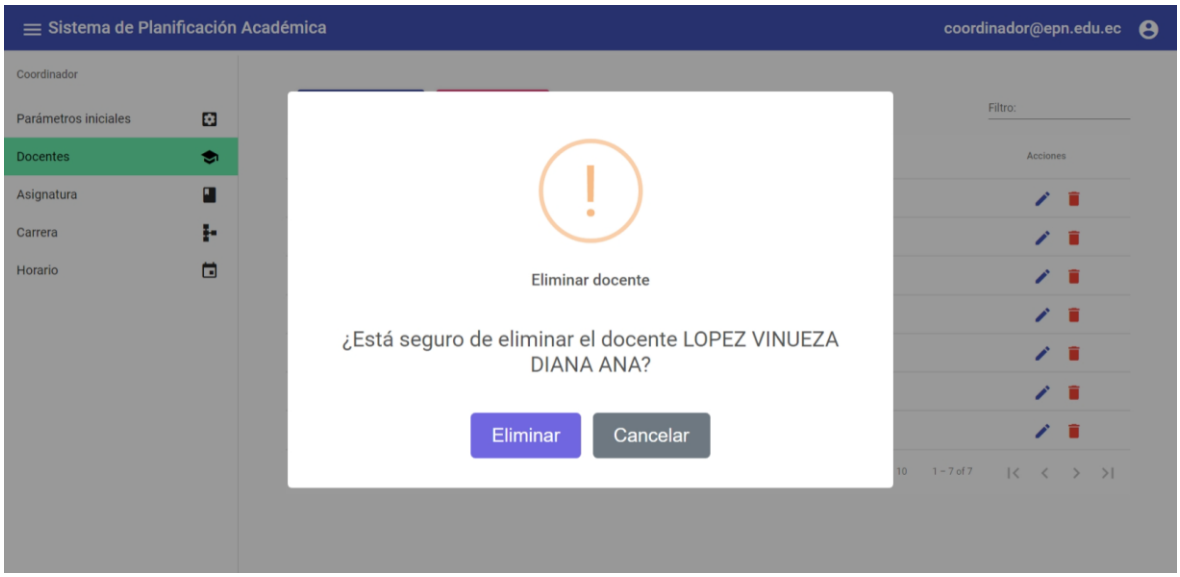
Pantalla 8: Actualización de un docente existente



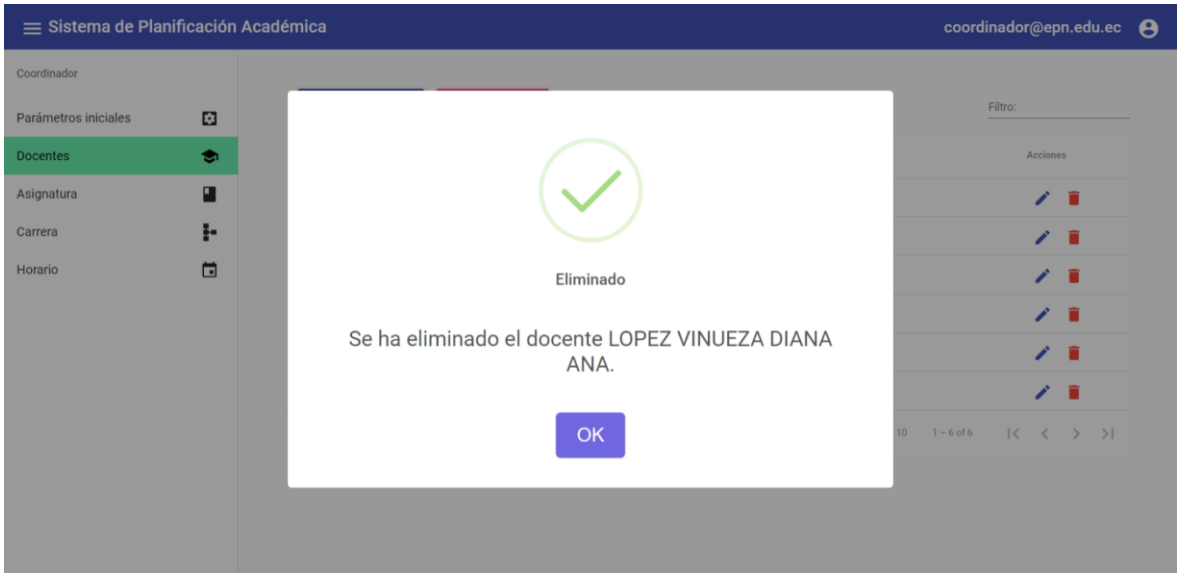
Pantalla 9: Actualización exitosa de un docente existente



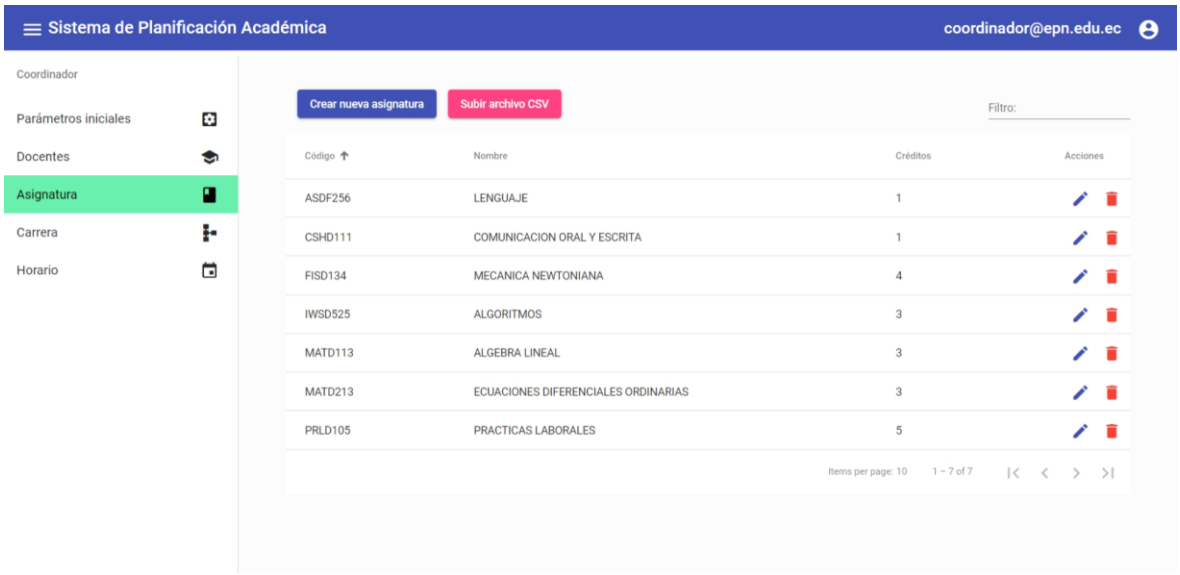
Pantalla 10: Eliminación de un docente existente



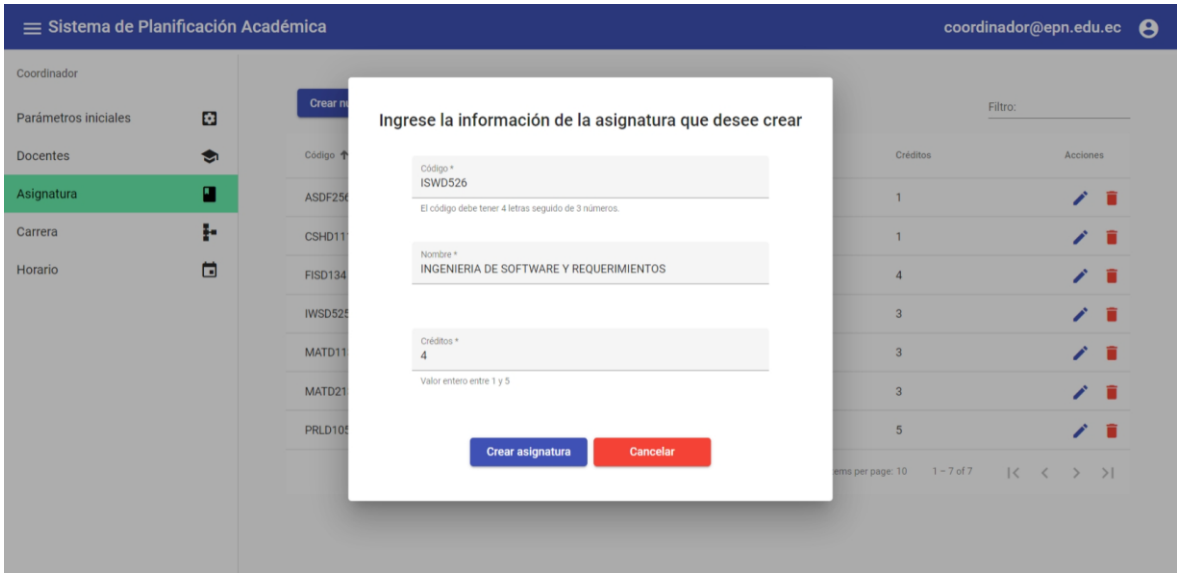
Pantalla 11: Eliminación exitosa de un docente existente



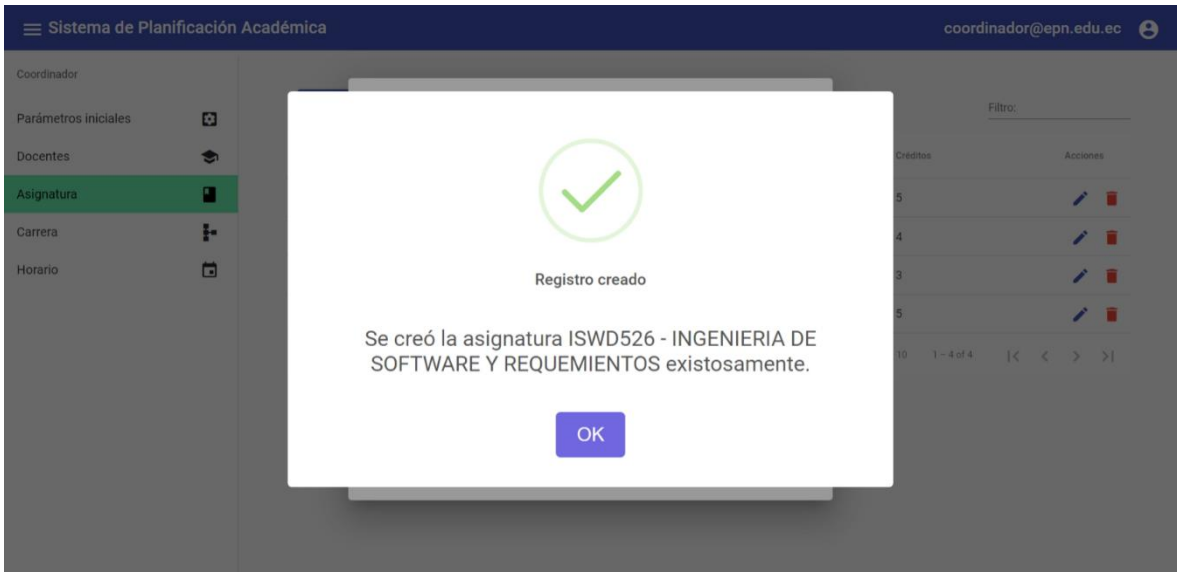
Pantalla 12: Visualización de asignaturas



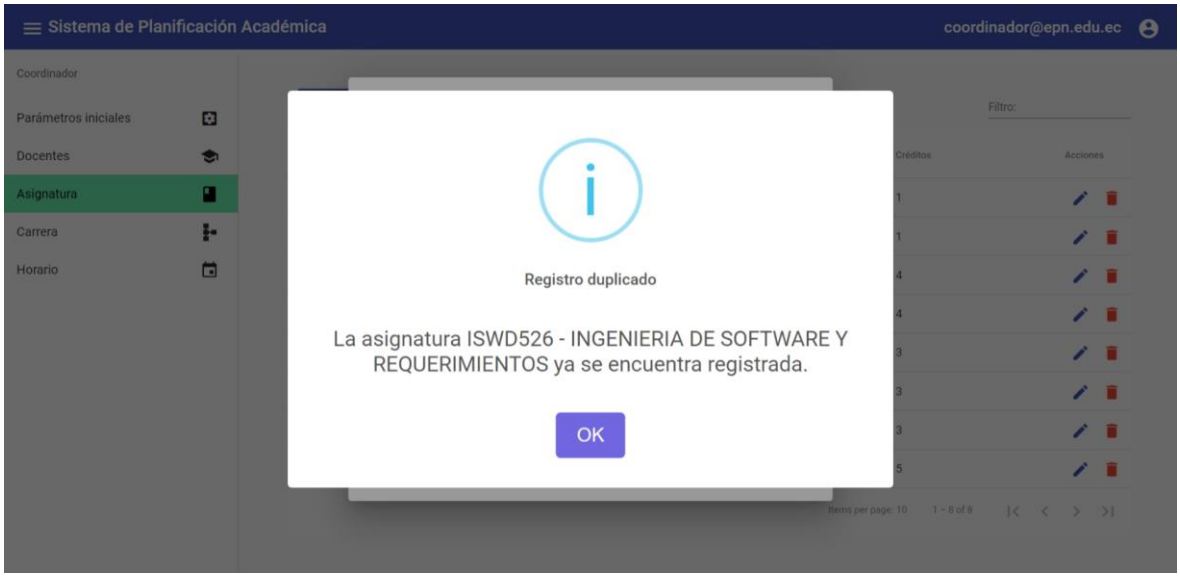
Pantalla 13: Crear una asignatura



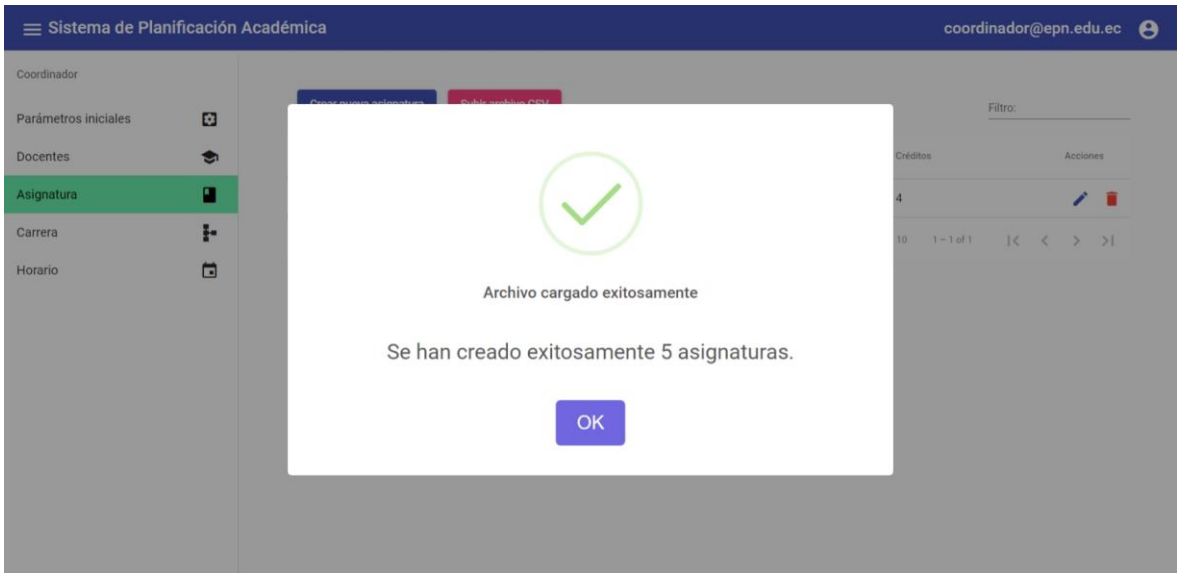
Pantalla 14: Creación exitosa de una asignatura



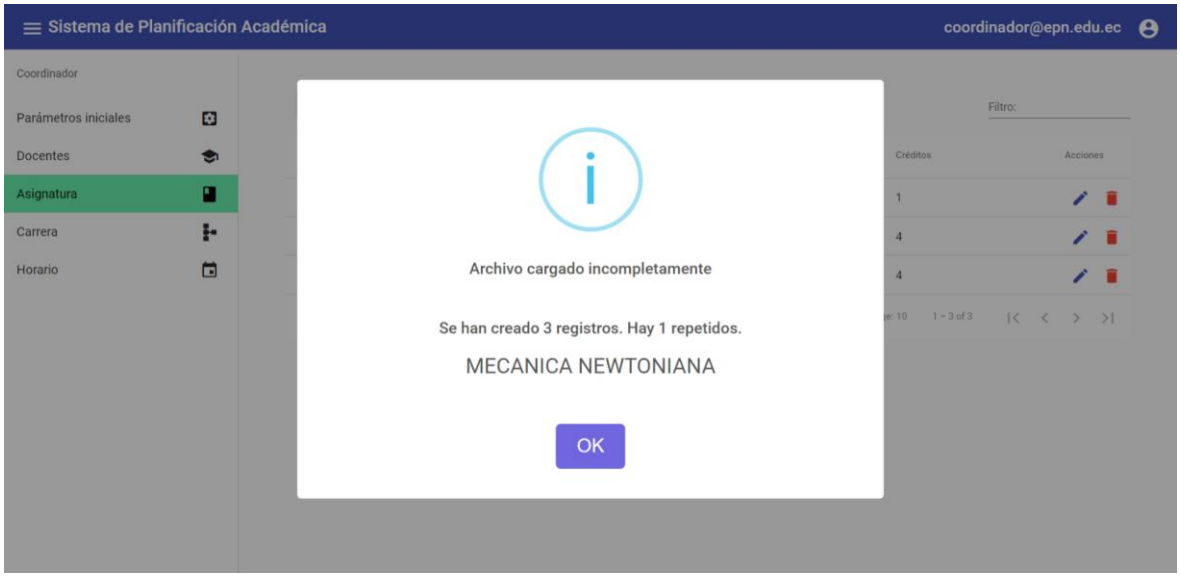
Pantalla 15: Creación fallida de una asignatura



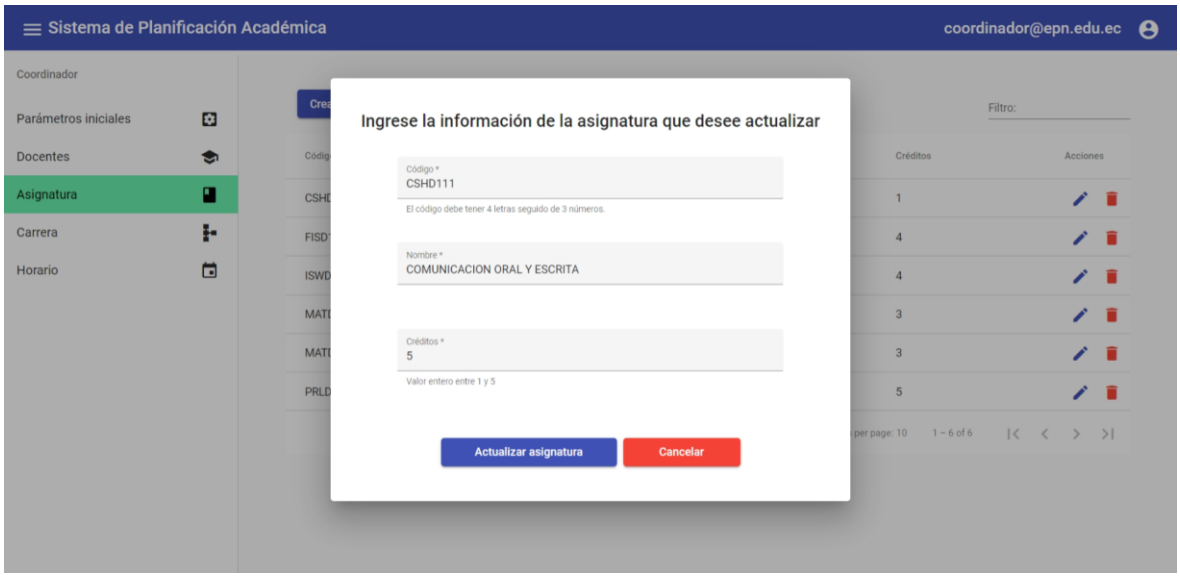
Pantalla 16: Creación exitosa de varias asignaturas por medio de un archivo *csv*



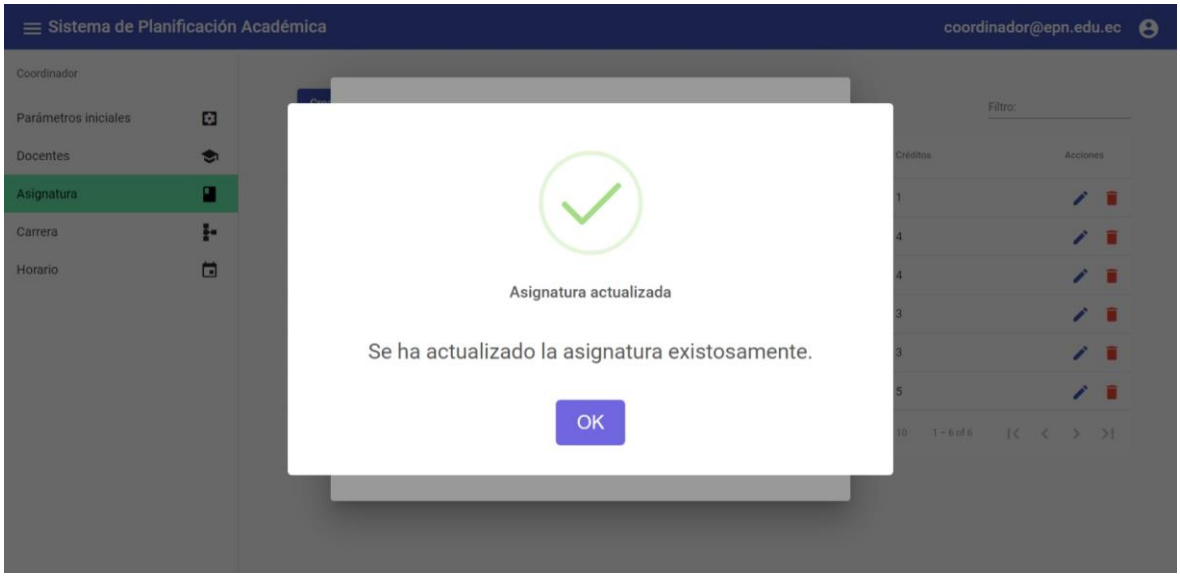
Pantalla 17: Creación fallida de varias asignaturas por medio de un archivo *csv*



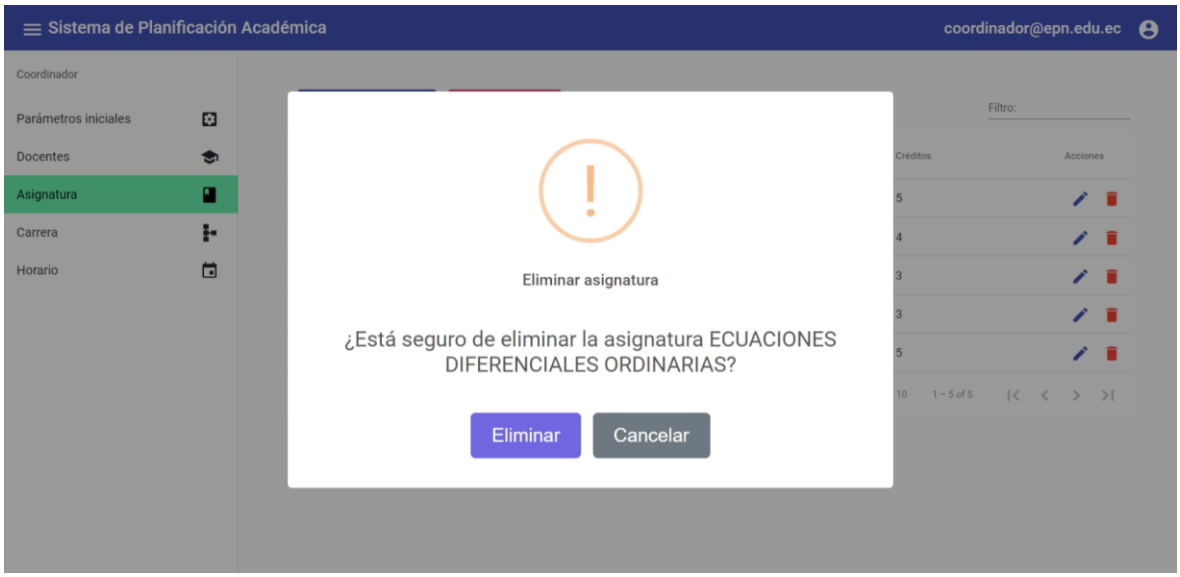
Pantalla 18: Actualización de una asignatura existente



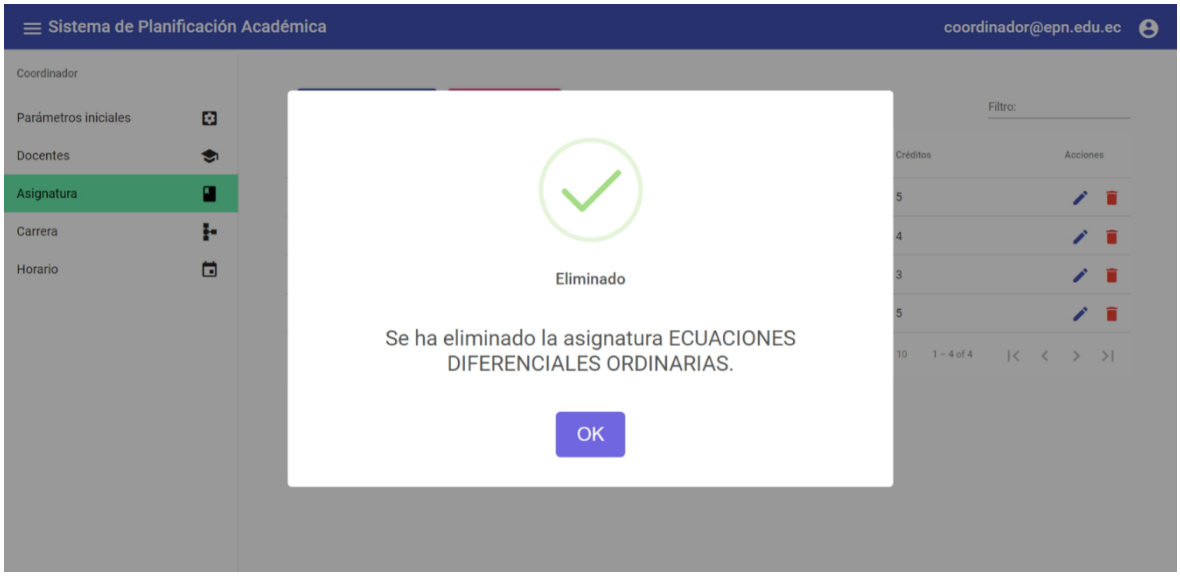
Pantalla 19: Actualización exitosa de una asignatura existente



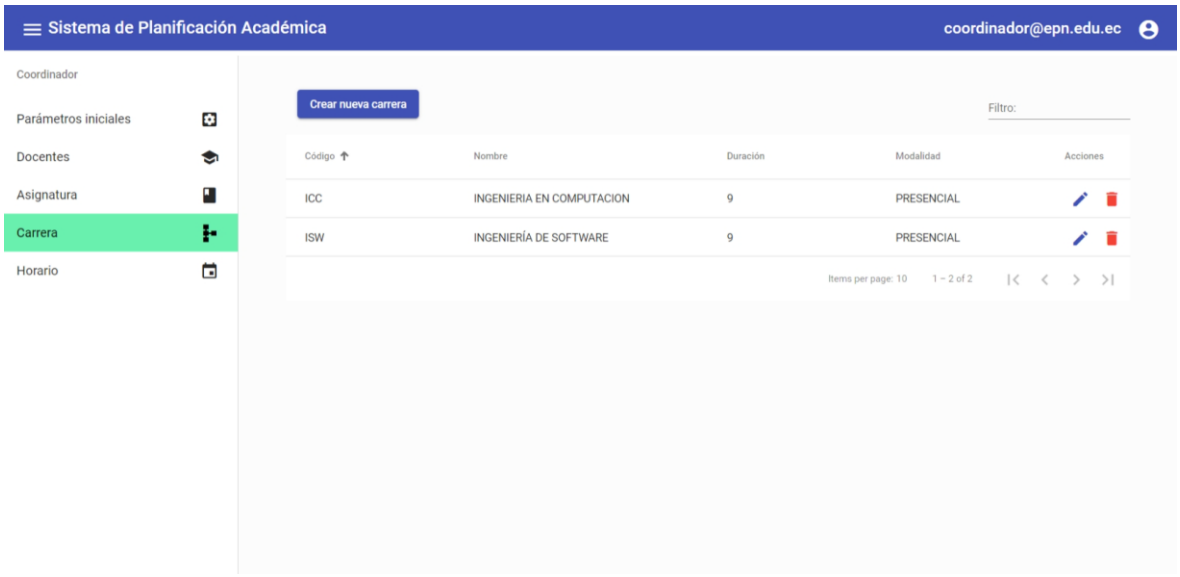
Pantalla 20: Eliminación de una asignatura existente



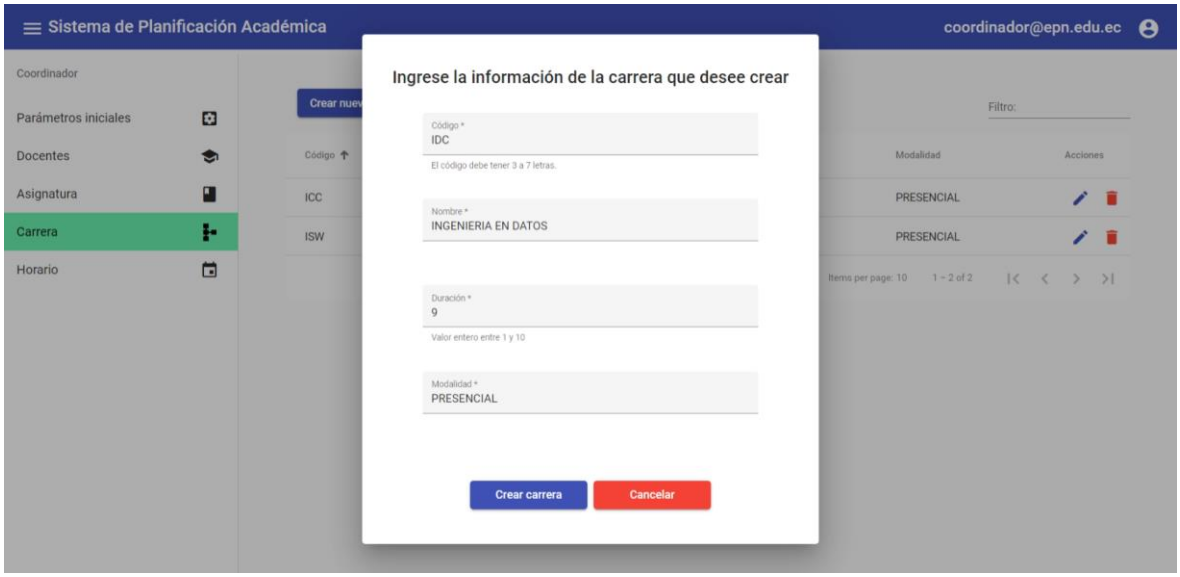
Pantalla 21: Eliminación exitosa de una asignatura existente



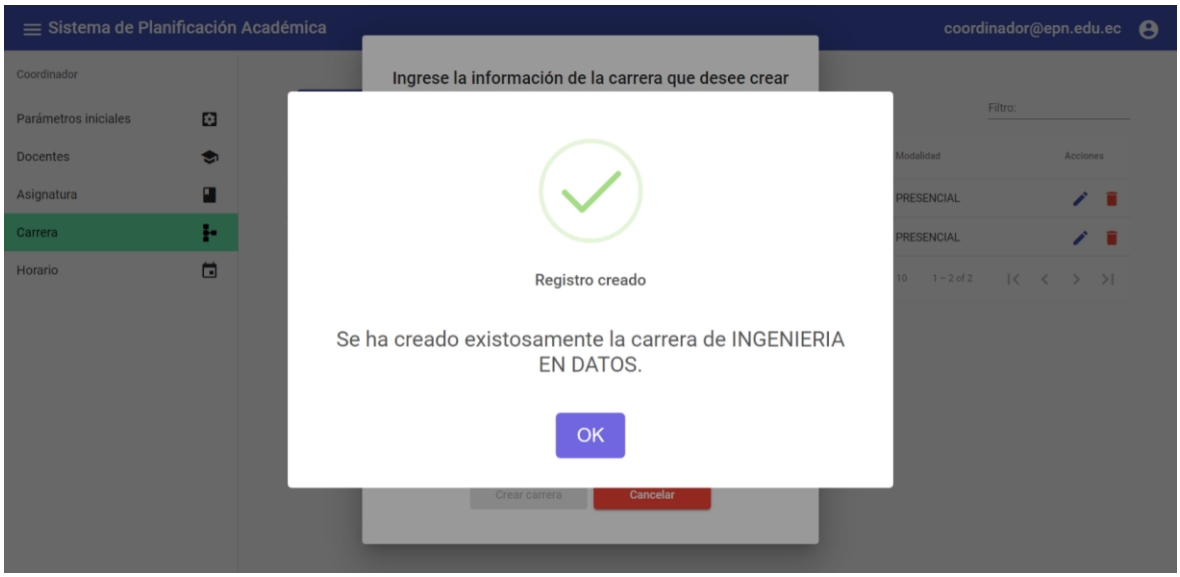
Pantalla 22: Visualización de carreras



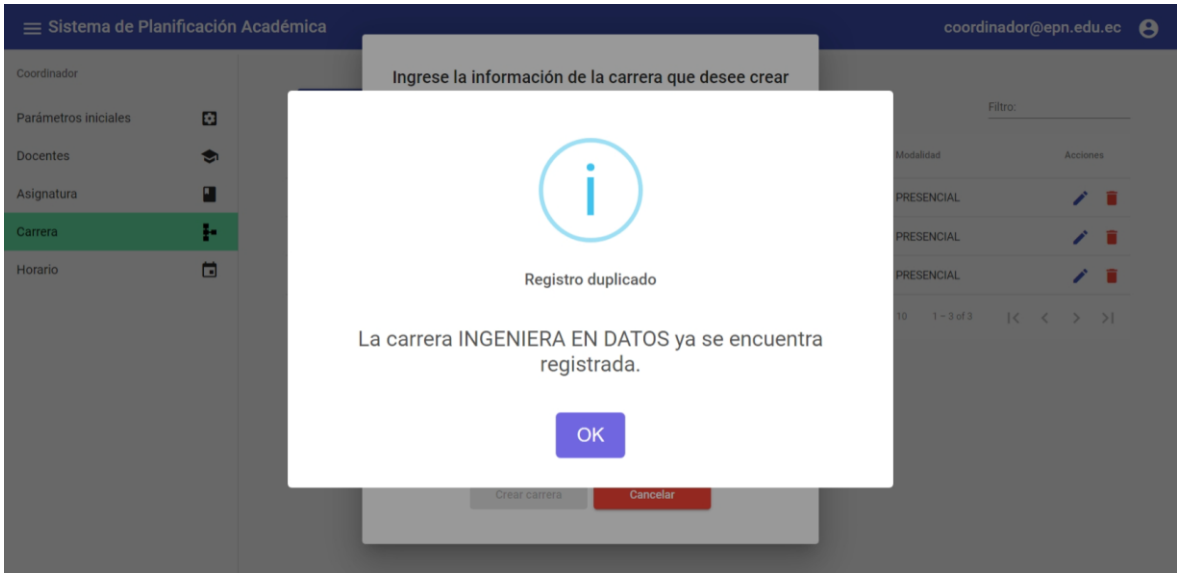
Pantalla 23: Crear una carrera



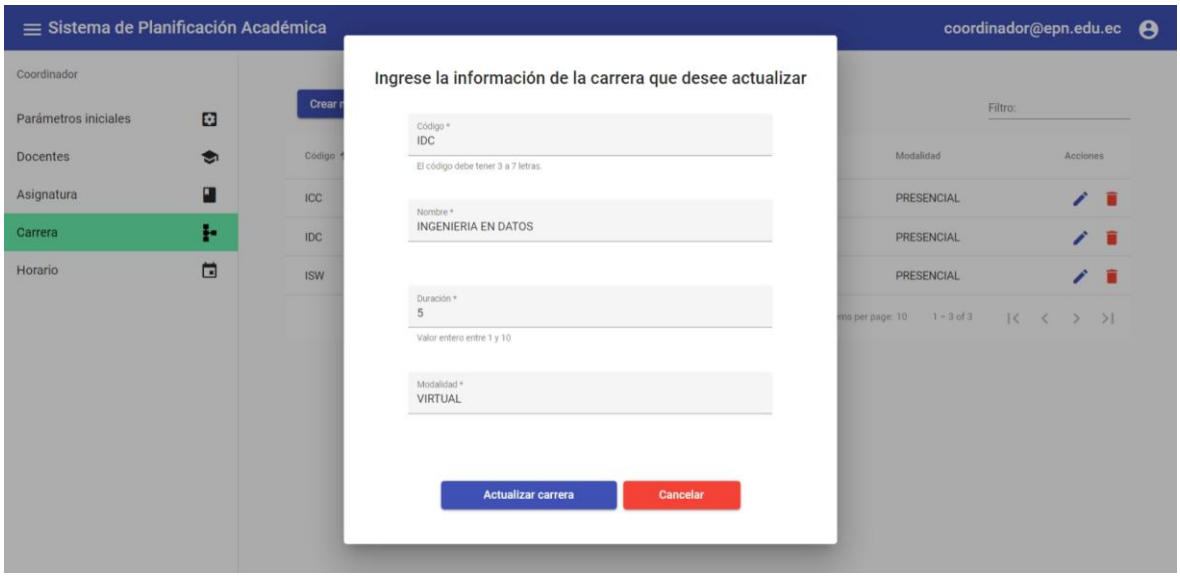
Pantalla 24: Creación exitosa de una carrera



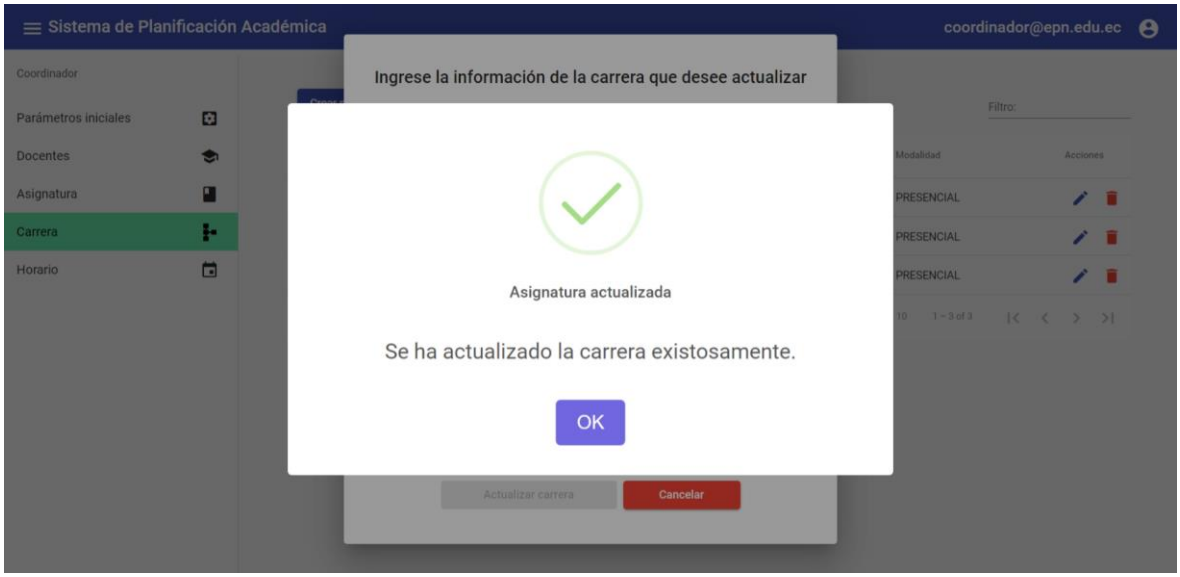
Pantalla 25: Creación fallida de una carrera



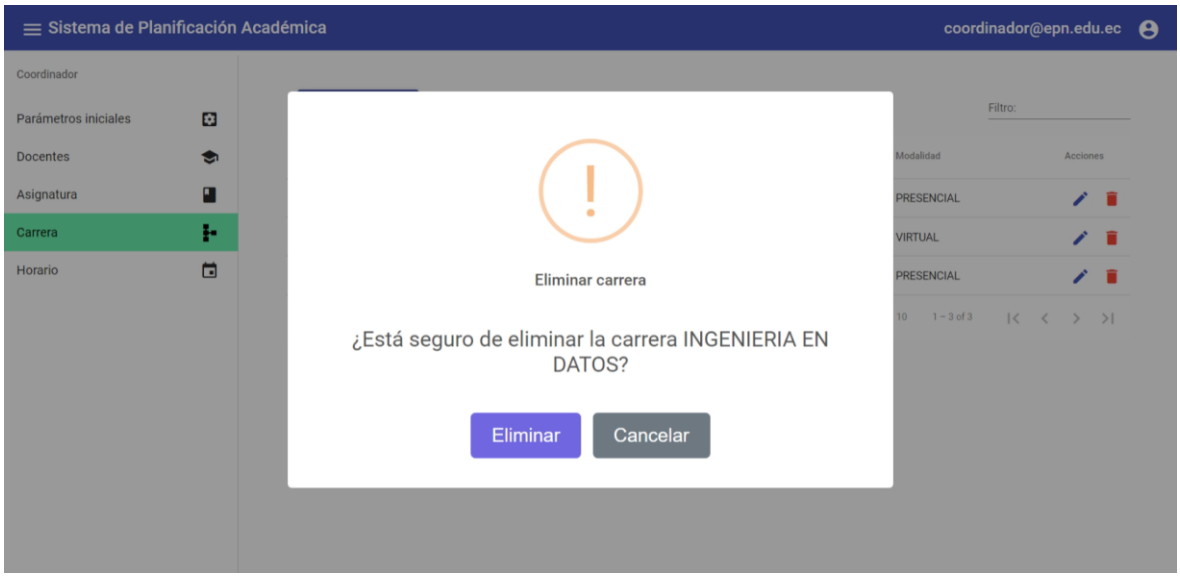
Pantalla 26: Actualización de una carrera existente



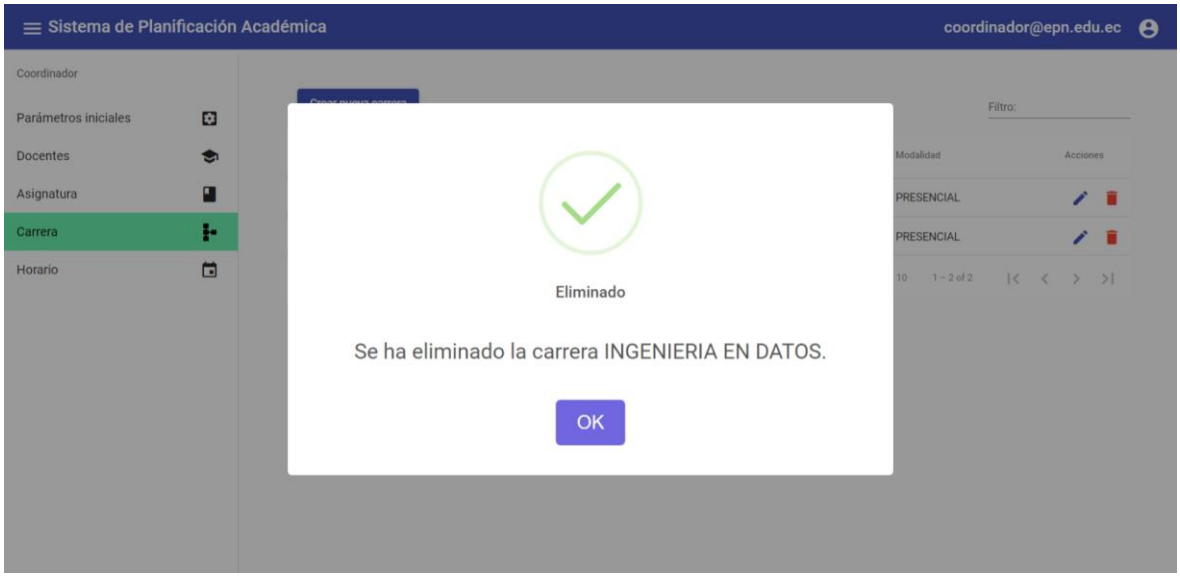
Pantalla 27: Actualización exitosa de una carrera existente



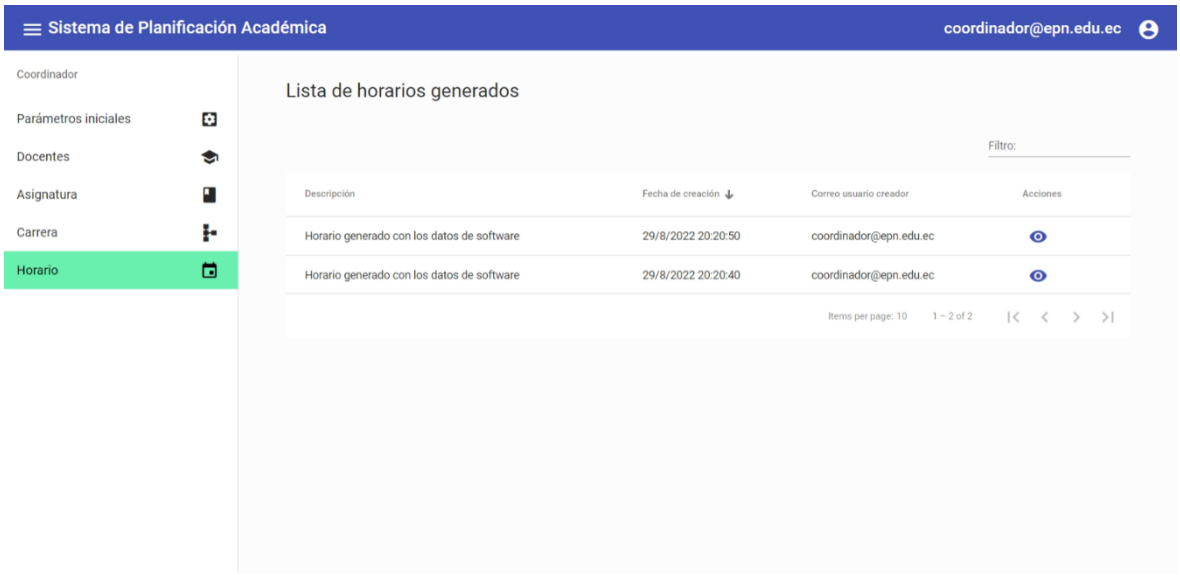
Pantalla 28: Eliminación de una carrera existente



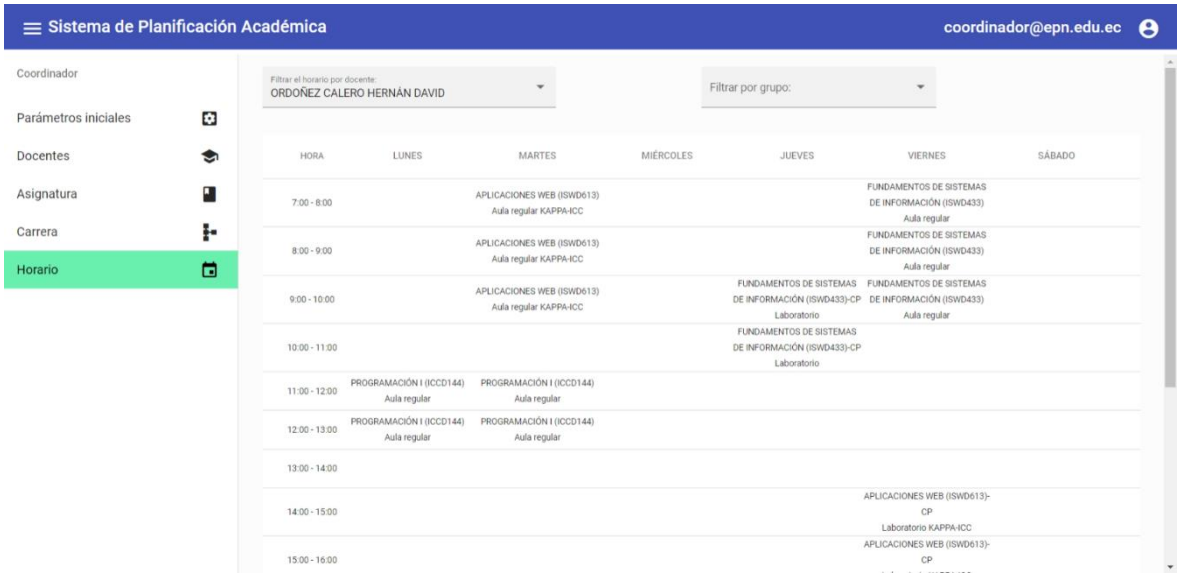
Pantalla 29: Eliminación exitosa de una carrear existente



Pantalla 30: Visualización de horarios generados previamente



Pantalla 31: Visualización de un horario seleccionado de un docente en específico



Pantalla 32: Visualización de un horario seleccionado de un grupo en específico