ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

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

GENERACIÓN Y DIFUSIÓN DE HORARIOS

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

LUIS ALEJANDRO LLANGANATE VALENCIA

info@alejandrollanganate.com

DIRECTOR: CARLOS EDUARDO ANCHUNDIA VALENCIA

carlos.anchundia@epn.edu.ec

Quito, febrero 2023

CERTIFICACIONES

Yo, Luis Alejandro Llanganate Valencia, 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.

ALEJANDRO LLANGANATE

Certifico que el presente trabajo de integración curricular fue desarrollado por Luis Alejandro Llanganate Valencia, bajo mi supervisión.

CARLOS ANCHUNDIA

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.

ALEJANDRO LLANGANATE

CARLOS ANCHUNDIA

DIANA LÓPEZ

MAHATMA QUIJANJO

DEDICATORIA

Para mis padres, Irene Valencia y Luis Llanganate, quienes siempre me enseñaron que únicamente el estudio me haría libre y que nunca tema en cumplir mis metas.

Para mis hermanas por su apoyo incondicional en toda mi etapa universitaria y su ánimo en los días más difíciles.

Para los estudiantes quienes conformaron durante mi presidencia el Club de Software EPN y la Asociación de Estudiantes de Ingeniería de Sistemas de nuestra institución.

Definitivamente dejamos una huella en nuestra Facultad que nunca será olvidada…

Alejandro Llanganate

AGRADECIMIENTO

When reality doesn't go
according to plan, you need
to adapt to reality, rather
than trying to force reality
to fit into your plan.

John Ferguson Smart

Extiendo un profundo agradecimiento al docente Carlos Anchundia por su guía en la elaboración de este trabajo de titulación. Sobre todo, por haberme permitido desaprender conceptos erróneos para comprender realmente la Ingeniería de Software.

Agradezco a los docentes y profesionales que influenciaron indirectamente en mi formación, especialmente a Galoget Latorre (CEO de Hackem) y Michael Salazar (Referente de tecnología en Rappi). Así también, a mis compañeros Diana López y Mahatma Quijano por ser parte de este trabajo de integración curricular.

Agradezco a toda mi familia: Luis, Irene, Isabel, Fernanda, Esperanza, Blanca, Robert, Alex, Daniel, Freddy y Manuel. Gracias por cuidarme desde pequeño y por sus grandes enseñanzas que me acompañarán el resto de mi vida.

A mi pareja Gabriela por ser mi baluarte estos años, por sus valiosas clases de economía y por haber confiado en mi desde "prepo".

A todos quienes he dejado por escrito mi agradecimiento, cuando lean esta página recuerden: **nunca se rindan**.

Agradezco a las autoridades de la Facultad de Ingeniería de Sistemas de la Escuela Politécnica Nacional por haber brindado su tiempo y recursos para llevar a cabo este trabajo.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

La planificación académica es un proceso realizado por cada unidad académica de la Escuela Politécnica Nacional. El subdecano de cada facultad está encargado de este proceso, obteniendo como salida los horarios de clases que son un insumo importante en el proceso de matriculación estudiantil. La Facultad de Ingeniería de Sistemas actualmente lleva a cabo este proceso utilizando el software de escritorio FET (Free Timetabling Software), donde se generan horarios considerando la información de docentes, asignaturas, espacios físicos, tipos de aula, niveles, grupos, restricciones de espacio y tiempo, entre otros. No obstante, este software no permite el acceso concurrente de múltiples usuarios lo que obliga a los involucrados a intercambiar archivos. Aquello es un problema que implica el reproceso de datos y la posible pérdida de información. Como solución se plantea el sistema SPA (Sistema de Planificación Académica), una aplicación web que reutiliza el motor del software FET mediante la integración de la línea de comandos FET-CL. De esta forma, con el componente "generación y difusión de horarios", los usuarios podrán acceder de forma concurrente, respaldar su información utilizando una base de datos, y generar horarios a partir de los datos presentes en el sistema. Además, podrán exportar estos horarios en formato EXCEL para difundirlos con los interesados en la planificación académica. Para lograr un sistema de valor y que cumpla estos objetivos se utilizó el desarrollo dirigido por comportamiento que conlleva aplicar un conjunto de buenas prácticas bajo una metodología ágil.

PALABRAS CLAVE: Generación de horarios, planificación académica, FET-CL, motor del software FET, difusión de horarios, desarrollo dirigido por comportamiento, BDD, Facultad de Ingeniería de Sistemas.

ABSTRACT

Academic planning is a process carried out by each academic unit of the Escuela Politécnica Nacional. The sub-dean of each faculty is responsible for this process, which produces schedules that are an important input in the student registration process. The Facultad de Ingeniería de Sistemas currently carries out this process using desktop software called FET (Free Timetabling Software), which generates schedules considering information about teachers, subjects, physical spaces, classroom types, levels, groups, space, and time constraints, among others. However, this software does not allow multiple users concurrent access, forcing those involved to exchange files. This is a problem that involves data reprocessing and possible loss of information. As a solution, SPA (Academic Planning System) is proposed, a web application that reuses the FET software engine by integrating FET-CL. In this way, with the "generation and dissemination of schedules" component, users will be able to work concurrently, using a database, and generating schedules based on the data present in the system. Additionally, the users will be able to export schedules to EXCEL format and share them with academic planning stakeholders. Behavior-driven development allows the project to achieve objectives by applying a set of best practices under an agile methodology.

KEYWORDS: Schedule generation, academic planning, FET-CL, FET software engine, schedule dissemination, behavior-driven development, BDD.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

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

Por otro lado, un sistema de planificación académica permite programar materias, docentes y asignaturas por carrera y semestre, considerando la capacidad máxima de estudiantes de cada curso [2]. Actualmente, los directores de cada carrera y el Subdecano de la Facultad de Ingeniería de Sistemas utilizan el software FET para el proceso de creación de la planificación académica [3]. El motor del software FET permite a través de su algoritmo heurístico la generación de horarios como también la creación de "actividades". En las que se asignan tanto docentes, horarios y asignaturas para los diferentes niveles [1]. No obstante, el uso del software FET presenta varias limitaciones con respecto al acceso concurrente de usuarios. Por lo tanto, los involucrados no pueden acceder simultáneamente a la información; es decir, los usuarios se ven obligados a intercambiar los archivos que genera este programa de escritorio para avanzar con sus actividades de planificación, lo cual implica reprocesos de datos y posibles pérdidas de información [3].

Por lo tanto, el proyecto está orientado a desarrollar una aplicación web de planificación académica para la Facultad de Ingeniería de Sistemas que permita el acceso concurrente entre usuarios que abarque y cumpla criterios de calidad. De este modo, el componente de "Generación y difusión de horarios" aprovechará el motor del software FET, utilizándolo para que funcione sobre un aplicativo web y que permita generar los horarios académicos por cada nivel o semestre considerando restricciones de aforo de las aulas, docentes y sus horas disponibles, asignaturas y número de estudiantes.

Por otro lado, estos horarios generados dentro del proceso deberán ser visualizados tanto por las partes interesadas como el Subdecanato de la Facultad [4]. Por esto, el componente también contempla desarrollar lo necesario para que los horarios generados puedan ser visualizados por los involucrados dentro del proceso de planificación. Esta visualización contará con los filtros respectivos para que sean de utilidad en la toma de decisiones en el contexto de la planificación académica de la FIS.

1.1 Objetivo general

Contar con una aplicación web multiusuario, usable e intuitiva, que gestione respaldos a nivel de base de datos, y permita generar y difundir los horarios académicos semestrales por carrera mediante la reutilización del motor del software FET, con el fin de evitar reprocesos y pérdidas de la información durante el proceso de planificación académica.

1.2 Objetivos específicos

1. Disponer de una arquitectura segura que permita respaldar la información mediante una base de datos para facilitar el acceso rápido a la información a cada usuario y así evitar pérdidas de la información.
2. Disponer de una aplicación web que permita el acceso concurrente de usuarios mediante su arquitectura distribuida, para que cada uno de los usuarios pueda realizar sus actividades de planificación académica simultáneamente.
3. Disponer de una aplicación web que contemple generar y almacenar los horarios académicos semestrales, mediante la integración del motor del software FET en el aplicativo, para evitar reprocesos innecesarios.

1.3 Alcance

El componente "Generación y difusión de horarios" comprende la entrega de un sistema web que reutilice el motor del software FET para agilizar la implementación y la generación de horarios académicos. Para conseguir aquello se implementará los módulos necesarios que permitan integrar otros módulos de la planificación académica como: parámetros iniciales, docentes, asignaturas, actividades, número de estudiantes, niveles, grupos y espacios físicos.

Para el proyecto se ha considerado una metodología ágil de desarrollo de software considerando que el proyecto necesita que las partes interesadas estén involucradas fuertemente en el proyecto. Esta metodología permite dar demostraciones del estado del aplicativo en ciclos de lanzamientos cortos promoviendo el interés de las partes interesadas. Por lo cual el subdecanato de la Facultad de Ingeniería de Sistemas ha participado activamente en el proyecto para sugerir y proveer nuevas ideas de mejora para el aplicativo.

Cada módulo del componente contempla buenas prácticas de desarrollo de software y pruebas automatizadas. Su ejecución cuenta con un enfoque a la funcionalidad y a la

entrega continua de valor real para el negocio. De modo que este proyecto empleará las diferentes prácticas de desarrollo de software del Desarrollo dirigido por comportamiento (*Behavior-Driven Development o BDD*) para la entrega de un sistema valioso y de mayor calidad. El proyecto también hará uso de prácticas y herramientas para una integración y entrega continua del sistema. De tal forma que el aplicativo pueda ser desplegado en los servidores de la Facultad de Ingeniería de Sistemas o en la nube en un ambiente de prueba.

Cabe recalcar que el presente trabajo no contempla la migración de datos de planificaciones académicas sin el uso del software FET, ni la operación del sistema en un ambiente de producción. Así también, el sistema no contempla restricciones personalizadas de actividades más que únicamente la información ingresada en los módulos presentes en el sistema.

2 MARCO TEÓRICO

En esta sección se presentan los diferentes temas relacionados al proyecto que fueron analizados en la revisión sistemática de este trabajo.

2.1 Planificación académica

La planificación académica, término conocido en inglés como “Academic Timeteable Scheduling”, es un proceso que consiste en la planificación, organización y control de espacios, personal o docentes, grupos de estudiantes y secuencia de actividades de acuerdo con ciertas restricciones [5]. Este proceso tiene como resultado la elaboración de horarios que relacionan docentes, aulas, paralelos o grupos, asignaturas, entre otros recursos. Los horarios resultantes sirven como un insumo importante a las instituciones de educación para su planificación en un ciclo académico.

Sin embargo, el principal reto de la planificación académica es la generación de horarios debido a la variedad de restricciones determinadas a considerar. Al tratarse de un problema de optimización en la mayoría de los casos se emplean algoritmos heurísticos como una solución para la generación de horarios [6]. Sin importar otras soluciones factibles, las instituciones de educación deben abarcar este problema ya sea en la organización de cursos o en la planificación de exámenes.

Existen otros desafíos en cuanto a la generación de horarios que según [7] implican considerar dos categorías de restricciones: fuertes y blandas. Las restricciones fuertes están sujetas a las limitaciones físicas como el foro máximo de un laboratorio o la cantidad de estudiantes de un grupo específico. Por otro lado, las restricciones blandas refieren a preferencias específicas que indican como la solución debe ser. Como por ejemplo la asignación personalizada de actividades académicas o el número de días laborales.

2.2 Free Timetabling Software (FET)

Es un software libre de escritorio que permite automáticamente generar horarios de instituciones académicas como universidades o colegios [8]. Este software puede ser utilizado en diferentes sistemas operativos como Windows, Linux y MacOS. También dispone de una interfaz de línea de comandos que viene en los ejecutables del software de escritorio para distribuciones Linux.

Los datos de entrada que utiliza el software FET para la generación de horarios deben ser agregados manualmente en el programa. Esta información está relacionada con elementos propios de una unidad académica que incluyen [9]:

1. Nombre de la institución y comentarios.
2. Días por semana y períodos por día, que refieren a los días laborables académicos y los intervalos de tiempo de su jornada habitual.
3. Profesores, asignaturas, aulas y edificios.
4. Etiquetas de actividad, que son un tipo de restricción para posteriormente aplicar restricciones a las actividades que únicamente cuenten con dicha etiqueta.
5. Años, que son el conjunto de estudiantes que cursan la misma etapa de su escolaridad. Por ejemplo, en la carrera de Ingeniería de Software de la Escuela Politécnica Nacional existen 9 niveles sin considerar el curso de nivelación. Cada nivel podría ser considerado con la denominación de “año” para el software FET.
6. Grupos y subgrupos, en donde los grupos refiere a una división ordenada de estudiantes que compartirán a futuro una actividad (un docente, una asignatura, en un espacio y en un horario determinado). De modo que la opción de grupos puede ser empleada para crear diferentes paralelos para un nivel especifico.
7. Actividades, que relacionan a una asignatura, un grupo de estudiantes, un docente y una duración en horas. Una actividad puede dividirse en subactividades y también puede contener una etiqueta como se mencionó previamente.
8. Restricciones, en donde a cada una le pertenece un porcentaje para ser considerada. Si el porcentaje de una restricción es del 100%, entonces el algoritmo debe considerarla como obligatoria. Para un porcentaje inferior la restricción podría o no aplicarse. Las restricciones básicas para el software FET son de tiempo y de espacio. Así también, es posible añadir restricciones blandas que permiten preferir una actividad en un horario y espacio en particular.

Como se mencionó en la sección 2.1, los programas relacionados a la generación de horarios usan algoritmos heurísticos para solucionar problemas de optimización. Es así como el software FET emplea un algoritmo heurístico que intercambia actividades recursivamente comenzando por las más difíciles. Esta implementación es conocida en el presente proyecto de titulación como el “motor del software FET.

FET-CL

La interfaz de línea de comandos del software FET, es una solución que permite aprovechar el motor del software FET para generar horarios con el uso de un comando. Para ejecutar correctamente el comando `fet-cl` se debe pasar un archivo con extensión `.fet` y haciendo uso del modificador `–inputfile`. El archivo `.fet` contiene etiquetas con una estructura específica que permite reunir los datos de entrada previamente mencionados que pueden ser visualizadas en la Figura 1.

Figura 1. Nombre de las etiquetas del fichero con extensión `.fet`.

Por otro lado, el resultado de ejecutar este comando es la generación de dos directorios:

1. ***logs:*** Guarda el detalle de la salida del comando, es decir, un registro si se ha llegado a completar correctamente el comando o si ha existido un error.
2. ***timetables****:* Contiene un directorio con archivos HTML y XML con los horarios generados. En caso de existir un error, únicamente se genera un directorio vacío sin ningún horario. Los archivos XML dentro de este directorio contienen los horarios generados por subgrupos, actividades y docentes.

2.3 Planificación académica en la Facultad de Ingeniería de Sistemas

El proceso de planificación académica en la Facultad de Ingeniería de Sistemas se lleva a cabo actualmente utilizando el software FET de escritorio. En este proceso participan el subdecanato de la Facultad de Ingeniería, los coordinadores de las carreras de Ingeniería de Software e Ingeniería de Ciencias de la Computación, la secretaria del subdecanato y la jefa de departamento [3]. Este proceso se lleva a cabo luego del primer bimestre del semestre en curso. Comienza con una encuesta inicial de preplanificación en la cual los estudiantes las asignaturas que se podrían inscribir en el semestre subsecuente, y finaliza con la generación de horarios para cada grupo de estudiantes de los niveles correspondientes.

Si bien el software FET es útil para la generación de horarios académicos, este presenta limitaciones para los coordinadores de las carreras. El software FET al ser un software de escritorio con un almacenamiento no distribuido, no permite el acceso concurrente de usuarios. Es decir, cada usuario debe intercambiar archivos con modificaciones del otro para continuar con sus actividades. Este intercambio implica reprocesos de datos, pérdidas de información en diferentes versiones que se generan a lo largo del proceso.

La generación de horarios no obstante es un aspecto fundamental de todo el proceso. Y pese a las limitaciones el software FET es utilizado para generar los horarios en base a los datos de entrada que se ingresan manualmente y otras restricciones. Estos horarios son luego socializados en la facultad y son enviados a la DGIP como un insumo importante para el proceso de matriculación de los estudiantes.

El proceso que actualmente lleva la Facultad de Sistemas busca ser mejorado con este proyecto para resolver las limitaciones que presenta el Software FET hoy en día. La creación de un sistema web que permita guardar los datos en una base de datos y pueda ser accedido por diferentes usuarios desde su navegador, soluciona el problema de concurrencia y reprocesos previamente explicado. Lo esencial dentro de la resolución de este ejercicio es ocupar el componente más importante del FET: su motor o algoritmo de generación de horarios en base a restricciones.

2.4 Behavior-Driven Development (BDD)

Según John Ferguson el desarrollo dirigido por comportamiento (*Behaviour-Driven Development* en inglés) es un conjunto de prácticas ágiles de Ingeniería de Software que permiten la entrega de un software de calidad que aporte valor al negocio. A diferencia

del Desarrollo dirigido por pruebas (*Test Driven Development* en inglés), BDD se centra en lo que debería hacer el código que genere valor para el negocio, en vez de probar cada implementación particular del código [10].

La Figura 2 ilustra el proceso que conlleva el Desarrollo dirigido por comportamiento que empieza con identificar las metas del negocio para luego definir *features* (características) que permitan cumplir estos objetivos comerciales. A partir de ellos se crean ejemplos concretos que, con un vocabulario común, permiten ser comprendidos por los usuarios finales y el equipo de desarrollo. Permitiendo obtener requisitos claros, acoplados a las metas de negocio y fáciles de entender para los involucrados.

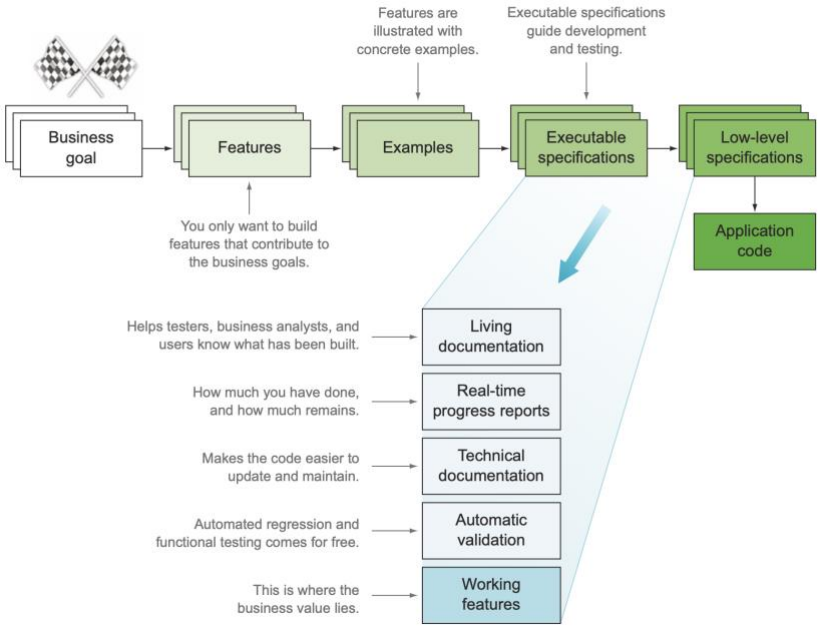


Figura 2. Principales actividades del BDD. Tomado de [10].

Gherkin

Las herramientas de BDD generalmente utilizan un lenguaje llamado Gherkin que permite plasmar los requisitos en un archivo y ejecutar pruebas automatizadas de dichos requisitos. En este lenguaje se establecen:

- ***Features* o características:** que describen una pieza tangible y entregable de funcionalidad que pueden ser escritos como una historia de usuario.
- ***Scenarios o escenarios:*** que son ejemplos concretos de cómo funciona una característica. Los ejemplos, como se muestra en la Figura 3, utilizan el formato *Dado… Cuando… Entonces…* para poder describir un ejemplo como un criterio de aceptación.

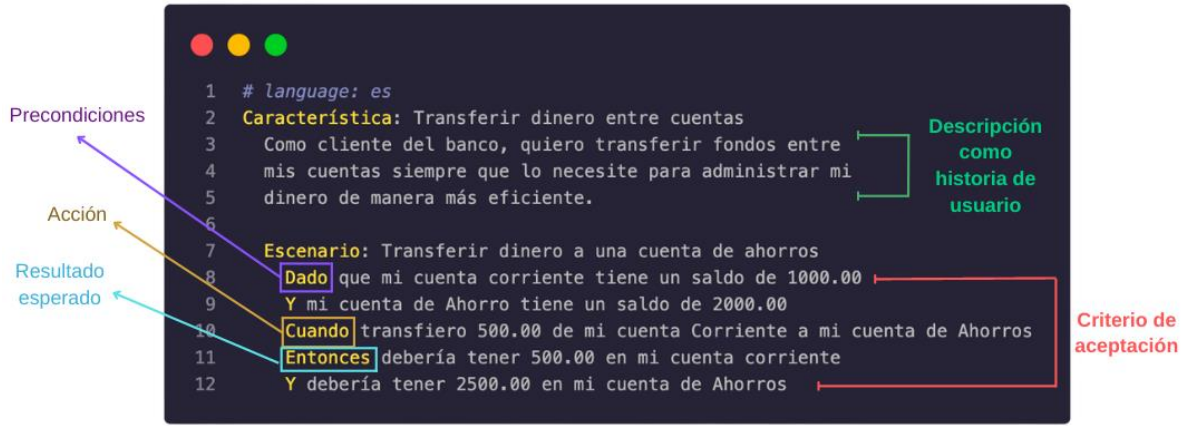


Figura 3. Ejemplo de una característica y un escenario en lenguaje Gherkin.

Los elementos mencionados previamente permiten a los desarrolladores comprender qué es lo que deben construir. Particularmente en BDD los ejemplos descritos en cada escenario se pueden asimilar como criterios de aceptación. Los equipos convierten estos criterios en especificaciones ejecutables que son pruebas automatizadas que verifican que una porción de código cumpla con el requisito de negocio.

Cucumber de Smartbear es una herramienta útil para validar especificaciones ejecutables de un aplicativo desarrollado. Esta herramienta permite crear definiciones de *Steps* que conectan cada *Step* en lenguaje Gherkin con el código de programación [11]. Por lo tanto, una definición de un *Step* es la acción que se debe implementar en dicho paso. Un ejemplo de aquello se puede visualizar en la Figura 4.

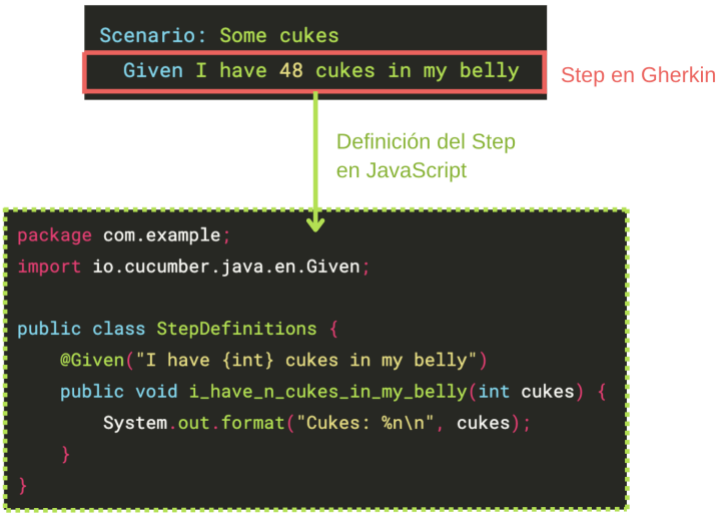


Figura 4. Ejemplo de la definición de un Step. Tomado de [12].

3 METODOLOGÍA

Para la realización del componente de generación y difusión de horarios se empleará el desarrollo dirigido por comportamiento. Como se mencionó en la sección 2, este tipo de desarrollo compone diferentes prácticas de software que permiten entregar un sistema alineado a los objetivos comerciales bajo una metodología ágil. De este modo, al inicio del proyecto se establecerán reuniones con los involucrados en el proceso de planificación para comenzar a entender el dominio del problema y posteriormente identificar los requisitos principales plasmados en *features*. Posteriormente se diseñarán pruebas que guiarán el desarrollo del sistema con la definición de *steps* que se llevarán a cabo en diferentes sprints.

3.1 Reuniones

En una etapa temprana del proyecto se realizaron tres entrevistas para entender el problema de planificación académica de la facultad y conocer los requisitos funcionales del sistema a desarrollar. Los profesionales que participaron como entrevistados fueron:

- José Lucio, subdecano de la Facultad de Ingeniería de Sistemas hasta el periodo 2022-B.
- Julio Sandobalin, coordinador actual de la carrera de Ingeniería en Software.
- Josafá Aguiar, coordinador actual de la carrera de Ingeniería en Ciencias de la Computación.
- Alex Ulloa, autor de la tesis “Desarrollo de una aplicación web de Planificación Académica para el Subdecanato de la Facultad de Ingeniería de Sistemas”.

Los aspectos más importantes de cada entrevista, que permitieron analizar requisitos y tomar decisiones, se detallan a continuación:

1. **Entrevista a los coordinadores de las carreras de computación y software, y al subdecano de la facultad:** En esta entrevista realizada por Carlos Anchundia (ANEXO I), director del presente trabajo, se pudo comprender qué funcionalidades utilizan los coordinadores del software FET para la planificación semestral de la facultad. Así también las limitaciones y problemas que se presentan cuando utilizan el software de escritorio FET que han sido contemplados en los objetivos de este trabajo de integración curricular.

2. **Entrevista al subdecano de la facultad:** La entrevista realizada por el equipo (ANEXO II) permitió comprender el proceso de pre-planificación que se realiza luego de la entrega de notas del primer bimestre en función de una encuesta. Así también que la tesis presentada Alex Ulloa no fue en ningun momento utilizada por los involucrados y por lo cual el subdecano desconocía del estado de dicho proyecto. Debido a aquello el equipo decidió agendar una entrevista con el tesita.

3. **Entrevista al tesista Alex Ulloa:** En esta entrevista (ANEXO III) el tesista afirmó que en su trabajo de titulación no existió ningún tipo de seguridad, ni buenas prácticas de desarrollo o pruebas, y que existe una documentación limitada del sistema ya que varios de los enlaces estaban rotos. Por otra parte, indicó que había utilizado un CLI del software FET para utilizar su motor, no obstante, no disponía de documentación que respalde su proceso de instalación.

Con estos insumos de análisis el equipo inicial con formado por los estudiantes Mahatma Quijano, Diana López y Alejandro Llanganate, decidió que era necesario realizar nuevamente el sistema con buenas prácticas de desarrollo y pruebas que aseguren la aceptación del usuario con los requisitos planteados. Así tambíen que el proyecto contemple mantener el uso del CLI del software FET y generar la documentación respectiva para entender su instalación y funcionamiento dentro del sistema. Por último, que quien ocupe el rol de Subdecano de la Facultad sea el Product Owner del proyecto y que el Scrum Master sea el estudiante Alejandro Llangante para gestionar el proceso Scrum y apoyar al equipo de desarrollo.

3.2 Requisitos

Especificación de requisitos

En base al análisis realizado en la sección 3.1 se establecieron diferentes reuniones del equipo para especificar los requisitos en forma de historias de usuarios. Para esto primero se comprendió los componentes esenciales del sistema en base a los objetivos de negocio identificados. Cada *feature* se tituló en base a la descripción la cual en su mayoría una historia de usuario. La descripción de cada uno de los *features* representados en la Tabla 1 son las historias de usuario del componente a desarrollar en este trabajo. En general, diferentes *features* fueron designados a los miembros del equipo gracias a que las historias de usuario cumplían con el principio INVEST (Independiente, negociable, valiosa, estimable, pequeña y comprobable).

Tabla 1. Features del componente “generación y difusión de horarios”.

Identificador	Título del feature	Descripción
A1	Parámetros iniciales	Como coordinador deseo establecer los parámetros iniciales de la planificación semestral como por ejemplo el semestre en curso, los días hábiles laborables y su horario en intervalos de horas del día, el nombre de la facultad y sus tipos de aulas, para que posteriormente al generar los horarios estos cumplan con las restricciones iniciales y se evite obtener horarios no aplicables en la modalidad de trabajo de la institución.
A2	Creación de actividades	Como coordinador deseo poder registrar una actividad que permita relacionar una asignatura, con un grupo de estudiantes, con un docente y una duración para que sean tomados en cuenta en la generación de horarios.
A3	Generación de horarios con el software FET	El sistema necesita enviar los datos de entrada al software FET en un formato de etiquetas con extensión .fet para obtener como salida los horarios de cada nivel académico.
A4	Difusión de horarios	Como subdecano deseo difundir los horarios académicos que fueron previamente generados para compartirlos a los involucrados en la planificación académica como docentes, secretaria y jefa de departamento.

Estimación

Se utilizó la técnica llamada Planning Pocker para estimar el esfuerzo de las historias de usuario o ítems del Product Backlog del proyecto. Esta técnica se basa en el consenso y para realizarla el equipo definió un rango de la escala de Fibonacci para dar un tamaño específico a cada historia. El rango definido fue del 1 al 3 y su interpretación es la siguiente:

- **Valor de 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.
- **Valor de 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.
- **Valor de 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).

La reunión del equipo para realizar la técnica Planning Pocker se llevó a cabo de forma virtual utilizando la herramienta “Planning Poker Online” disponible en internet. El resultado de emplear está técnica permitió posteriormente realizar una asignación justa de cada feature a los miembros del equipo. En el caso de las historias de usuario del presente componente se obtuvieron las estimaciones de esfuerzo representadas en la Tabla 2.

Tabla 2. Valores de estimación de las historias.

Historia	**Estimación**
A1	2
A2	2
A3	3
A4	3

De modo que las historias de usuario relacionadas a la generación de horarios con el software FET y la difusión de horarios son las que presentan un tamaño mayor debido a su complejidad. Por otro lado, al finalizar la técnica de Planning Pocker el equipo pudo obtener un mejor entendimiento de las historias de usuario planteadas, y a su vez descubrir la mejor forma de asignarlas entre los miembros del equipo.

Arquitectura del sistema

La arquitectura propuesta del sistema contiene los componentes representados en la Figura 5. Estos componentes funcionan sobre un servidor Ubuntu cuya versión es la 20.04 que actualmente dispone de actualizaciones y un gran soporte de la comunidad.

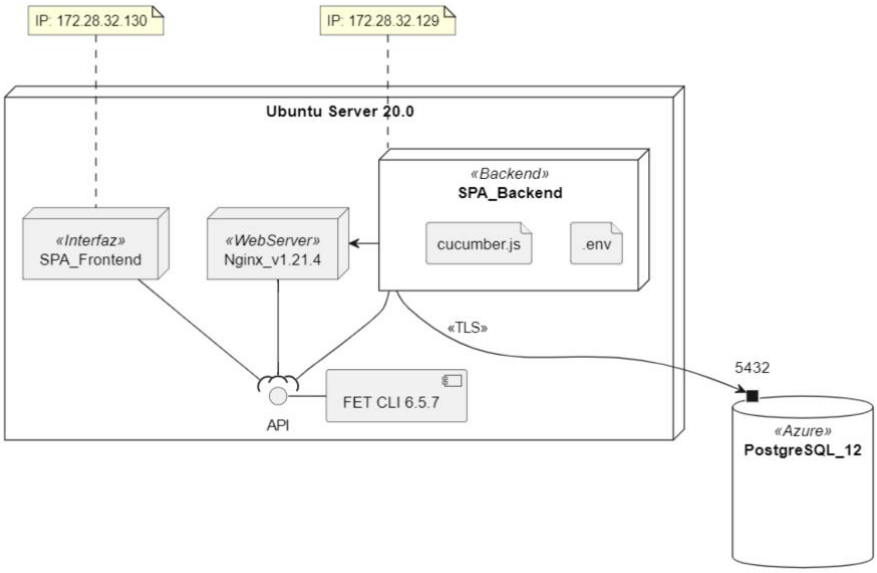


Figura 5. Diagrama de arquitectura del sistema.

Dentro de este servidor se tiene los componentes:

- **SPA_Frontend:** Un componente desarrollado con el framework Angular que permitió crear una aplicación web SPA que renderice páginas web dinámicas consumiendo la API REST desarrollada. Además, este componente utiliza Angular Material para que los componentes visuales del sistema sean responsivos y tengan un diseño y comportamiento preestablecido de otros sistemas web.
- **SPA_Backend:** Un componente desarrollado con el framework Nestjs que permite exponer una API REST que contenga los métodos respectivos de para lógica de negocio. Este componente necesita de dos archivos de configuración para su funcionamiento. El primero es el archivo “cucumber.js” que permite establecer configuraciones necesarias para las especificaciones ejecutables. El segundo es un archivo oculto denominado “.env” que contiene las variables de entorno necesarias para el funcionamiento de la API.
- **PostgreSQL_12:** Es una instancia de base de datos en Postgres versión 12, alojada en la nube con el uso del servicio Azure Database de la plataforma Azure.
- **FET-CL:** es el CLI del software FET con la versión 6.5.7, previamente compilado e instalado localmente en el servidor.

El servidor Ubuntu de esta arquitectura fue proporcionado por la Facultad de Ingeniería de Sistemas y es una máquina virtualizada o VDI que debe ser accedido vía SSH. Por

otro lado, es posible acceder al componente SPA_Frontend y SPA_Backend en el ANEXO IV.

3.3 Diseño dirigido en pruebas

En esta fase se contempló definir los criterios de aceptación de cada historia de usuario por medio de la creación de escenarios con ejemplos concretos para validar los *features* establecidos. De modo que fue necesario escribir los escenarios de cada *feature* considerando redactarlos antes de la fase codificación y centrándose en el comportamiento de la historia antes que detalles secundarios que no podrían aportar ningún valor a los objetivos comerciales. El resultado de esta fase fueron las especificaciones ejecutables del sistema con un acuerdo de los criterios de aceptación para empezar con la implementación. A continuación, se muestran los escenarios creados para los *features* descritos en la sección 3.2 del presente documento.

Parámetros iniciales

Dentro del proceso de planificación académica es fundamental establecer la jornada laboral académica, los tipos de aulas y facultades. Al establecer la jornada laboral se crean intervalos de una hora que son utilizados para la generación de horarios académicos. Estos aspectos fueron considerados como parámetros iniciales del sistema y de esta forma se crearon los escenarios plasmados en la Figura 6 que buscan validar estos objetivos de negocio.

```
Escenario: Se establece el horario laboral de la institución
    Dado el semestre '2022-A' con estado "PLANIFICACION_EN_PROGRESO",
    Cuando se registre el día laboral "LUNES" en el horario de "07:00" a "21:00" horas con hora de almuerzo a las "13:00",
    Entonces se obtienen 13 horas laborales e intervalos de una hora.

Escenario: Se registran tipos de aulas para diferentes unidades académicas
    Dado que existe la unidad académica "FACULTAD DE INGENIERÍA DE SISTEMAS",
    Y otra unidad académica con nombre "FORMACIÓN BÁSICA"
    Cuando se registre el tipo de aula "LABORATORIO" para la primera unidad,
    Y el tipo de aula "AULA REGULAR" para la segunda unidad académica,
    Entonces la segunda unidad acedémica dispone de 0 laboratorios.
```

Figura 6. Escenarios del feature “Parámetros iniciales”.

Creación de actividades

La creación de una actividad, como se explicó en la sección 2.2, permite relacionar un docente, con una asignatura, con un subgrupo, un tipo de aula y una duración. Estos registros de actividades son un insumo importante para la generación de horarios de software FET en base a su algoritmo heurístico. Es por ello que es importante registrar

diferentes actividades por cada docente como se representa en el escenario de la Figura 7.

```
Esquema del escenario: Se registra una actividad con una duración correcta y errónea
    Dado que existe un docente con correo "<correo_docente>",
    Y la carrera "<nombre_carrera>" con código "<codigo_carrera>" con modalidad "presencial",
    Y la asignatura "<nombre_asignatura>" con código "<codigo_asignatura>" de "<num_creditos>" créditos,
    Y un grupo con código "<codigo_subgrupo>" del nivel "<nombre_nivel>" de la carrera previamente mencionada,
    Y un tipo de aula con nombre "<tipo_aula>" en la facultad con nombre "<facultad>",
    Cuando se intenta registrar una actividad indicando un número de estudiantes de "<numero_estudiantes>" y una duración semanal de "<duracion>" horas,
    Y se valida la duración de dicha actividad,
    Entonces se obtiene el mensaje "<mensaje>".
```

Figura 7. Esquema del escenario del *feature* “Creación de actividades”.

Generación de horarios con el software FET

A partir del análisis realizado se pudo identificar que un requerimiento de mucho valor era reutilizar el motor del software FET en el sistema web. Este requerimiento podría parecer técnico, sin embargo, la reutilización del motor del software FET permite que se puedan generar los horarios académicos. Estos horarios son justamente la salida del proceso de planificación académica por lo cual son uno de los elementos más importantes del proceso.

Por otra parte, el motor del software FET debe utilizar datos de entrada y restricciones para la generación de horarios. Los datos de entrada deben ser los datos presentes en el sistema ingresados por los usuarios finales previamente. Como se explicó en la sección 2.2 el comando o CLI del software FET recibe como parámetro un archivo `.fet` con toda esta información. Luego el motor del software se encargará de realizar las validaciones correspondientes para obtener como salida un conjunto de archivos dentro de la carpeta *timetables* con el horario generado. Por lo tanto, lo más fundamental es probar que la API reutiliza el motor del software FET por lo que se definió el siguiente escenario de la Figura 8. Este escenario pese a ser corto implica varios pasos técnicos, configuraciones e implementaciones a seguir para cumplirlo.

```
Escenario: Se genera los horarios con el motor del software FET
    Dado el archivo de planificación "HorariosFIS_2022B_v45_horarios.fet",
    Cuando se reutilice el motor del software FET desde la API,
    Entonces se obtendrán los horarios por subgrupo generados en etiquetas XML.
```

Figura 8. Escenario del *feature* “Generación de horarios con el software FET”.

Difusión de horarios

Los horarios generados deben ser almacenados en el sistema para que posteriormente puedan ser aprobados y difundidos a los diferentes involucrados. La forma de difusión propuesta consistió en que un horario consultado por el usuario final pueda ser descargado en formato Excel desde el navegador. Esta solución fue socializada directamente con la Product Owner y permite disponer de un archivo con un horario por docente o grupo para así compartirlo en diferentes medios digitales. Por lo tanto, no se estableció un escenario debido a que esta solución tiene que ver con el uso de una librería de *frontend* con una funcionalidad ya implementada.

3.4 Implementación

En esta sección se describen las actividades realizadas para implementar los *features* descritos en la sección 3.2. En esta fase se llevó a cabo un total de siete sprints con una duración de tres semanas. Inicialmente en el periodo 2022-A se propuso únicamente tres sprints, sin embargo los resultados obtenidos no permitieron conseguir los objetivos propuestos para entregar un sistema de valor. De modo que se llevó a cabo más sprints en el periodo 2022-B hasta alcanzar un entregable que cumpla los objetivos propuestos.

Sprint 0

El propósito de este sprint fue analizar las historias con cual cada uno de los miembros del equipo empezaría a trabajar. Así también, preparar el entorno de desarrollo y disponer de otras herramientas para la organización de las actividades del equipo. Por ejemplo, se utilizó la herramienta Notion para disponer de un sitio web (Figura 9 y ANEXO V) donde el equipo pueda documentar y organizar sus actividades a lo largo del proyecto. Esta herramienta también permitió personalizar el formato de las páginas del proyecto para crear un espacio de trabajo basado en el framework Scrum.

Figura 9. Sitio web en Notion creado para el proyecto.

Por otro lado, el equipo utilizó la plataforma GitHub para crear dos repositorios de código fuente del ANEXO IV para el desarrollo de los componentes: “SPA_Frontend” y “SPA_Backend”. Por consiguiente, se utilizó el software Git para el control de versiones y

se estableció seguir la convención “Conventional Commits 1.0.0” para disponer de un historial de *commits* entendible y descriptivo.

Con relación a la implementación del proyecto el equipo acordó utilizar el editor de código Visual Studio Code con el uso de la extensión Cucumber. Esta extensión permitió generar definiciones de *steps* en base a los escenarios propuestos con el lenguaje de programación TypeScript. Por otra parte, fue necesario crear un archivo de configuración para el uso de Cucumber en el proyecto utilizando el framework Nestjs como se visualiza en la Figura 10.

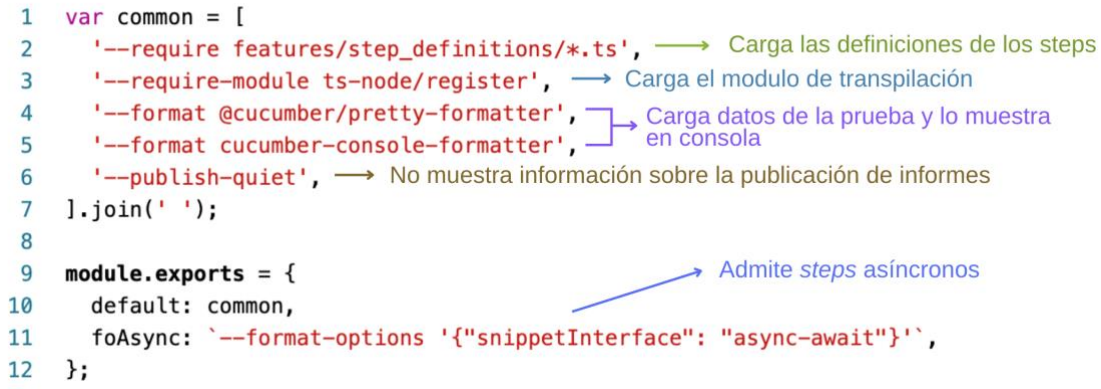


Figura 10. Drescpción del archivo “cucumber.js”

Al finalizar este sprint todos los integrantes del equipo tuvieron su entorno de desarrollo preparado y su planificación inicial lista para el siguiente sprint. Sin embargo, existió una gran incertidumbre de cómo se podría construir el sistema con las tecnologías propuestas y bajo las prácticas que componen el desarrollo dirigido por el comportamiento. Por lo tanto, los miembros que menos conocimientos tenían decidieron capacitarse individualmente en el desarrollo de APIs REST con el framework Nestjs y el desarrollo dirigido por el comportamiento.

Sprint 1

El objetivo de este sprint fue implementar la funcionalidad del *feature* A1 tanto en el componente SPA_Backend y SPA_Frontend. De esta forma el entregable sería las especificaciones ejecutables del *feature* y una demostración del sistema en el módulo de parámetros iniciales. No obstante, existían obstáculos todavía presentes como el desconocimiento de cómo crear correctamente la definición de steps utilizando el framework Nestjs.

Para empezar el equipo comenzó definiendo una estructura específica para los archivos *feature* y la definición de *steps* dentro del proyecto SPA_Backend. Esta estructura se puede visualizar en la Figura 11 con una convención de nombres de los archivos propuesta por el propio equipo. De esta organización se estableció el directorio "documents_test" para ficheros externos como archivos de datos que podrían ser necesarios para ejecutar los *steps*. Por otro lado, el archivo "beforeStep_afterStept.ts" contiene las configuraciones necesarias para crear y finalizar una instancia del proyecto exclusivamente para pruebas.

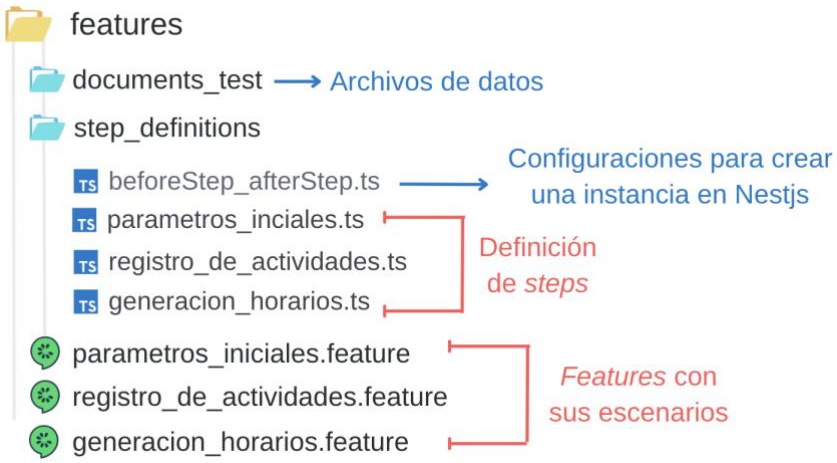


Figura 11. Organización de archivos para pruebas.

Con la definición de steps se llevo a cabo la implementación de servicios y entidades que cumplan los objetivos del *feature*. Inicialmente en la definición de los steps se utilizaba la base de datos de la aplicación para la creación de datos de prueba. No obstante, esta no fue una buena práctica puesto que si el sistema estuviera ya con datos reales y en ejecución se necesitaba obligatoriamente eliminar los datos creados en la prueba para la ejecución de la misma operando así directamente en la base de datos. Por lo tanto, la definición de los steps se mejoraron en el transcurso del entendimiento del proyecto y evitando antipatrones del Desarrollo dirigido por el comportamiento. Finalmente, se llevó a cabo con éxito la ejecución de la prueba (Figura 12) donde existieron ocho steps que guiaron el desarrollo y que permitieron validar los dos escenarios inicialmente propuestos.

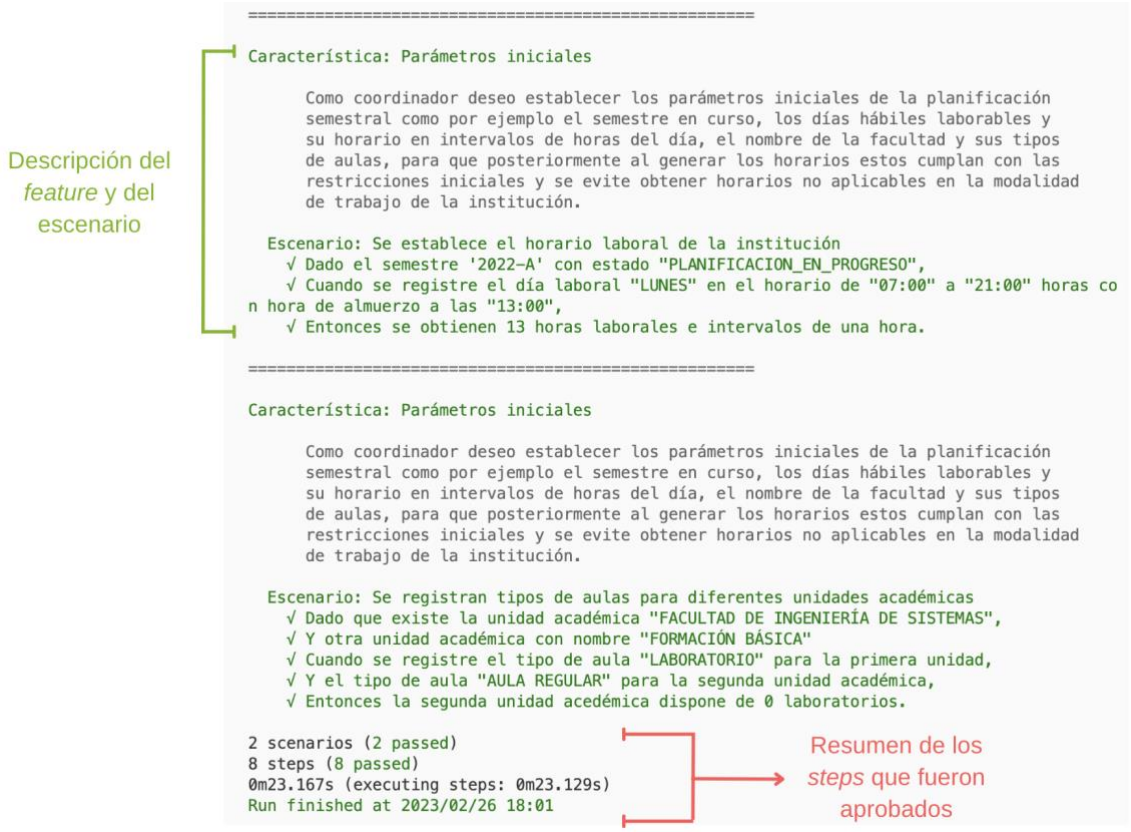


Figura 12. Ejecución de la prueba del feature A1.

En este *sprint* también se realizaron otras actividades para documentar las rutas de la API y establecer un método de autenticación, y autorización de usuarios en el sistema. De esta forma se implementó el método de autenticación por credenciales con correo electrónico y contraseña. Así también, se estableció una política de contraseñas como que los usuarios puedan ingresar únicamente con su correo institucional que provee la Escuela Politécnica Nacional. Por último, se utilizó el estándar JWT para proteger rutas privadas, se integró la librería *Swagger* para documentar la API, y se crearon los roles del sistema representados en la Tabla 3.

Tabla 3. Roles del sistema.

Nombre del rol en el sistema	Descripción
COORDINADOR	Refiere a los coordinadores de cualquiera de las carreras que oferta la facultad. Estos usuarios pueden ingresar datos de entrada y generar horarios académicos.
DOCENTE	Refiere a los docentes de la facultad que son tomados en cuenta en la creación de actividades

	del sistema. Este rol permite que un docente pueda ingresar su horario disponible.
SUBDECANO	Refiere a la autoridad que puede realizar las mismas operaciones que un coordinador y puede aprobar o no un horario generado durante la planificación académica.
GESTOR_ESPACIOS_FISICOS	Este rol permite que un usuario pueda registrar las aulas de las facultades y su capacidad.
JEFE_DE_DEPARTAMENTO	Este rol permite que un usuario pueda únicamente visualizar los horarios académicos de un semestre.

Al finalizar este *sprint* se obtuvo las especificaciones ejecutables del *feature* A1, los roles del sistema y los métodos de autenticación, y autorización de la API. No obstante, no se completó la implementación del *feature* A1 en el componente SPA_Frontend, por lo cual se omitió una demostración funcional del aplicativo debido al tiempo. Los incovenientes presentados durante el *sprint* fueron relacionados con el tiempo de aprendizaje del equipo sobre el manejo de las tecnologías y la carga académica que cada integrante no consideró en su planificación inicial.

Sprint 2

Este *sprint* tuvo como propósito completar las tareas pendientes del anterior *sprint* como la implementación del *feature* A1 en el componente SPA_Frontend. En este punto del proyecto los miembros del equipo desarrollaron las primeras vistas de la aplicación web en Angular. Este framework dispone de un sistema modular que permitió al equipo trabajar de forma independiente sin afectar el progreso del resto.

La interfaz desarrollada del módulo de parámetros iniciales cuenta con dos submódulos: jornada laboral y aulas disponibles. El módulo de jornada laboral que se muestra en la Figura 13 dispone de dos secciones. La primera que con resaltado amarillo que permite seleccionar la hora de inicio, almuerzo y fin de la jornada laboral. En la sección con resaltado verde es posible seleccionar los días laborables de lunes a domingo en los cuales se impartirán clases.

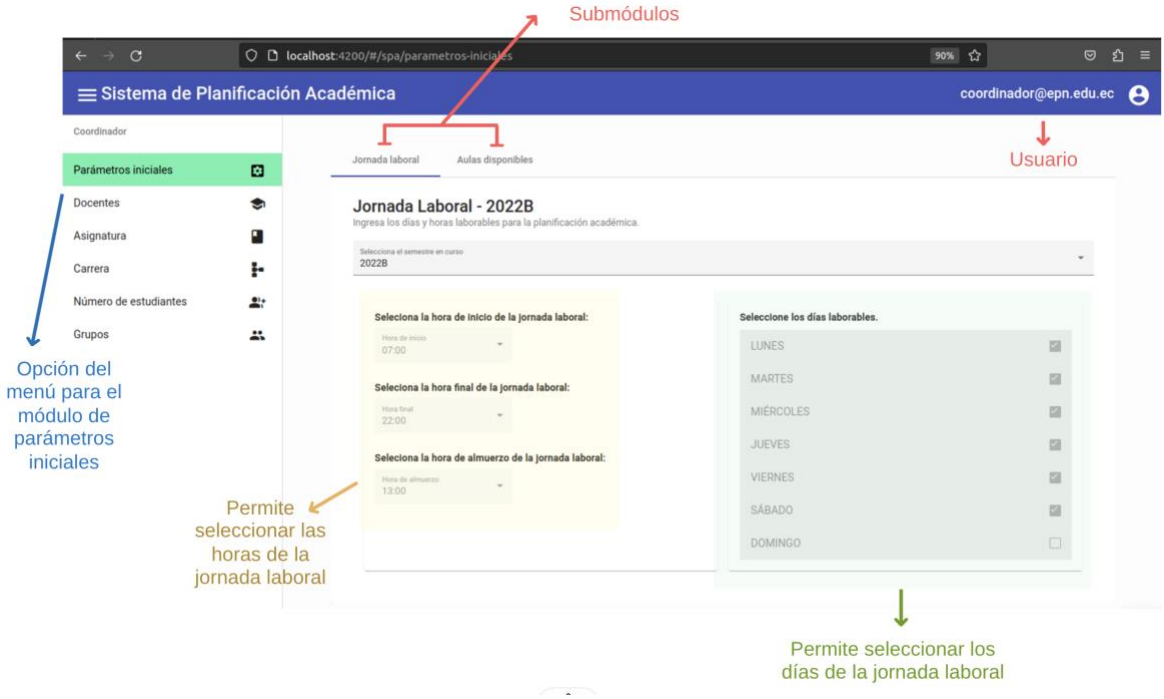


Figura 13. Interfaz del módulo del submódulo de jornada laboral.

El siguiente submódulo permite ingreso de tipos de aulas por facultad o unidad académica como se muestra en la Figura 14. Para ingresar una nueva aula es necesario seleccionar una facultad que debe estar creada previamente en el sistema.

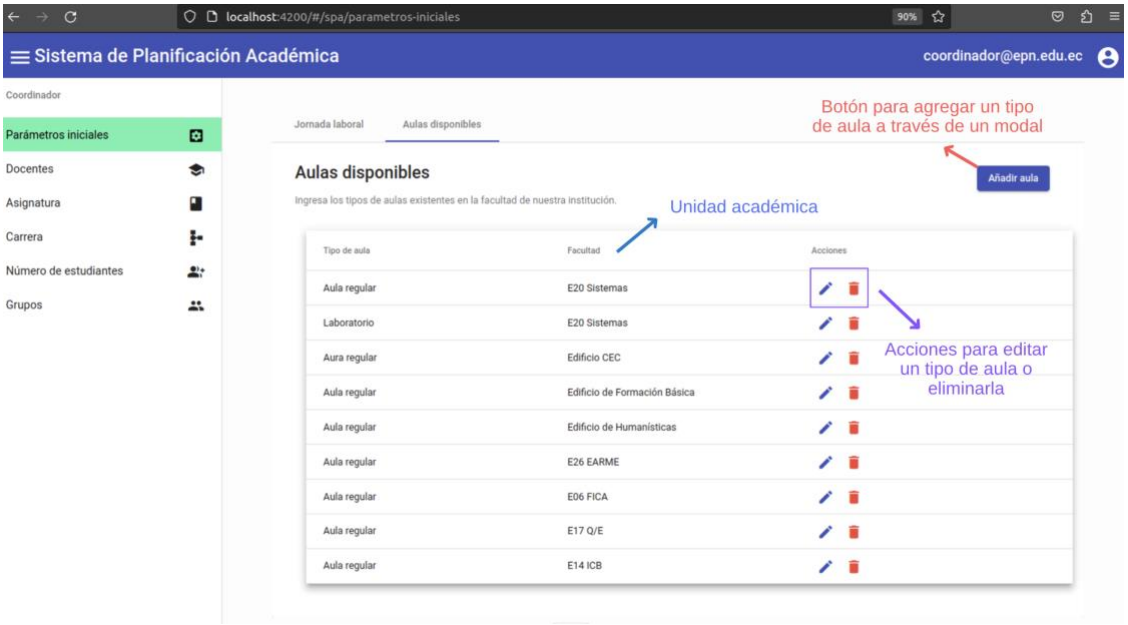


Figura 14. Interfaz del módulo del submódulo de aulas disponibles.

Durante la implementación de las interfaces de la Figura 13 y 14 se crearon diferentes rutas de la API y estas fueron documentadas utilizando la librería Swagger. En la Figura

15 se puede visualizar el nombre de las rutas, los métodos HTTP, una descripción y otra información relevante. Una ventaja de utilizar Swagger en el proyecto fue que era posible probar los métodos antes de utilizarlos directamente en el sitio del documento.

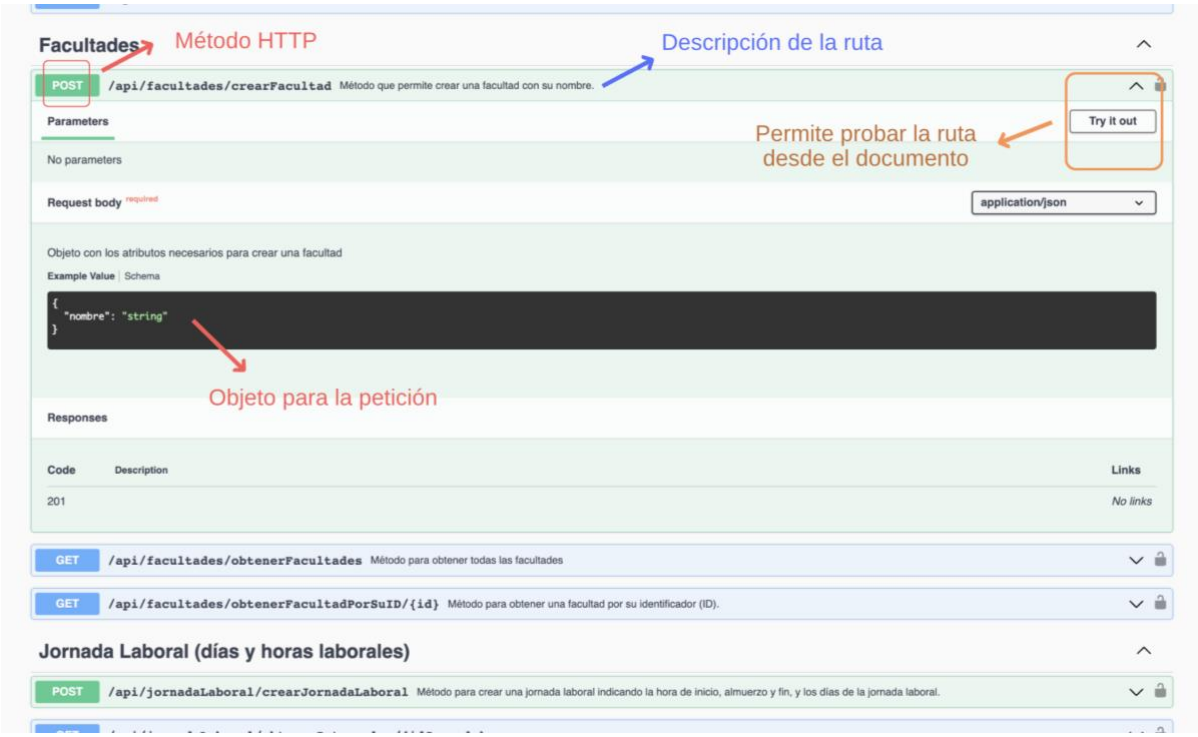


Figura 15. Rutas documentadas en Swagger.

Al concluir con el *sprint* se logró conseguir el objetivo y se tuvo una versión del sistema donde el usuario podía navegar por los módulos iniciales de cada componente. Por otro lado, el equipo concluyó la importancia de instalar el CLI del software FET para poder entender a mejor manera su funcionamiento en el servidor. Esto debido a que en la etapa inicial de la investigación no existía una documentación clara de cómo utilizar el CLI del software FET en un servidor.

Sprint 3

En este *sprint* se planteó iniciar la integración con el CLI del software FET y el sistema generando la documentación respectiva para lograr cumplir con el *feature* A3. Como se mencionó anteriormente un problema durante la realización del proyecto fue la compleja y reducida documentación del uso del software FET en distribuciones Linux. Por lo cual, el hecho de documentar la instalación del CLI del software FET y entender su funcionamiento desde un entorno Linux, representa un recurso importante de documentación a futuro para entender y replicar este proyecto.

De esta forma se creó una página web dentro del sitio del proyecto en Notion que tenga instrucciones entendibles sobre la instalación del CLI del software FET (ANEXO VI). Como se ilustra en la Figura 16, a partir de este recurso es posible instalar las dependencias necesarias del CLI y generar los binarios luego de compilar el proyecto. En este manual de instalación se considera el uso del software Qmake y Make para realizar una compilación automatizada ya que el componente FET-CL necesita de varios paquetes nativos del sistema operativo del servidor.

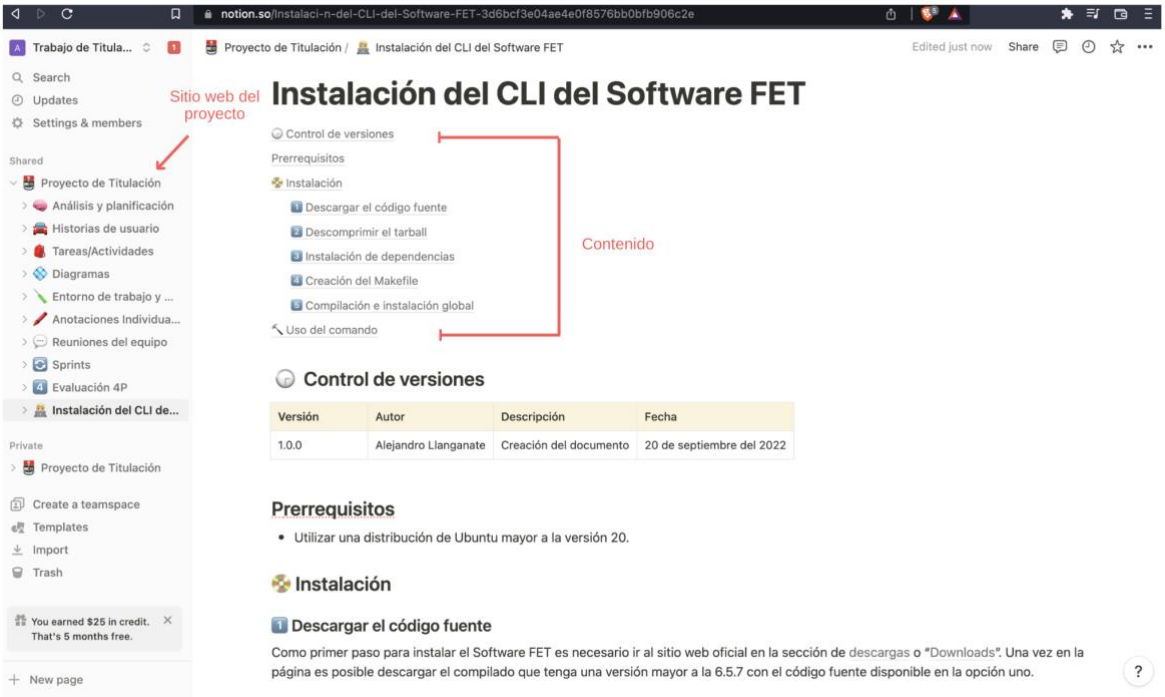


Figura 16. Documentación de la instalación del CLI del software FET.

La instalación del CLI del software FET en el servidor permite reutilizar su motor a través del uso del comando "fet-cl". Posteriormente a la instalación fue posible identificar el funcionamiento de esta herramienta y cómo podría ser integrada con el aplicativo web. Al ejecutar el comando del CLI del software FET se establece el proceso representado en la Figura 17 que finaliza con el horario generado en formato XML dentro del directorio *timetables*. Este horario se encuentra en un archivo cuyo nombre finaliza con "subgroups" y gracias a su formato es posible obtener dichos datos y transformarlos a JSON con el fin que puedan ser utilizados usando los componentes de Angular Material.

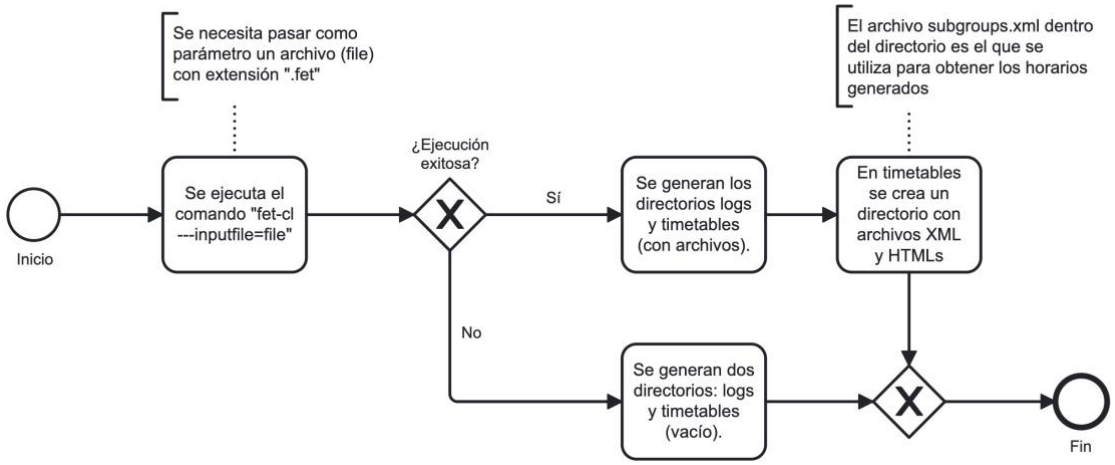


Figura 17. Diagrama del proceso para la generación de horarios con el FET-CL.

Por otra parte, en el transcurso de este *sprint* el Consejo Politécnico de la Escuela Politécnica Nacional posicionó a la MSc. María Monserrate Intriago Pazmiño como nueva subdecana de la Facultad de Ingeniería de Sistemas. Al finalizar el *sprint* se tuvo una reunión con el equipo, la nueva subdecana y el coordinador de la carrera de Ingeniería de Software. En esta reunión (ANEXO VII) se presentó el sistema con los módulos iniciales funcionales y las autoridades recalcaron que era fundamental conseguir la integración del FET en el sistema web.

Por lo tanto, los avances presentados no fueron suficientes para cumplir con los objetivos propuestos inicialmente en este componente. Debido a lo cual, el autor del presente trabajo de integración curricular decidió seguir con el proyecto otro semestre académico para cumplir con la integración del FET en la aplicación. Este *feature* fue el que se identificó como el que más valor aportaba en el proyecto. Además, en este *sprint* ya se había documentado e identificado la forma de reutilizar el motor del software FET a través del directorio *timetable*.

Por último, en la reunión con las autoridades se designó al coordinador de carrera de Ingeniería de Software como nuevo Product Owner. No obstante, para el periodo académico “2022-B” la subdecana de la facultad tuvo una mayor participación en el proyecto convirtiéndose en la Product Owner del mismo.

Sprint 4

En el anterior *sprint* se había identificado que para la generación de horarios con el CLI del software FET era fundamental contar con el registro de actividades en el formato del archivo con extensión `.fet`. En virtud de ello, este *sprint* tuvo el objetivo de implementar

el *feature* A2 tanto en el componente SPA_Backend y SPA_Frontend. Cabe recalcar que el módulo de actividades abarca la mayoría de información de los módulos desarrollados por los integrantes del equipo.

En la ejecución del *sprint* se implementó lo necesario para llevar a cabo la definición de *steps* del escenario propuesto para este *feature*. Por lo tanto, se utilizó los ejemplos que se visualizan en la Figura 18 para el esquema de escenario relacionado a registrar una actividad con una duración correcta y errónea.

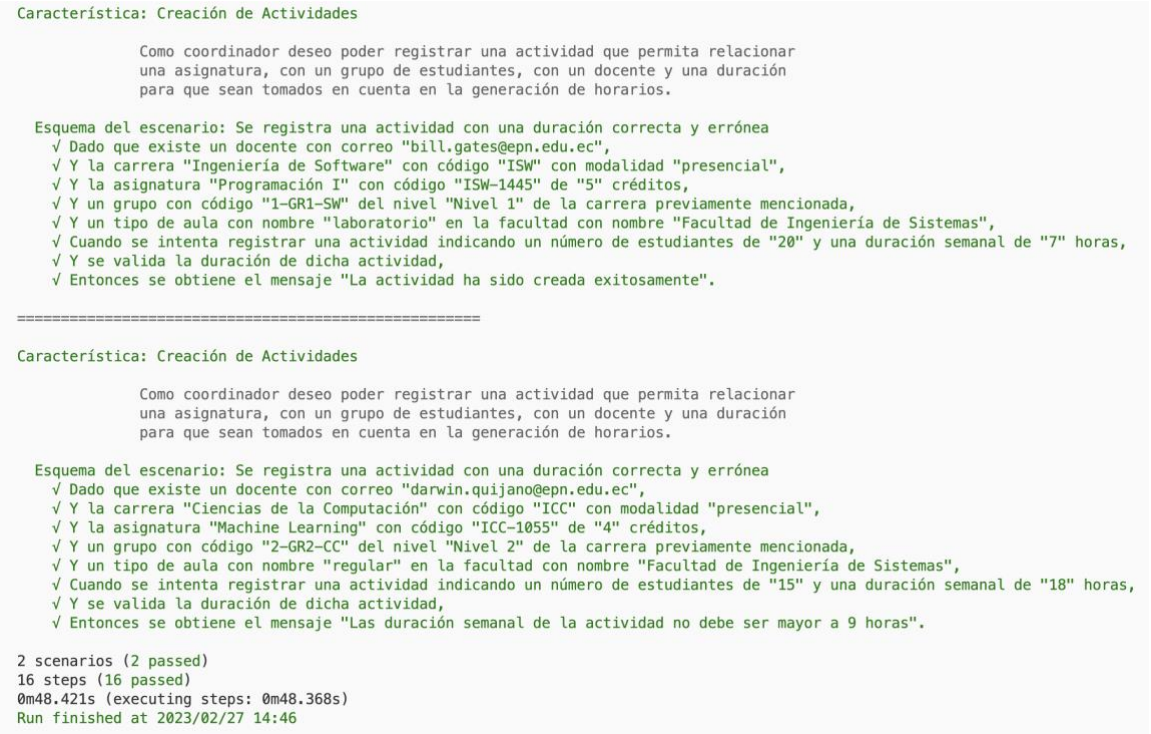


Figura 18. Ejecución de la prueba del feature A2.

Posteriormente se implementaron dos módulos necesarios con respecto a este *feature*. El primero es el módulo de grupos (Figura 19) que permite generar los niveles correspondientes por cada carrera. Actualmente en la Escuela Politécnica Nacional existen nueve niveles por cada carrera que oferta la Facultad de Ingeniería de Sistemas. Así también, se pueden crear grupos según los niveles de cada carrera con los cuales se les puede definir un número de estudiantes por grupo.

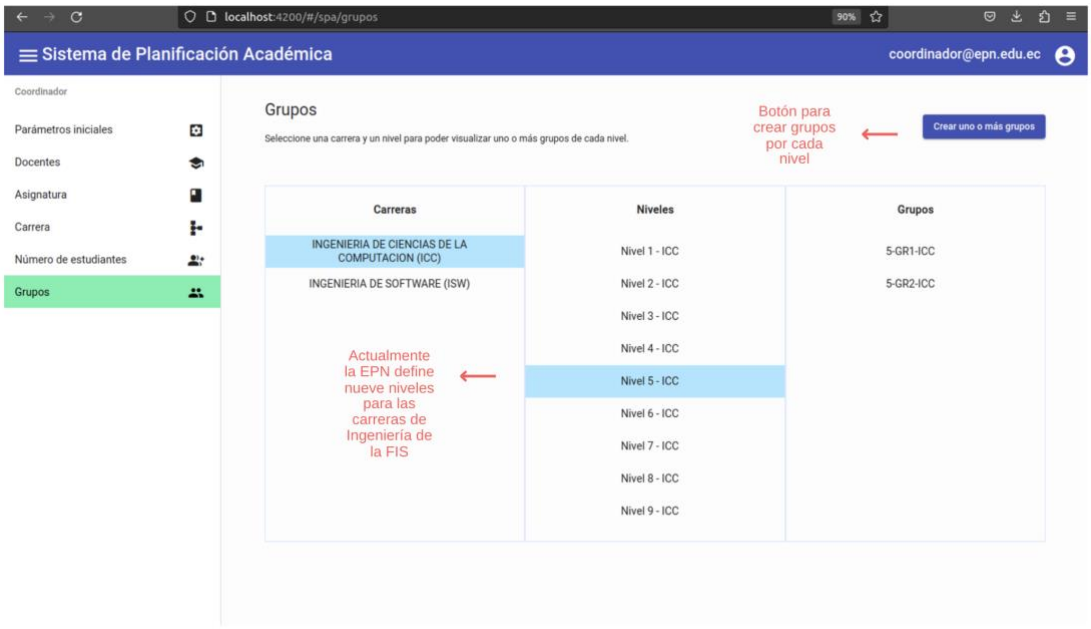


Figura 19. Interfaz del módulo de grupos del aplicativo.

El segundo módulo implementado es el de actividades que permite visualizar las actividades registradas en el sistema como añadir nuevas. Como se visualiza en la Figura 20, al registrar una nueva actividad se debe seleccionar un docente, una asignatura, un tipo de aula y un grupo. En el modal implementado cada una de estas opciones del usuario son resumidas en una sección en la cual el usuario finalmente debe ingresar una duración por dicha actividad.

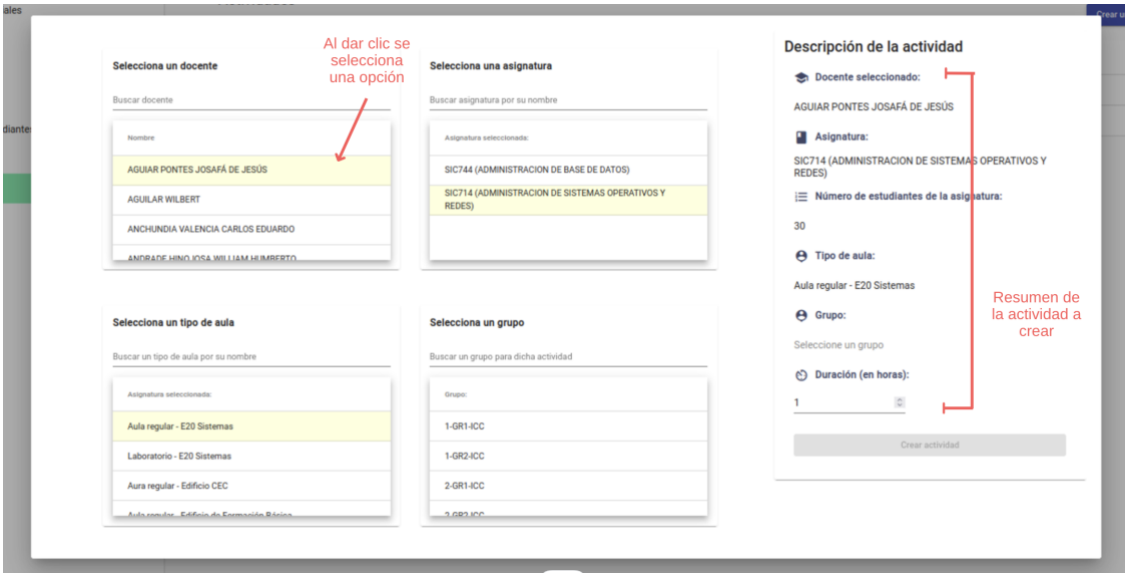


Figura 20. Modal de registro de una actividad.

Al concluir el *sprint* se completó exitosamente el objetivo propuesto y se socializó estos avances con el Product Owner del proyecto. En esta reunión, se identificó que el proyecto no contempló un miembro del equipo que pueda abarcar restricciones personalizadas donde se fija una actividad a un horario específico (hora, día y aula). Esta funcionalidad es la más utilizada actualmente por los encargados de la facultad en la generación de horarios con el software FET. No obstante, el alcance del presente componente no cuenta con la realización de este módulo por lo cual se planteó como una mejora a futuro para un próximo trabajo de integración curricular.

Sprint 5

La finalidad de este *sprint* fue abarcar el *feature* de generación de horarios utilizando el motor del software FET para los componentes SPA_Backend y SPA_Frontend. Para lograr este objetivo se empleó el conocimiento obtenido en el *sprint* 3 sobre el funcionamiento del FET CL. Así mismo el Product Owner entregó el archivo de la última planificación semestral de la facultad “HorariosFIS_2022B_v45_horario.fet” (ver ANEXO VIII) con datos reales que apoyan la implementación inicial del *feature*. Otro objetivo de este *sprint* fue ingresar en el sistema datos reales del archivo de la planificación académica de la facultad en el periodo “2022-B”.

La solución propuesta, con respecto a la integración del motor del FET, consistió en implementar los métodos respectivos en el servicio de horarios donde se pueda:

1. Recuperar los datos de asignaturas, tipos de aulas, docentes, niveles, grupos, actividades, facultades y espacios físicos.
2. Transformar los datos en etiquetas estructuradas XML, como se visualiza en la Figura 21, utilizando la librería XMLBuilder.
3. Utilizar la función *exec* de la librería “node:child_process” para poder ejecutar el comando “fet-cl” desde el servidor.
4. Utilizar la función xml2js de la librería “xml-js” para convertir el formato de etiquetas XML del horario a un formato JSON.

Figura 21. Fragmento de código donde se construye el archivo `.fet`.

De esta forma se logró la implementación de la integración del motor del software FET con el componente "SPA_Backend" del proyecto. Además, se validó este *feature* utilizando el archivo `.fet` de la facultad y utilizándolo para la definición de *steps*. La ejecución de la prueba relacionada al *feature* A3 se completó correctamente tal como se ilustra en la Figura 22.

```
=================================================

Característica: Generación de horarios con el software FET

  El sistema necesita enviar los datos de entrada al software FET en un
  formato de etiquetas con extensión .fet para obtener como salida los
  horarios de cada nivel académico.

  Escenario: Se genera los horarios con el motor del software FET
    √ Dado el archivo de planificación "HorariosFIS_2022B_v45_horarios.fet",
    √ Cuando se reutilice el motor del software FET desde la API,
    √ Entonces se obtendrán los horarios por subgrupo generados en etiquetas XML.

1 scenario (1 passed)
3 steps (3 passed)
0m07.134s (executing steps: 0m07.094s)
Run finished at 2023/02/27 20:53
```

Figura 22. Ejecución de la prueba del *feature* A3.

Con respecto al componente "SPA_Frontend", se añadió un botón a la interfaz del módulo de horarios como se visualiza en la Figura 23. El usuario al dar clic en dicho botón genera un registro del horario en la base de datos utilizando los datos presentes

del sistema. Este registro se almacena en formato JSON para que posteriormente pueda ser visualizado

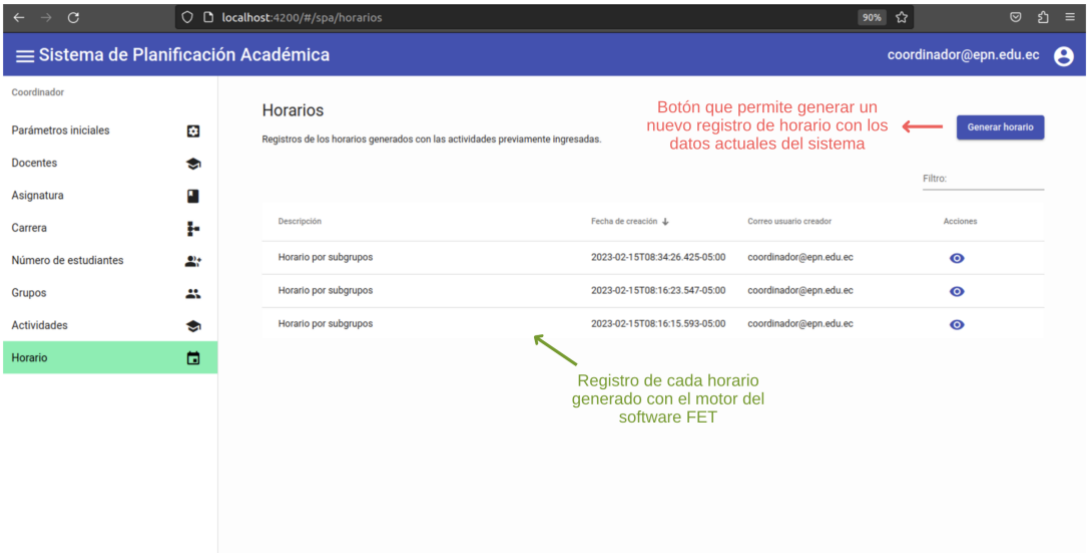


Figura 23. Interfaz del módulo de horarios con botón para generar un horario.

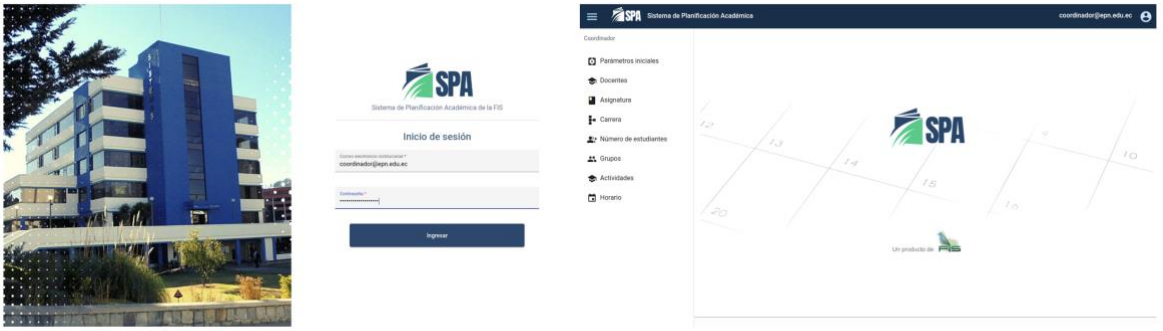
Para el ingreso de datos reales en el sistema se creó un script en JavaScript (ANEXO IX) para generar dos archivos con extensión “.csv” con la información de docentes y asignaturas. En los módulos de las interfaces de docentes y asignaturas fue posible cargar dichos archivos y generar, en el caso de las asignaturas, cerca de 141 registros como se visualiza en la Figura 24. Por otra parte, el resto de datos de entrada tuvieron que ser ingresados manualmente en el sistema.

Figura 24. Datos reales ingresados en el módulo de asignaturas.

Al concluir este *sprint* se cumplieron los objetivos planteados y se logró finalmente la integración del motor del software FET en el sistema. No obstante, en la reunión con la Product Owner se pudo observar que existían problemas en la visualización de horarios por lo cual era necesario analizar el componente de visualización de horarios desarrollado por otro miembro del equipo. Esto permitió plantear el último *sprint* del proyecto con la visualización correcta de los horarios generados para poder utilizar los filtros desarrollados y descargarlos en un formato accesible para los involucrados en el proceso de planificación académica.

Sprint 6

En este *sprint* se planteó abarcar el *feature* A4 y mejorar el *feature* de visualización de horarios del componente desarrollado previamente por Diana López (ANEXO X). Así también, el Product Owner solicitó cargar horarios de anteriores planificaciones para ser visualizados en el componente "SPA_Frontend". Por último, se propuso mejorar el aspecto de las interfaces del sistema en general para que el proyecto sea concebido como un producto de la facultad como se ilustra en la Figura 25.



a) Inicio de sesión b) Home

Figura 25. Nuevo aspecto visual del sistema.

Para el requerimiento referente a mostrar horarios de planificaciones anteriores se propuso que el usuario pueda cargar un archivo `.fet` de una planificación previa desde el módulo de horarios. Para ello, se creó un botón que permita subir un archivo desde el navegador para que posteriormente una ruta de la API pueda procesar este archivo utilizando el motor del software FET y finalmente se almacene el horario en el formato del sistema. Esta funcionalidad se ve representada en la Figura 26 en donde la única precondición para cargar un archivo es que este cumpla con el formato `.fet` establecido.

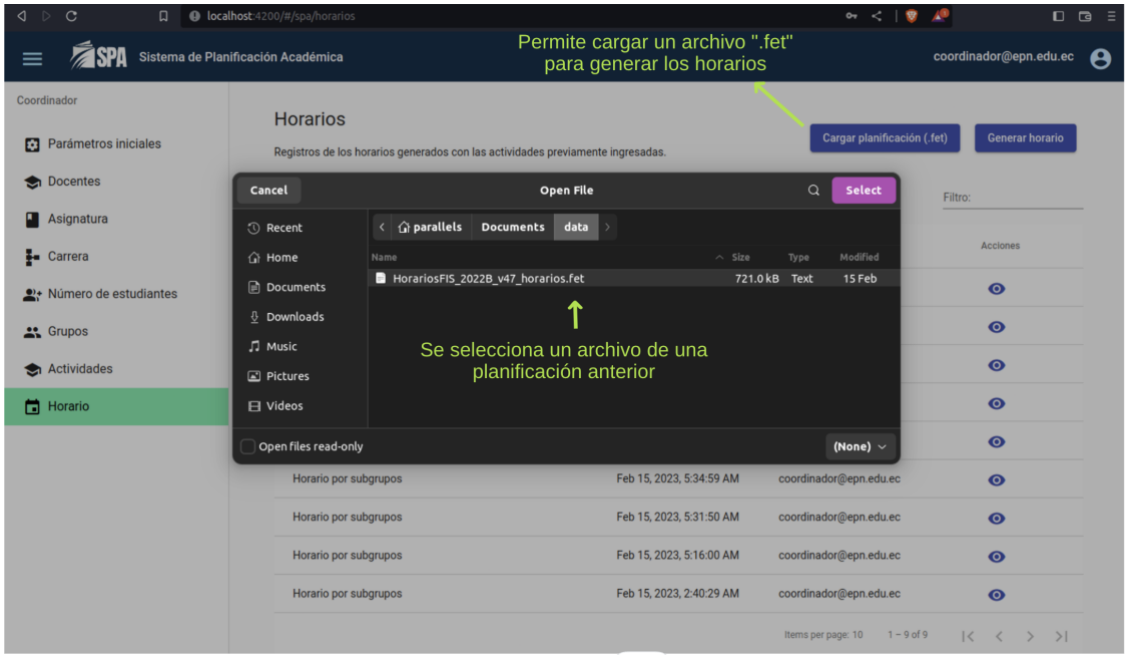


Figura 26. Carga de una planificación anterior en el sistema.

Por otra parte, el problema con respecto a la visualización de horarios consistió en que el usuario al aplicar cualquiera de los filtros no dispone de una visibilidad clara del contenido del horario. Como se muestra en la Figura 27 cada uno de los valores de las celdas son ilegibles ya que presentan un tamaño de letra muy reducido.

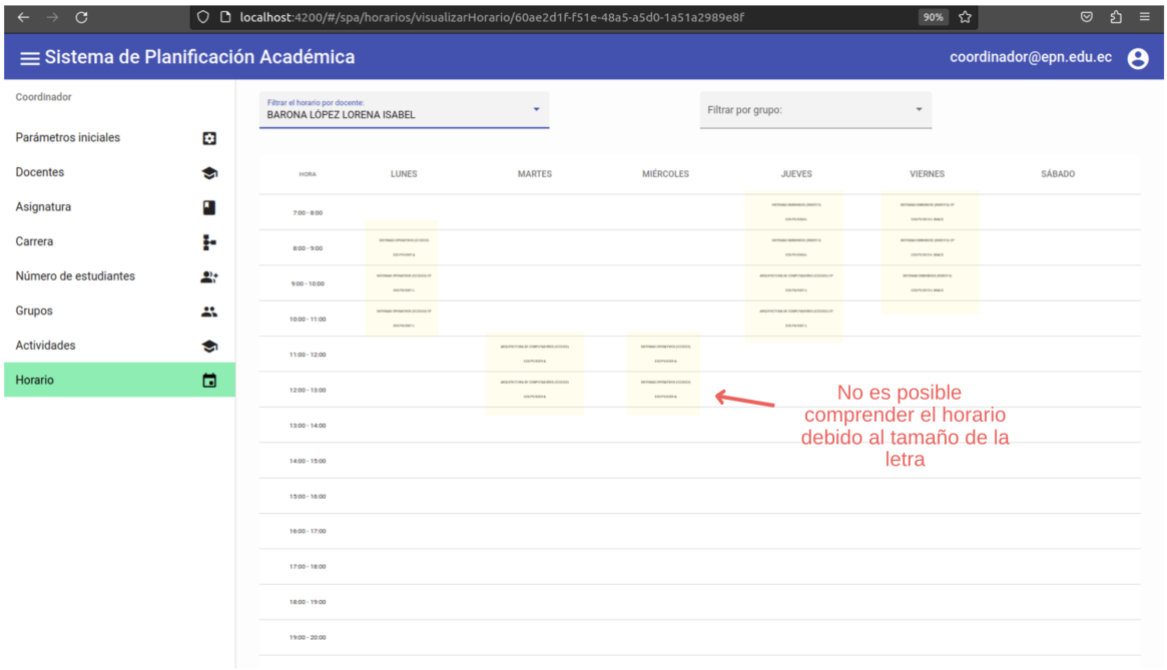


Figura 27. Problema presentado en la visualización de horarios.

Como solución a lo anteriormente mencionado, se adaptó el componente visual del horario (Figura 28) a uno que tenga barras de desplazamiento y un tamaño de letra legible. Este cambio favoreció a la comprensión del contenido de los horarios que son mostrados según el filtro por docente o grupo que se aplique.

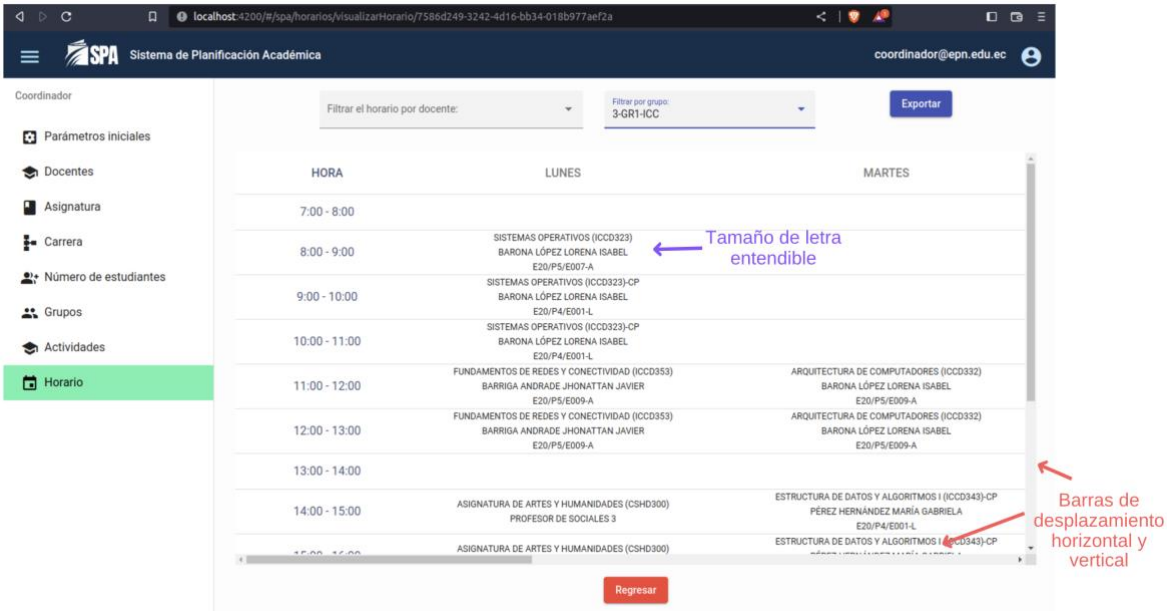


Figura 28. Mejoras en la visualización de horarios.

Para concluir, se implementó la descarga de horarios como un mecanismo que permita difundir los horarios a diferentes involucrados en el proceso de planificación. Tal como se ilustra en la Figura 29 la interfaz de visualización de horario dispone de un botón que permite exportar desde el navegador un archivo excel (ANEXO XI) cuyo formato debe ser mejorado en futuras versiones.

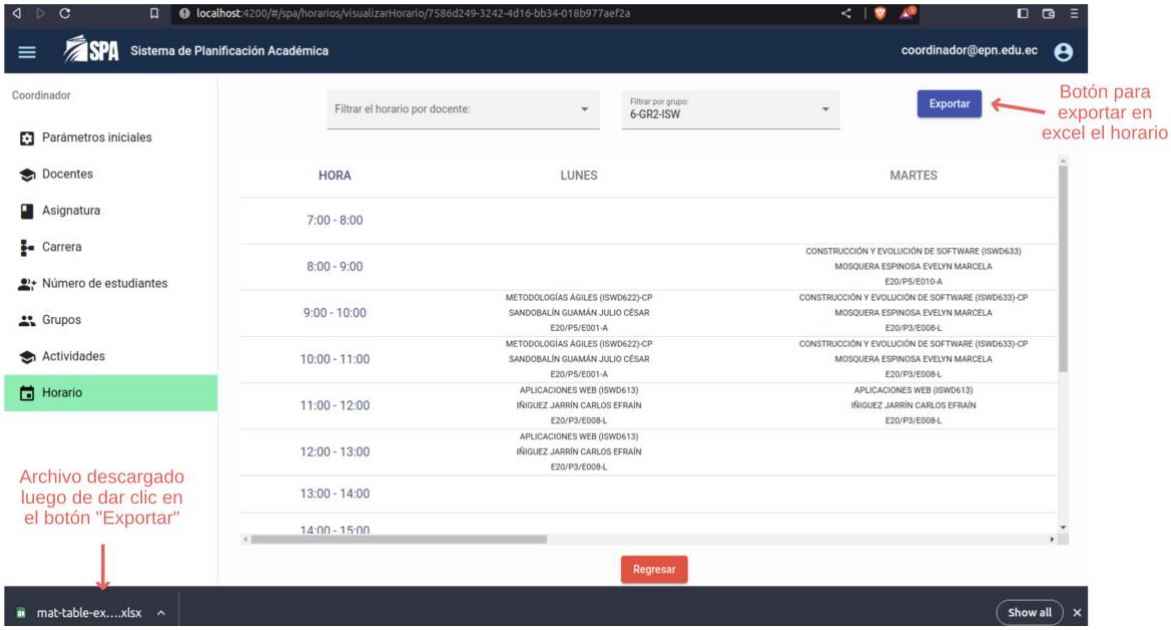


Figura 29. Implementación del *feature* A4.

El *sprint* finalizó exitosamente con una revisión final de los objetivos alcanzados. Por lo cual, se realizó una reunión presencial con la Product Owner, es decir la subdecana de la facultad, y el docente Adrián Eguez como invitado. En está reunión la Product Owner aprobó los *features* del sistema implementados a lo largo del *sprint* y se identicaron futuras mejoras. Entre estas, cambiar el modal de la creación de una actividad por una vista completa, y mejorar el diseño de cada una de las interfaces para que puedan ser visualizadas correctamente desde un dispositivo móvil.

4 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

4.1 Resultados

El equipo de desarrollo al final del proyecto acumuló un conocimiento sobre las necesidades del usuario que permitieron proponer soluciones factibles alineados a los objetivos de la planificación académica de la facultad. Sin embargo, al iniciar el proyecto existió una gran incertidumbre sobre cómo encontrar el mejor camino para la generación de horarios con la reutilización del motor del software FET. Esta incertidumbre fue gestionada correctamente aplicando el desarrollo dirigido por comportamiento que permitió entender los objetivos de negocio subyacentes del proyecto y definir *features* que progresivamente brindaron una solución efectiva con especificaciones ejecutables que validaban dichos objetivos. Por lo tanto, el progreso del entendimiento de la mejor solución planteada del proyecto fue similar al del diagrama mostrado en la Figura 30.

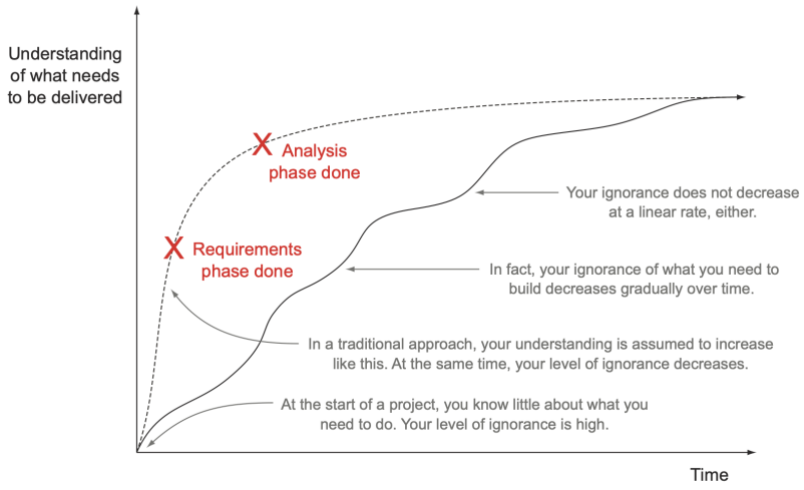


Figura 30. Progreso del entendimiento de un proyecto real. Tomado de [10].

Por otro lado, al finalizar el proyecto se cumplieron todos los *features* establecidos en el transcurso de siete *sprints*. Como se visualiza en la Figura 31, inicialmente se había planificado finalizar los cuatro *features* del componente en el transcurso de cuatro *sprints* cuando en la realidad se los completó en siete *sprints*.

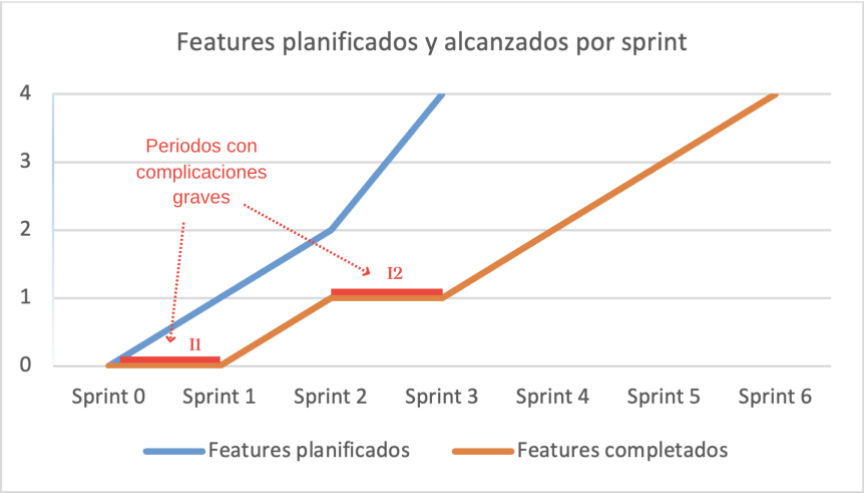


Figura 31. Diagrama de *features* planificados y alcanzados por *sprint*.

Aquello significó que el proyecto tomó más tiempo de lo planificado inicialmente debido a las complicaciones presentadas en los intervalos:

- **I1:** La excesiva carga académica de los miembros del equipo, el tiempo de aprendizaje de las herramientas y tecnologías utilizadas que tomó más de lo esperado.

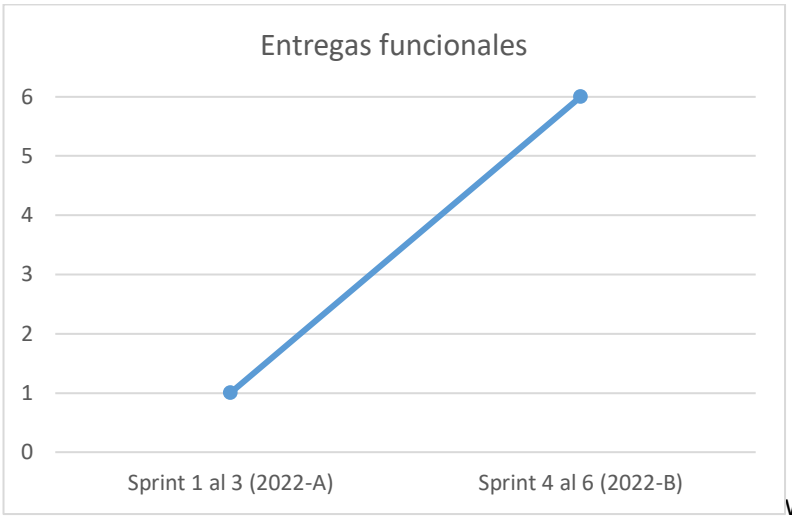
  Medidas correctivas: Realizar una nueva planificación considerando la carga académica de los miembros del equipo. Así también, compartir en conjunto el conocimiento adquirido por cada integrante para solventar obstáculos durante el proceso de aprendizaje.

- **I2:** El cambio de autoridades de decano y subdecano de la Facultad de Ingeniería de Sistemas, los sucesos de manifestaciones sociales que descoordinaron las actividades del equipo y otros problemas de comunicación sucitados.

  Medidas correctivas: Fomentar la participación de la nueva subdecana en el proyecto a través de una reunión demostrativa de la aplicación.

Además, la cantidad de *features* completados en el periodo 2022-B (durante el *sprint* 4 al *sprint* 6) fue tres veces mayor que las completadas en el periodo 2022-A (del *sprint* 1 al sprint 3) como se aprecia en la Figura 32. El hecho de haber establecido en ese intervalo de *sprints* alrededor de seis reuniones con la Product Owner, permitió poder cumplir cada uno de los *features* satisfactoriamente ya que se presentaron en cada reunión una entrega funcional del sistema. Al contrario del los *sprints* iniciales del periodo 2022-A

donde la participación de la Product Owner se limitó a la revisión de una sola entrega funcional del proyecto.



w

Figura 32. Diagrama de entregas funcionales por intervalos de *sprints*.

4.3 Conclusiones

El Sistema de Planificación Académica implementado es una aplicación web que reutiliza el motor del software FET mediante la integración de la línea de comandos del FET-CL con un componente denominado “SPA_Backend” de la arquitectura del proyecto. Además, esta arquitectura cuenta con una base de datos en la nube para almacenar y acceder a la información de todos los módulos del sistema y así evitar reprocesos innecesarios o pérdidas de información.

Por otra parte, el sistema implementado cuenta con un componente llamado “SPA_Frontend” el cual es la parte de la aplicación que interactúa con los usuarios y a su vez consume los métodos de una API. La cual provee las rutas necesarias para poder realizar diferentes operaciones como el generar y almacenar horarios en la base de datos. Así también, el sistema al tratarse de una aplicación web que implementa un mecanismo de autenticación y autorización de usuarios. Esta permite el acceso concurrente de diferentes usuarios con un rol específico. Dichos usuarios pueden acceder al sistema desde un navegador web para continuar con sus actividades en el proceso de la planificación académica simultáneamente.

De igual forma, la aplicación web cuenta con un módulo en el cual es posible visualizar los horarios generados según un docente o un grupo. Y, estos horarios pueden ser exportados en formato Excel para que puedan ser difundidos a través de cualquier otra plataforma digital. Si bien para la implementación de las interfaces de cada módulo se utilizaron componentes de Angular Material que tienen características de usabilidad y accesibilidad, no se ha realizado ninguna evaluación adicional sobre qué tan usable o accesible es el sistema.

Finalmente, la implementación del proyecto fue acompañado por buenas prácticas de desarrollo como el desarrollo dirigido por el comportamiento que permitió crear especificaciones ejecutables para automatizar los criterios de aceptación del sistema. Por lo tanto, la implementación no se concentró únicamente en construir un sistema correctamente sino en construir el sistema correcto que permita alcanzar los objetivos de la planificación académica.

4.5 Recomendaciones

Un problema relevante en la construcción del componente de generación y difusión de horarios fue disponer de una documentación clara y actualizada de la instalación y funcionamiento del FET-CL. De modo que se recomienda en documentar los pasos del proyecto en los cuales intervenga el CLI del software FET. Esto debido a que cuando se desee levantar el proyecto en una máquina distinta se deberá replicar los pasos seguidos y es importante contar con dicha información.

Para replicar este componente se recomienda tener conocimientos en las tecnologías empleadas y en desarrollo dirigido por el comportamiento (BDD), o al menos haber tenido una experiencia previa trabajando con el desarrollo dirigido por pruebas (TDD). La mayoría de los integrantes del equipo no tenían conocimientos previos en las tecnologías usadas o en BDD y el progreso de aprendizaje tomó más de lo esperado lo que implicó retrasos en el proyecto.

Por otra parte, pese a utilizar una metodología ágil en los *sprints* iniciales el equipo no incluyó la participación del Product Owner de forma activa lo que no permitió la entrega de incrementos funcionales del sistema. Esto se solucionó en los *sprints* posteriormente y los resultados fueron positivos por lo que se sugiere que para este proyecto la participación del Product Owner sea totalmente activa.

Finalmente, se recomienda seguir con el proyecto en las mejoras propuestas en el último *sprint* las cuales son: cambiar el modal de la creación de una actividad por una vista completa, y mejorar el diseño de cada una de las interfaces para que puedan ser visualizadas correctamente desde un dispositivo móvil.

5 REFERENCIAS

[1] Escuela Politécnica Nacional, «Comisión de Gestión de Calidad y Evaluación Interna,» 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,» 5 Febrero 2021. [En línea]. Available: https://bibdigital.epn.edu.ec/handle/15000/21432.

[3] J. J. Aguiar Pontes, J. C. Sandobalin Guamán y J. F. Lucio Naranjo, Interviewees, *Entrevista de requerimientos del Sistema de Planificación Académica para la Facultad de Ingeniería en Sistemas.* [Entrevista]. 2021.

[4] J. F. Lucio Naranjo, Interviewee, *Entrevista para aclarar dudas generales sobre los requerimientos del Sistema de Planificación Académica para la Facultad de Ingeniería en Sistemas.* [Entrevista]. 2022.

[5] Manitoba Education, Citizenship and Youth, Scheduling kindergarten to grade 8 physical education/health education : a resource for school administrators., Manitoba, Winnipeg: Minister of Education, 2007.

[6] K. Vinod y S. Yadav, «Academic timetable scheduling: revisited,» *International Journal of Research In Science & Engineering,* pp. 417-423, Abril 2016.

[7] A. Fiechter, «University timetable scheduling,» 2018. [En línea]. Available: https://www.inf.usi.ch/faculty/lanza/Downloads/BSc/Fiec2018a.pdf.

[8] L. Lalescu, «FET Description,» 2023. [En línea]. Available: https://lalescu.ro/liviu/fet/. [Último acceso: 18 Abril 2022].

[9] L. Lalescu y V. Dirr, «FET Manual,» 17 Agosto 2018. [En línea]. Available: https://www.timetabling.de/manual/FET-manual.en.html#id_11. [Último acceso: 24 Julio 2022].

[10] J. Ferguson Smart, BDD in Action: Behavior-driven development for the whole software lifecycle, 1st Edición ed., Shelter Island, New York: Manning Publications Co, 2015, p. 385.

[11] SMARTBEAR, «What is Cucumber?,» [En línea]. Available: https://cucumber.io/docs/guides/overview/. [Último acceso: Julio 2022].

[12] SMARTBEAR, «Step Definitions,» [En línea]. Available: https://cucumber.io/docs/cucumber/step-definitions/?lang=java. [Último acceso: Junio 2022].

6 ANEXOS

ANEXO I. ENLACE A LA ENTREVISTA DE REQUERIMIENTOS DEL SISTEMA DE PLANIFICACIÓN ACADÉMICA PARA LA FACULTAD DE INGENIERÍA EN SISTEMAS

Disponible en: https://www.youtube.com/watch?v=c6QVJtO-f4A&feature=youtu.be

ANEXO II. ENLACE A LA ENTREVISTA PARA ACLARAR DUDAS GENERALES CON EL SUBDECANO DE LA FIS

Disponible en: https://www.youtube.com/watch?v=8Y60S9U4XAg&feature=youtu.be

ANEXO III. ENLACE A LA ENTREVISTA SOBRE LA TESIS DEL ESTUDIANTE ALEX ULLOA

Disponible en: https://www.youtube.com/watch?v=r8gOMTsNGzQ

ANEXO IV. ENLACES A LOS REPOSITORIOS DE LOS COMPONENTES SPA_FRONTEND Y SPA_BACKEND

Código fuente del componente SPA_Frontend disponible en:

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

Código fuente del componente SPA_Backend disponible en:

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

ANEXO V. ENLACE AL SITIO WEB DEL PROYECTO

Es necesario que cuente con una cuenta en Notion para ingresar al sitio de documentación del proyecto.

Disponible en: https://www.notion.so/Proyecto-de-Titulaci-n-69e313d3b42347cbba104e0bdfa3cb29?pvs=4

ANEXO VI. ENLACE A LA DOCUMENTACIÓN DE LA INSTALACIÓN DEL FET-CL

Es necesario que cuente con una cuenta en Notion para ingresar a este sitio.

Disponible en: https://festive-salary-67b.notion.site/Instalaci-n-del-CLI-del-Software-FET-3d6bcf3e04ae4e0f8576bb0bfb906c2e

ANEXO VII. PRESENTACIÓN DEL SISTEMA A LA NUEVA SUBDECANA DE LA FIS

Para ingresar a este recurso se debe solicitar acceso a la subdecana Monserrate Intriago.

Disponible dando clic aquí.

ANEXO VIII. ARCHIVO ".FET" DE LA PLANIFICACIÓN ACADÉMICA DEL 2022-B

Disponible en: https://drive.google.com/file/d/1RqnDAKImhqTFFMuC0e8hybaWqkrvG6U2/view?usp=share_link

ANEXO IX. SCRIPT EN JAVASCRIPT UTILIZADO PARA OBTENER DOCENTES Y ASIGNATURAS DE UN ARCHIVO “.FET”

```
import * as fs from 'fs';
import * as xml2js from 'xml2js';

const parser = new xml2js.Parser();

// Variables para almacenar la información
const docentes = [];
const asignaturas = [];
const niveles = [];
let resultsDocentes = [];
let resultsAsignaturas = [];

fs.readFile('./Horarios_2022b.fet', function (err, data) {
  parser.parseString(data, function (err, result) {
    console.dir(result);
    docentes.push(result['fet']['Teachers_List']);
    asignaturas.push(result['fet']['Subjects_List']);
    niveles.push(result['fet']['Students_List']);
  });

  // registro de docentes
  docentes.map((docente) => {
    resultsDocentes = docente[0]['Teacher'].map(
      (e, index) => `${e['Name'][0]}, example${index}@epn.edu.ec`,
    );
  });
  asignaturas.map((asignatura) => {
    // registro de asignaturas
    resultsAsignaturas = asignatura[0]['Subject'].map((s) => {
      const sinFormato = s['Name'][0];
      let arrMateriaCodigoSinFormato;
      if (sinFormato[0] !== '*' && sinFormato[0] !== '-') {
        arrMateriaCodigoSinFormato = sinFormato.split(' (');
        arrMateriaCodigoSinFormato = arrMateriaCodigoSinFormato.map((x) => {
```

```
        return x.replace(')', '');
      });
      if (arrMateriaCodigoSinFormato.length === 1) {
        arrMateriaCodigoSinFormato.push(
          arrMateriaCodigoSinFormato[0]
            .split(' ')
            .map((e) => e[0].toUpperCase())
            .join(''),
        );
      }
    }
    if (typeof arrMateriaCodigoSinFormato === 'object') {
      return arrMateriaCodigoSinFormato;
    }
  });
});
const printDocentes = resultsDocentes.reverse().sort().join('\n');
const printAsignaturas = resultsAsignaturas
  .filter((e) => e)
  .sort()
  .join('\n');

// Se generan los archivos con datos para docentes y asingaturas
fs.writeFileSync('docentes.csv', printDocentes);
fs.writeFileSync('asignaturas.csv', printAsignaturas);
});
```

ANEXO X. ENLACE AL TRABAJO DE INTEGRACIÓN CURRICULAR DE LA ESTUDIANTE DIANA LÓPEZ

Disponible en: https://bibdigital.epn.edu.ec/bitstream/15000/23393/1/CD%2012813.pdf

ANEXO XI. ENLACE AL ARCHIVO EXCEL GENERADO AL EXPORTAR UN HORARIO

Disponible en:
https://docs.google.com/spreadsheets/d/1BEEHDW3VBnoUNAf75yhW0FSzDBue6RDB6ZpaJW40qgc/edit?usp=share_link