ABRIL DE 2022

Propuesta: Construyendo un nuevo planner para la Escuela

PROPUESTA POR

Consejería Académica
Open Source UC
Voluntarios

Índice

1 Introducción **3**

2 Motivación **3**

3 Resumen del proyecto **4**

4 Detalles del proyecto **4**

4.1 Requerimientos 4

4.2 Equipo y Metodología 6

4.3 Línea de Tiempo 7

4.4 Desafíos de implementación 8

4.5 Arquitectura y entorno 10

5 Conclusión **12**

6 Agradecimientos **12**

1. Introducción

Como estudiantes de la Escuela estamos agudamente conscientes de la complejidad que significa planificar nuestras carreras. Cada vez más, dependemos de plataformas como Planner para navegar la complejidad en nuestras mallas, habiéndose convertido en una herramienta fundamental para todo el estudiantado: desde novatos hasta estudiantes en vías de egreso. Esta es la razón por la que nos tomamos muy seriamente los diversos problemas y dificultades del actual Planner, y por lo mismo, estamos empeñados en resolverlos.

El Planner de la Escuela sufre no solo de una profunda deuda técnica, sino que hoy en día carece de la funcionalidad necesaria para soportar alrededor de 2.000 estudiantes pertenecientes a las nuevas generaciones, debido a la incapacidad de soportar las actualizaciones en sus mallas.

En vista de eso es que se propone la creación de un nuevo planner, creado por estudiantes y para estudiantes. Esta plataforma sería una adaptación de la plataforma de planner actualmente en uso en la Universidad de Princeton, TigerPath, cuyos desarrolladores, estudiantes de la propia universidad, nos otorgaron una licencia para utilizar su código fuente para la elaboración de nuestra propia plataforma.

Esta propuesta comienza reseñando la motivación detrás del proyecto, dando un resumen de alto nivel de este, y luego ahondando en los detalles, tanto técnicos como funcionales, de la plataforma.

2. Motivación

Actualmente, la Escuela cuenta con dos principales medios para que cada estudiante pueda planificar sus semestres en el transcurso de su período universitario: la vista de seguimiento curricular en SIDING y la aplicación web planner. Cabe destacar que esta última tiende a ser preferida por el esquema más amigable y visual que presenta para organizar cada semestre y el plan de estudios de cada estudiante, especialmente considerando la complejidad asociada al sistema de majors, minors y títulos de la Escuela.

Planner es una plataforma muy útil, pero lamentablemente la acumulación de deuda técnica en el proyecto ha hecho insostenible su uso para la Escuela. Esto dado que se ha tomado un costo altísimo en términos de mantenimiento sin haber logrado solucionar los problemas más comunes, sino que también se ha dificultado incorporar las funcionalidades necesarias para la planificación de las generaciones 2020 en adelante.

Se investigaron plataformas de sistemas de planificación de cursos existentes y se identificó TigerPath, una plataforma inicialmente desarrollada en el marco del curso de Programación Avanzada de Princeton, que luego se transformaría en el planner oficial de la universidad. Esta posee un sistema que permite a estudiantes planificar mallas con requisitos complejos, con una interfaz fácil de usar y algoritmos que permiten la validación de los requisitos de los cursos.

3. Resumen del proyecto

Un equipo de seis estudiantes de Ingeniería UC desarrollarán una aplicación web basada en la plataforma TigerPath de la Universidad de Princeton. Esta plataforma tendrá como objetivo reemplazar y extender la funcionalidad actual del planner de la Escuela.

Los estudiantes serán contratados de forma análoga a ayudantes, y trabajarán en un equipo de desarrollo ágil, siguiendo una metodología basada en Scrum que trabaje para acomodar ritmos flexibles en los ciclos de desarrollo ("sprints"). El equipo completo consistirá en un estudiante coordinador, cinco desarrolladores y un punto de enlace con la Escuela ("product owner").

Además de identificar los requerimientos funcionales y no funcionales de la plataforma, basándose en conversaciones con las distintas partes interesadas de la Escuela, se identifican varias oportunidades de mejoras sobre el planner actual, desde accesibilidad hasta usabilidad y apoyo al estudiante.

Finalmente, se propone una arquitectura de sistemas moderna que busca, por un lado, usar tecnologías familiares a tanto estudiantes como desarrolladores en el mercado laboral, y por el otro ser compatible con la infraestructura de la Escuela.

4. Detalles del proyecto

4.1. Requerimientos

A partir de las reuniones con las distintas partes interesadas de la Escuela, se identificaron tres tipos de usuarios y los siguientes requerimientos mínimos de la plataforma:

Tipos de usuarios:

- Por estudiante se entiende un alumno regular de la Escuela de Ingeniería UC cursando una malla curricular, sea de 2013, 2019 o de 2022.
- Por invitado se entiende una persona sin credenciales de acceso, como puede ser un estudiante de otra carrera, un futuro estudiante o personal administrativo.

- Por administrador, se entiende un usuario autorizado, que trabaja en la Escuela, que debe tener privilegios especiales.

Requerimientos funcionales

- Un estudiante deberá poder iniciar sesión con sus credenciales UC.
- Un estudiante o invitado deberá poder crear mallas, empezando a partir de los cursos ya tomados o siendo cursados en el caso del estudiante, o desde cero en el caso de un invitado.
- Un estudiante o invitado deberá poder escoger majors, minors o títulos y a partir de estos agregar, mover o reemplazar cursos que satisfagan los requisitos de la malla.
- Un estudiante o invitado deberá poder compartir sus planificaciones con otras personas.
- Un estudiante deberá poder guardar y volver a abrir sus planificaciones.
- Un administrador deberá poder revisar las planificaciones creadas por un estudiante, escogiendo a este por su RUT.
- Un administrador deberá poder ver la información de seguimiento curricular de un estudiante.
- Un administrador deberá poder crear nuevas planificaciones derivadas a partir de planificaciones ya creadas por estudiantes.
- Un administrador deberá poder crear y eliminar anuncios destinados a todos los usuarios de la plataforma.

Requerimientos no funcionales

- Alto grado de calidad de código, con un especial énfasis en documentación y extensibilidad.
- Una arquitectura con tecnologías relativamente ubicuas en tanto la Escuela como el mercado laboral de desarrolladores.
- Compatibilidad con la infraestructura existente de la Escuela. Específicamente, compatibilidad con un entorno Linux corriendo CentOS 7.

- Integración con el sistema CAS SSO UC, SIDING (para la información de mallas y cursos) y Seguimiento Curricular (para cursos pasados y carga académica).
- Acceso restringido a datos en el entorno de producción.

Adicionalmente, se identificaron además estas siguientes oportunidades de mejora sobre la funcionalidad del planner actual:

- Que un estudiante pueda ver visualmente las dependencias de los cursos, facilitando entender cómo se relaciona cada curso con el resto.
- Que un estudiante pueda acceder a información adicional de un curso, como su programa.
- Que un estudiante pueda estimar la carga semestral dado la carga académica que se ha identificado en semestres anteriores con la Encuesta de Carga Académica (ECA).
- Que un administrador pueda obtener una estimación de la demanda de cupos de cada curso dado las planificaciones seguidas por los estudiantes.
- Que el estudiante reciba advertencias en tiempo real de combinaciones permitidas, pero por distintas razones, poco recomendables.

4.2. Equipo y Metodología

El equipo de trabajo constaría de un profesor o personal administrativo de apoyo (product owner), un estudiante con el rol de coordinador de equipo, y cinco estudiantes como desarrolladores, contratados en modalidad análoga a la de ayudantes. Este modelo ya ha sido empleado con éxito en el caso de la Unidad de Comunicaciones y Difusión de la Escuela, CODI, y en el equipo de desarrollo de la plataforma Clearn, del Departamento de Ciencia de la Computación.

Se empleará una metodología basada en Scrum, teniendo como product owner a una persona, por definir, que actúe como punto de nexo con la Escuela. Esta persona deberá mediar la priorización de requerimientos durante las etapas de planificación, y otorgar retroalimentación después de cada sprint, opcionalmente involucrando a otros stakeholders en caso de ser necesario.

Para el reclutamiento de estudiantes, se creará un perfil del postulante y se reclutará utilizando los medios del Centro de Alumnos de Ingeniería, la Consejería Académica, las

instancias de desarrollo en los proyectos de cursos de programación, el Capítulo ACM, Open Source UC y los canales del Departamento de Ciencia de la Computación.

Una vez recibidas las postulaciones, dos personas evaluarán a cada candidato sobre la base de una entrevista técnica general, en complemento a su experiencia en cursos y proyectos pasados. Luego de elegir a los candidatos, se elaborará una propuesta de equipo, la cual deberá ser vetada por la Dirección de Pregrado, tomando en cuenta procesos abiertos en secretaría general y faltas al código de honor.

4.3. Línea de Tiempo

Preliminarmente, se estima un tiempo de desarrollo no superior a un año, pudiendo priorizar la funcionalidad necesaria para cubrir las necesidades básicas de las generaciones 2019 en adelante con tal de entregar una aplicación funcional dentro del primer semestre. Esta estimación, sin embargo, está fundamentalmente limitada por la falta de conocimiento sobre el equipo, la integración y la complejidad de ciertas combinaciones de requisitos curriculares.

Por esto, el primer mes y medio consistirá en una **marcha blanca**. Esta tendrá como objetivo evaluar la factibilidad, viabilidad, velocidad y riesgo del proyecto mediante la elaboración de un **Prototipo Mínimamente Viable (MVP)** que demuestre la funcionalidad crítica de la plataforma para un caso específico de un estudiante (major, minor, título, generación).

Este prototipo explícitamente no incluirá la funcionalidad de conexión con seguimiento curricular o de validación de reglas complejas y excepcionales de requisitos, entendido que ambos son fuertemente dependientes de la integración y esta no necesariamente estará lista durante el período de tiempo planificado.

Como mínimo, el MVP deberá poder:

- Ser capaz de iniciar sesión con sus credenciales UC.
- Visualizar una malla desde cero, y ser capaz de agregar, mover y eliminar cursos.
- La capacidad de escoger cursos que cumplen cada requisito en una malla específica, viendo si estos se encuentran satisfechos.
- La capacidad de guardar mallas y volver a abrirlas después.

El prototipo se desarrollará en dos sprints de tres semanas cada uno, tras la cual se presentarán los resultados y se evaluará la continuación del proyecto a base de los resultados. De decidir continuar con este, se hará una estimación de tiempo de desarrollo a partir de la velocidad promedio de este período, y se reevaluará el tamaño del equipo.

4.4. Desafíos de implementación

Se identifican varios desafíos que son fuentes de riesgo en el desarrollo de proyecto:

- **Contratación de estudiantes:** La contratación de estudiantes supone un riesgo en lo que respecta a la velocidad de desarrollo del proyecto, siendo estos afectados por exigencias académicas y una jornada laboral reducida y dinámica.

  → **Mitigación:** Se adaptará la metodología Scrum para acomodar mayor flexibilidad en el proceso de desarrollo como parte de los sprints. Igualmente, se trabajará alrededor de las fechas de mayor exigencia académica como parte de la planificación de cada sprint. Para evaluar la velocidad de desarrollo se dispondrá de un período de marcha blanca (véase "Línea de Tiempo") y se mantendrá visibilidad continua sobre el proyecto usando software como Asana o Linear.

- **Fuerte dependencia en servicios externos (integración):** Puede disminuir la agilidad en la medida que se necesite aprobación externa, como es el caso de CAS UC, que depende del equipo de Arquitectura e Integración de la Dirección de Informática, o en la medida en que se requiera desarrollo interno en paralelo, como es el caso particular de Seguimiento Curricular.

  → **Mitigación:** La integración tendrá máxima prioridad en los primeros ciclos de desarrollo y se designará un miembro del equipo para trabajar de forma dedicada en esta área junto a las distintas entidades involucradas.

- **Lógicas complejas de requisitos curriculares:** La complejidad asociada al sistema de requisitos, co-requisitos, restricciones, equivalencias y excepciones particulares implica que una parte importante del desarrollo es meramente mapear estos casos e implementarlos manteniendo generalidad. Esto es un riesgo importante porque tiene el potencial de retrasar significativamente el desarrollo.

  → **Mitigación**: Se intentará crear un marco flexible de modelación de requerimientos con especial cabida a excepciones que puedan identificarse más tarde en la implementación de las mallas curriculares. Será necesario un estudio temprano de las mallas y la lógica de negocios involucrada en las combinaciones posibles de cursos.

- **Transición (handoff):** Una vez finalizado el desarrollo será necesario traspasar control y mantenimiento de la plataforma al Núcleo de Desarrollo de la Escuela por lo cual el stack tecnológico escogido será crucial para asegurar que el equipo de recepción pueda navegar la plataforma.

→ **Mitigación:** La arquitectura escogida (véase "Arquitectura y entorno") fue diseñada teniendo en mente la popularidad de las herramientas utilizadas en el mercado de desarrollo en Chile. Adicionalmente, se dedicará un período explícito de handoff dónde ambos equipos trabajen en conjunto para asegurar una transición exitosa.

- **Mantenibilidad y extensibilidad**: Una vez finalizado el handoff, el Núcleo de Desarrollo tendrá que poder mantener la plataforma y poder extenderla si es necesario. Esto supone un riesgo si es que la documentación escasea o es de mala calidad, si es que la calidad de código es baja o si la plataforma no fue diseñada con generalidad en mente.

  → **Mitigación**: La metodología de desarrollo enfatizará en la creación de documentación suficiente como criterio de aceptabilidad a los requerimientos y se establecerán requerimientos estrictos de calidad de código y se tendrá un foco especial en la modularización y extensibilidad de la funcionalidad de la plataforma.

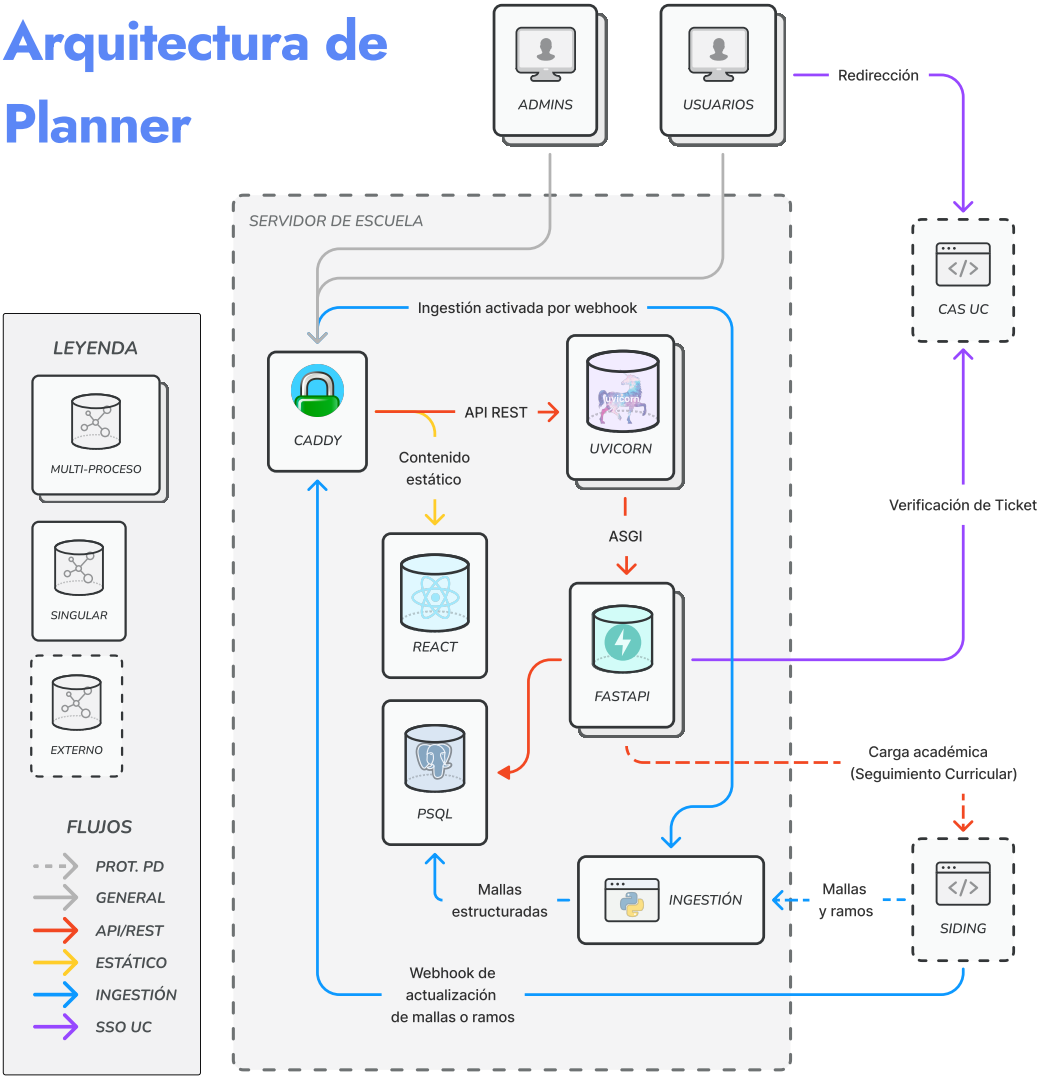
- **Accesibilidad y usabilidad**: La Escuela alberga una diversidad de estudiantes que requiere considerar criterios estrictos de accesibilidad y usabilidad de interfaces de la plataforma.

  → **Mitigación**: Se establecerá compliance con el estándar internacional WCAG 2 (ISO/IEC 40500) sobre accesibilidad de contenido web, implementando la verificación automática de partes del estándar dentro del sistema de integración continua del proyecto. Igualmente, se requerirá que al menos un miembro del equipo demuestre conocimiento relevante en buenas prácticas de experiencia de usuario (UX).

- **Seguridad de datos:** Si bien por diseño la plataforma no almacena datos críticos como podrían ser credenciales UC (véase "Arquitectura y Entorno"), si almacena las mallas curriculares de cada estudiante, al igual que credenciales de acceso a seguimiento curricular y SIDING, lo que amerita consideraciones especiales de seguridad.

  → **Mitigación**: Los entornos de desarrollo, staging y producción estarán segmentados entre ellos y se limitará el acceso directo a estos al coordinador del equipo de desarrollo. Se dispondrá de datos y endpoints de prueba con información falsa con tal de facilitar el desarrollo local del proyecto. Se utilizarán criterios de aceptabilidad que incluyan el uso de buenas prácticas de seguridad de la información.

4.5. Arquitectura y entorno



Se propone el uso de una arquitectura moderna, híbrida, altamente extensible y diseñada teniendo en mente la compatibilidad con la infraestructura existente de la Escuela (véase figura anterior).

Para este propósito, se utiliza una arquitectura clásica de proxy inverso (Caddy) que disminuye el acoplamiento del front-end del back-end mediante el uso de una API. Desde el proxy inverso se identifican tres rutas principales:

- Una ruta hacia el contenido estático de la página web (front-end), hecho utilizando React.
- Una ruta hacia el contenido dinámico de la aplicación, que dirige a Uvicorn (servidor HTTP/ASGI) que permite la comunicación con el back-end, escrito en Python utilizando el framework FastAPI.
- Un webhook que activa un script de ingestión, también escrito en Python, que descarga cambios de la información de planificación desde SIDING, transformando su estructura de datos y guardándolo en la base de datos de PostgreSQL. Este webhook permite a SIDING notificar actualizaciones en las mallas o cursos, sin incurrir en un llamado externo en cada solicitud al back-end o lentas transformaciones.

Todos los servicios internos (Caddy, Uvicorn, el back-end en FastAPI, PostgresSQL y el script de ingestión) se encuentran distribuidos en cuatro contenedores de Docker: proxy, ingestión, back-end y base de datos. Los contenedores permiten replicar localmente los entornos de desarrollo, staging y producción, y evitar potenciales incompatibilidades con el entorno del servidor, CentOS 7.

Se implementará pipelines de integración y deployment continuo que permitan iterar rápidamente y segmentar los entornos de deploy. Además, se limitará el acceso directo a estos entornos solamente al coordinador del equipo, y empleando una política de aprobación doble para deploys a producción, reduciendo el acceso a la administración de los entornos.

La arquitectura está compuesta exclusivamente de tecnologías maduras que tienen una historia de alta disponibilidad y rendimiento en entornos de producción. Crucialmente, no solo la arquitectura utiliza herramientas ya enseñadas en la Escuela, sino que también utiliza un stack tecnológico para el cual ya existe una significativa y creciente oferta de talento en el mercado de desarrolladores chilenos. Esto asegura la mantenibilidad futura del proyecto una vez que este haga una transición de control al Núcleo de Desarrollo de la Escuela.

Para entender los requerimientos de recuperación y resiliencia de la arquitectura, se realizó un análisis de impacto de negocios (BIA), que buscó determinar la criticidad de la plataforma y potenciales objetivos de tiempo de recuperación (RTO) y de punto de recuperación (RPO). De esto se concluyó que no se encuentran justificadas medidas avanzadas de recuperación de la plataforma, por lo que basta con atenerse a una política simple de copia de seguridad fuera de sitio. La arquitectura también incluye chequeos de salud automáticos y la regeneración automática de contenedores.

5. Conclusión

En esta propuesta, planteamos una alternativa al sistema de planificación actual de la Escuela, la cual presenta una profunda deuda técnica y carece de la funcionalidad para servir a las nuevas generaciones. Para solucionar este problema, se propone el desarrollo de una nueva plataforma, creada por estudiantes y para estudiantes, con un especial énfasis en mantenibilidad y extensibilidad. De esta manera, e incorporando elementos de la metodología ágil y prácticas de desarrollo moderno, se considera viable reemplazar el actual planner con esta nueva plataforma.

6. Agradecimientos

Se agradece las contribuciones de las siguientes personas en la propuesta:

- Ria Deane (CA)
- Agustín Covarrubias (OSUC)
- Benjamín Vicente (OSUC)
- Caua Paz (OSUC)
- Lucas Natero (OSUC)
- Alister Mac Cormack (OSUC)
- Fernando Smith (OSUC)
- Nicolás Berríos (OSUC)
- Vicente Lavagnino (CA)
- Ignacio Laval (CA)