

# Modelado de currículum y trayectorias

Taller de Ingeniería Dirigida por Modelos (TMDE) - Facultad de Ingeniería, Udelar - 2023

Santiago Nicolás Díaz Conde

*Universidad de la República*

santiago.nicolas.diaz.conde@fing.edu.uy

Santiago Freire López

*Universidad de la República*

santiago.freire@fing.edu.uy

**Abstract**—Este trabajo propone un enfoque basado en Model-Driven Engineering (MDE) para mejorar la toma de decisiones de los estudiantes en su trayectoria académica y poder visualizarla de una forma gráfica y personalizada. El enfoque MDE propuesto incluye un metamodelo para carreras, unidades curriculares y trayectorias de estudiantes, con lo que, procesando modelos que satisfacen estos metamodelos, permiten generar diversas salidas, como visualizaciones gráficas del currículum, planificaciones académicas interactivas, información de carreras, unidades curriculares, planes, entre otros. Este enfoque busca proporcionar herramientas más efectivas para que los estudiantes tomen decisiones informadas y planifiquen sus actividades académicas de manera más eficiente.

## I. INTRODUCCIÓN Y DESCRIPCIÓN DEL PROBLEMA

La disponibilidad de herramientas en línea destinadas a facilitar el seguimiento del progreso académico de los estudiantes de Ingeniería en Computación ha experimentado un notable incremento con el paso del tiempo. Sin embargo, a pesar de la presencia de estas plataformas, existen limitaciones sustanciales que impactan la experiencia del estudiante y la calidad de la información proporcionada. Este proyecto surge en respuesta a las carencias identificadas en las soluciones existentes, con el propósito de brindar una solución dinámica que solucione las problemáticas identificadas.

La primera limitación de estas herramientas existentes es que la información proporcionada es estática. Esto significa que los datos no se actualizan en tiempo real, lo que puede llevar a los estudiantes a tomar decisiones basadas en información obsoleta o incorrecta. Además, la información de las unidades curriculares proviene del programador y no de una fuente autorizada, por ejemplo Bedelías. Esto puede resultar en discrepancias entre la información proporcionada por la herramienta y la realidad académica del estudiante, lo que puede causar confusión y errores en la planificación académica.

En segundo lugar, estas herramientas no permiten a los estudiantes guardar su progreso para consultarlo o agregar datos más tarde. Esto dificulta el seguimiento del avance académico a largo plazo y limita la capacidad del estudiante para planificar su futuro académico de manera efectiva.

Luego, las herramientas existentes no proporcionan información detallada sobre cómo los estudiantes han pasado

sus materias, ya sea por curso o examen, ni la nota obtenida o la fecha de aprobación. Esta información es fundamental para que los estudiantes comprendan su rendimiento académico y planifiquen su futuro académico de manera efectiva. Actualmente, los estudiantes deben obtener esta información de su escolaridad, lo que representa un paso adicional en el proceso de planificación académica y puede ser una fuente de inconvenientes. Esta falta puede impedir que los estudiantes identifiquen unidades curriculares a cursar en sus áreas de interés basándose en su rendimiento académico.

Por último, estas herramientas están diseñadas específicamente para el plan 1997 de Ingeniería en Computación, y no se pueden aplicar a otras carreras, facultades o universidades. Esto limita su aplicabilidad y utilidad para un gran número de estudiantes que pueden beneficiarse de tales herramientas. Además, con la salida próximamente del nuevo plan de estudios ellas quedarán obsoletas, y el trabajo que puede ser rearmarlas casi desde cero puede ocasionar que nunca sean actualizadas.

En este contexto, este proyecto busca abordar estas limitaciones y proporcionar una solución más eficiente y efectiva. El objetivo es desarrollar una herramienta interactiva y dinámica que permita a los estudiantes de diversas disciplinas académicas visualizar y seguir su progreso académico, con información actualizada y autorizada. Este proyecto no solo mejorará la experiencia académica de los estudiantes, sino que también les permitirá tomar decisiones informadas y planificar su futuro académico de manera más eficiente. Al hacerlo, este proyecto tiene la capacidad de potenciar la forma en que los estudiantes interactúan con su trayectoria académica, mejorando su capacidad para alcanzar sus metas y tener éxito en su recorrido académico.

## II. SOLUCIÓN Y EJEMPLOS

La solución que se desarrolló es una plataforma web de creación, modificación y consulta de modelos de estudiante, facultades, carreras, planes y demás, con un enfoque de microservicios en base a Java Servlets. Todo el procesamiento (en backend) se realiza con MDE mediante la librería EMF de Java. No hay bases de datos, se hacen consultas o escrituras directas al modelo y de allí se devuelven los resultados.

Como fue mencionado antes, el sistema está basado en microservicios, por lo que las funcionalidades están brindadas como un conjunto de pequeños servicios ligeros. Esto significa que, por ejemplo, no se guardan los datos de los usuarios (por ejemplo, modelos de trayectoria de estudiantes) internamente para después recuperarlos de allí. En cambio, en el caso de uso *Crear Modelo* se crea un modelo en formato *.xmi*, que se descarga al navegador del usuario. Luego para los casos como *Inscripción a Plan* o *Ver Currícula*, el estudiante debe cargar el modelo que tiene su trayectoria, el sistema hará los procesamientos correspondientes en el backend y si corresponde, descargará un nuevo archivo de modelo con los cambios.

La arquitectura del sistema está dividida en tres capas:

### A. Capa 1: Backend con Eclipse Modeling Framework (EMF) y Java Servlets

En esta capa se definieron los modelos y metamodelos. Como fue mencionado en la Sección I, existen dos metamodelos: el de facultades y el de estudiante. Este último tiene referencias al primero, en la parte de inscripciones a planes y evaluaciones tomadas. Esto permite separar la aplicación en dos actores:

En primer lugar, tenemos a un actor autorizado para modificar la información de las carreras, que carga la información relevante de las unidades curriculares, cursos, evaluaciones y planes académicos. Un ejemplo de este actor podría ser Bedelías u otra entidad encargada de gestionar los datos académicos. En segundo lugar está al estudiante, quien registra su propia información de los planes, cursos y exámenes previamente ingresados por el actor autorizado.

Esta división de roles delega responsabilidades de manera eficiente, evitando carga adicional de trabajo para los usuarios y asegurando una gestión efectiva de la información académica. Además, elimina la problemática de la información cargada por una fuente no autorizada, ya que se puede delegar esta parte a quien corresponda sin afectar el funcionamiento correcto del resto del sistema.

El metamodelo del estudiante es como sigue:

En este metamodelo se guardan las inscripciones a planes y cursos de los estudiantes, así como las evaluaciones tomadas por él. Cabe destacar que los atributos *Plan*, *CUCourse* y *evaluation*, así como fue explicado anteriormente, hacen

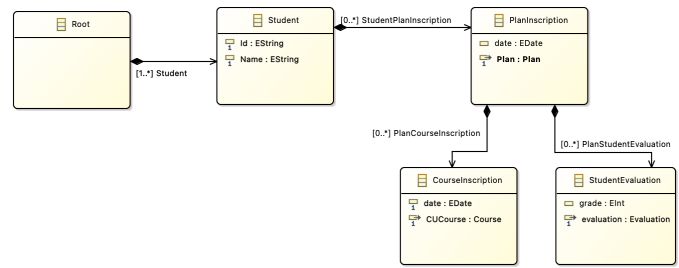


Fig. 1. Metamodelo de estudiante.

referencia a elementos del otro metamodelo.

El metamodelo de facultades, al ser muy grande, no se mostrará en su totalidad - se puede visualizar en el repositorio de Gitlab -, pero se destacarán y explicarán ciertos aspectos relevantes.

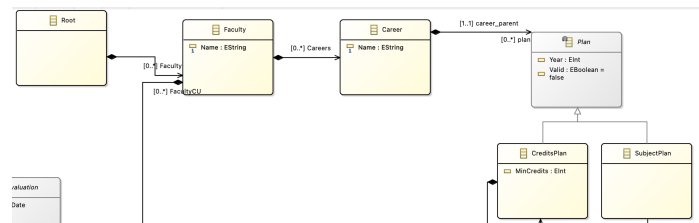


Fig. 2. Facultades en el metamodelo.

Dentro de este segundo metamodelo, tenemos que las facultades tienen carreras, que tienen planes, quienes a su vez pueden ser de dos tipos: de créditos, como la gran mayoría de los planes actuales, o de materias. De esta manera se pueden representar currículas de muchos casos.

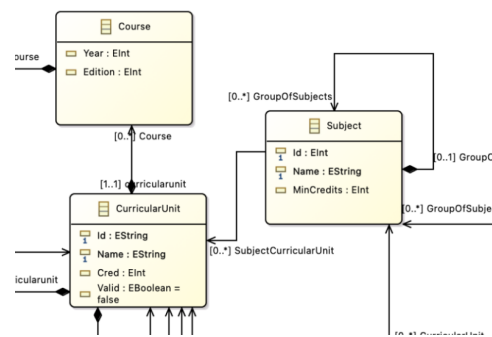


Fig. 3. Materias, cursos y UCs en el metamodelo.

En cada plan de carrera existen materias. Las materias son grupos de unidades curriculares, que no se deben confundir con las UCs mismas. Ejemplos de esto en Ing. en Computación pueden ser Materias Básicas, Programación, Arquitectura, Sistemas Operativos y Redes de Computadoras, entre otras. Ellas están compuestas de determinadas unidades curriculares, y hay un mínimo de créditos por cada una que

se deben cumplir para poder obtener el título.

Las unidades curriculares pueden ser válidas o no, en el sentido de que ya no son más dictadas pero se siguen considerando en la trayectoria del estudiante. Un ejemplo de esto pueden ser los viejos Cálculos 1, 2 y 3 vs los nuevos CDIV, CDIVV y CV. Este atributo toma especial importancia en la capa 3, como se verá más adelante, para no mostrar información desactualizada a la hora de mostrar materias posibles a cursar (y que en definitiva, no les sirve a los estudiantes verlas, pues no podrán inscribirse a ellas).

A la vez, los dictados de estas unidades curriculares se dan en cursos, que tienen una edición y año - por ejemplo, GAL 2 2020 edición 1 (semestre impar) y GAL 2 2020 edición 2 (semestre par).

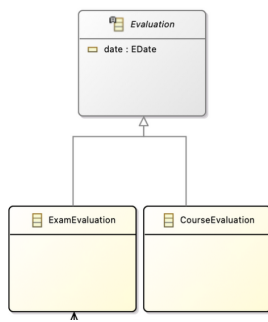


Fig. 4. Evaluaciones en el metamodelo.

Las evaluaciones se dividen en dos tipos: de curso y de examen. Las evaluaciones de curso se corresponden con un curso de unidad curricular, y es la nota final que obtuvo un estudiante en una cursada. En este caso, una nota mayor o igual a 3 significa una *aprobación de curso*, y una nota mayor o igual a 6 *exoneración de curso*, que a la vez equivale a una aprobación de examen. Las materias sin nota y examen obligatorio se pueden representar utilizando únicamente evaluaciones de examen y, si el estudiante obtuvo el curso, con evaluaciones de curso con nota 3.

Para el caso de los exámenes, ellos se corresponden con una unidad curricular, y una nota mayor o igual a 3 equivale a una aprobación. No se vinculan con un curso porque un estudiante puede ganar el derecho a examen en cierto momento, y dar el examen cuando se está dictando otra edición. Los exámenes son independientes a las ediciones y por lo tanto se vinculan directo con la UC.

Por último, las unidades curriculares pueden tener requerimientos para cursarlas, que a su vez son de distinto tipo:

- RegisteredTo: Estar registrado en una UC
- SomeOf: Cumplir al menos N de los requerimientos que lo componen
- NOT: No cumplir un requerimiento
- Coursed: Tener el curso aprobado de una materia

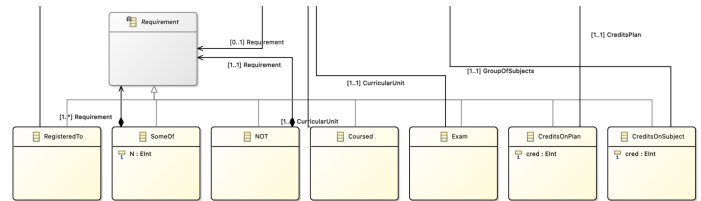


Fig. 5. Requerimientos en el metamodelo.

- Exam: Tener el examen aprobado de una materia
- CreditsOnPlan: Tener *cred* créditos aprobados en el plan
- CreditsOnSubject: Tener *cred* créditos aprobados en cierta materia

Por la forma de nuestro modelado, cada materia tiene directamente solo un requerimiento asignable, por lo que para incluir varios requerimientos - por ejemplo, tener aprobadas tres materias específicas - es necesario el uso de SomeOf con su valor de N correspondiente.

Con estos metamodelos, ejemplos de - parte de - los modelos de facultades y estudiante pueden ser los siguientes:

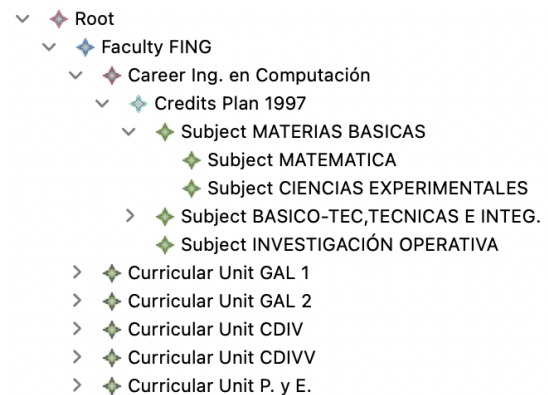


Fig. 6. Modelo de Facultades.

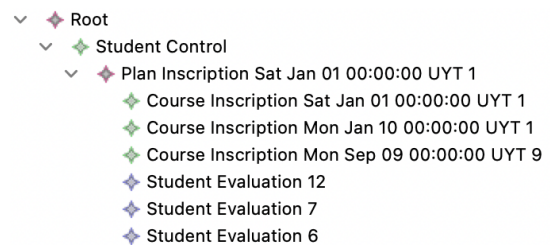


Fig. 7. Modelo de Estudiante.

De estos resta ver los atributos de cada parte del modelo; cuáles son específicamente y sus tipos se pueden ver en base al metamodelo en figuras anteriores.

En lo que a las operaciones con los metamodelos y modelos respectan, se implementaron mediante la librería Java Eclipse Modeling Framework, y Java Servlets. De esta manera todo el

backend está basado en las mismas tecnologías. Asimismo las operaciones que devuelven información lo hacen en formato JSON, para que sea simple de analizar (*parse*) en el frontend.

### B. Capa 2: OpenAPI

Para poder interactuar con el metamodelo y los modelos, tanto en operaciones de lectura como escritura, definimos una API mediante la especificación OpenAPI para poder hacerlo.

Todas las operaciones del frontend que involucran obtener o enviar datos al backend se hacen mediante las operaciones de esta API. Asimismo, los Servlets del backend tienen su comportamiento como está definido en esta API, con las operaciones y atributos que allí se especifican.

Las operaciones de esta API se pueden dividir en dos grandes partes, según el método HTTP que ellas utilicen.

1) *Operaciones GET*: Las operaciones GET son las que no requieren enviar información adicional al servidor; en este caso, un modelo de estudiante. Por lo tanto con ellas se puede obtener información del modelo de facultades, como puede ser consultar una Unidad Curricular, Materia, Plan, etc.

2) *Operaciones POST*: Las operaciones HTTP POST son las que envían datos al servidor en su solicitud; que en este caso, es el modelo del estudiante. Por lo tanto ellas se utilizan para realizar las consultas y modificaciones a este modelo, como puede ser inscripciones a planes, agregar evaluaciones, evaluar si se puede cursar una unidad curricular, mostrar el grafo con la trayectoria del estudiante, etc.

Las operaciones específicas son muchas, y para mayor detalle se pueden consultar en el archivo *openapi.yml* del GitLab del proyecto, o en la documentación en HTML, también en el GitLab del proyecto.

### C. Capa 3: Frontend

El frontend de la aplicación se desarrolló con HTML, CSS y JavaScript. Ella da una interfaz gráfica a las operaciones de la API y las presenta de forma amigable al usuario.

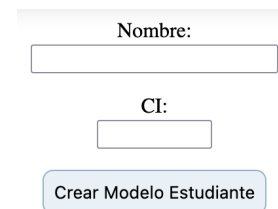
Los casos de uso disponibles son los siguientes:

- 1) Crear Modelo - Crea un modelo nuevo de estudiante
- 2) Agregar Inscripción a Plan - Inscribe al estudiante al plan de carrera seleccionado
- 3) Agregar Evaluación - Agrega una evaluación (de curso o examen) con su nota al estudiante
- 4) Evaluar Unidad Curricular - Evalúa si el estudiante puede cursar una Unidad Curricular
- 5) Ver Currícula - Muestra un grafo con el progreso actual del estudiante y las materias que puede cursar
- 6) Consultar Facultad - Muestra información de una facultad

- 7) Consultar Plan - Muestra información de un plan
- 8) Consultar Materia - Muestra información de una materia
- 9) Consultar Unidad Curricular - Muestra información de una UC
- 10) Ver Currícula de Plan - Muestra un grafo con las UC de un plan y las previas de cada uno
- 11) Ver Documentación - Muestra la documentación de la API

Pasaremos a mostrar algunos casos de uso relevantes para que se vea cómo nuestra solución resuelve las problemáticas para las que fue creada en un principio.

#### Crear Modelo



Nombre:

CI:

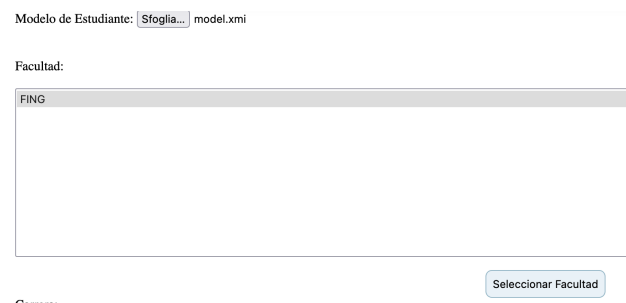
Fig. 8. Caso de uso Crear Modelo.

El estudiante ingresa su nombre y número de documento, se crea el modelo base y se descarga a su equipo. Luego para agregar su inscripción a plan deberá subir ese modelo.

#### Agregar Evaluación

*Nota: Para que este caso de uso funcione, el estudiante debe estar inscripto en el plan al que agrega la evaluación.*

El estudiante selecciona su facultad, carrera y plan al que está inscripto, luego elige una unidad curricular y evaluación de ella, ingresa la nota que se sacó y se registra en el modelo. Estos datos son los que luego se toman en cuenta en casos como Evaluar Unidad Curricular o Ver Currícula.



Modelo de Estudiante:  model.xml

Facultad:

Carrera:

Fig. 9. Subida de modelo y selección de Facultad

Destacamos que en la Figura 11 se puede seleccionar y distinguir entre evaluaciones de curso o examen mediante el slider que allí se encuentra.

#### Evaluar Unidad Curricular

El estudiante sube su modelo, selecciona una facultad y

Carrera:

Ing. en Computación  
 Ing. Eléctrica

Plan:

1997

Selección de carrera y plan

Fig. 10. Selección de carrera y plan

Unidad Curricular:

GAL1  
 GAL2  
 CDIV  
 CDIVV  
 PYE  
 MD1  
 MD2  
 LOGICA  
 CAL1  
 P1

Curso Examen

Selección de unidad curricular

Evaluación:

Thu Jan 02 00:00:00 UYT 2020

Nota: 9

Agregar Evaluación

Fig. 11. Selección de UC, evaluación e ingreso de nota

unidad curricular, y en base a las previas de la UC y las evaluaciones registradas en el modelo, se devuelve si puede cursar la materia seleccionada.

Unidad Curricular:

P1  
 P2  
 P3  
 P4  
 PROGFUN  
 PROLOG  
 TPROG

Evaluar previas

No está habilitado para cursar la materia.

Fig. 12. Evaluar UC

## Ver Currícula

El estudiante carga su modelo, elige una facultad, carrera y plan, y en base a las materias que tiene aprobadas (ya sea solo curso o también examen) muestra un grafo con su trayectoria y las materias que puede cursar. Las materias con

examen aprobado se muestran en verde, con curso aprobado en amarillo, y las que se pueden cursar en blanco. Una flecha con línea punteada indica previatura de curso y flecha con línea continua previatura de examen. Los elementos del grafo se pueden mover horizontalmente para una mejor visualización.

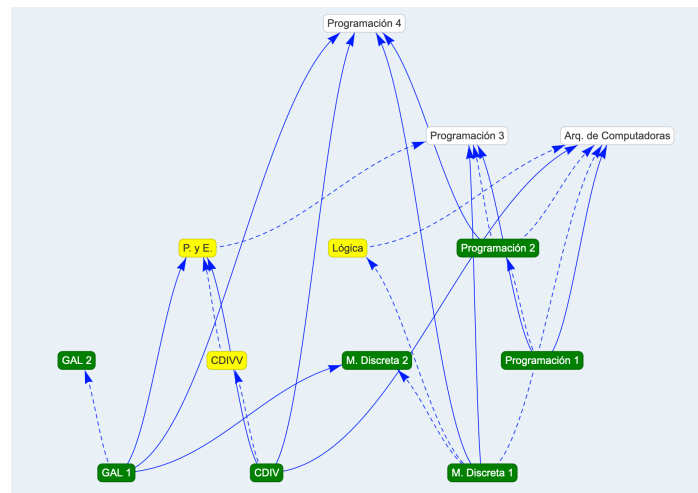


Fig. 13. Grafo de trayectoria

## Consultar Materia

Se selecciona una facultad, carrera, plan y materia (por su número de identificación), y se muestran las UCs que lo componen, nombre, mínimo de créditos y si contiene otras materias.

Materia:

1000  
 1001  
 1002  
 1010  
 1011  
 1012  
 1013  
 1014

Consultar Materia

Id: 1012  
 Nombre: ARQUIT, S.OP Y REDES DE COMP.  
 MinCredits: 30  
 Materias:  
 Unidades Curriculares:  
 - ARQCOMP  
 - SISTOPER  
 - REDESCOMP

Fig. 14. Consultar Materia

## Ver Currícula de Plan

Se selecciona una facultad, carrera y plan, y se muestran las UCs del plan con las previas entre ellas. El orden vertical es generado automáticamente en base al número de requerimientos, aunque para el caso de Ing. en Computación lo ajustamos para las materias que solo se dictan en cierto semestre. Nuevamente flecha con línea punteada indica previatura de curso y flecha con línea continua previatura de examen, y los elementos del grafo se pueden mover horizontalmente para una mejor visualización.

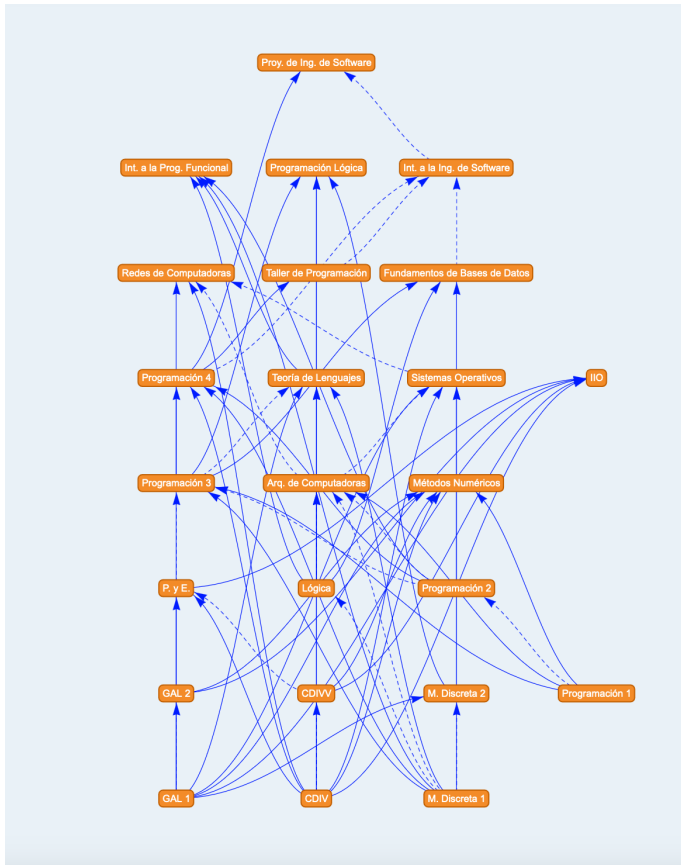


Fig. 15. Ver Currícula de Plan

### III. CONCLUSIONES

En conclusión, esta implementación para el modelado de currículo y trayectorias con enfoque MDE demostró ser una buena solución para abordar las limitaciones identificadas en las herramientas existentes de registro de progreso académico.

La motivación del proyecto se basa en la necesidad de los estudiantes para tomar decisiones informadas sobre y planificar eficientemente sus actividades académicas. La posibilidad de visualizar gráficamente la trayectoria académica, junto con la información actualizada y autorizada, supera las limitaciones de las herramientas estáticas existentes y ayuda a lograr estos objetivos.

Además, la posibilidad de guardar el progreso y ajustar planes académicos según sea necesario mejora la experiencia del usuario y la eficiencia en la planificación a largo plazo. Esto proporciona un seguimiento personalizado y una mayor flexibilidad en la gestión de la trayectoria académica.

La aplicación de MDE en este contexto resultó útil, ya que permitió universalizar la forma de ingresar datos de carreras, como unidades curriculares y sus requerimientos, que lo hace menos propenso a errores y en consecuencia aumenta la productividad.

En definitiva el proyecto logró proporcionar una solución integral que no solo aborda las limitaciones identificadas en las herramientas existentes, sino que también ofrece beneficios significativos para los estudiantes al mejorar la toma de decisiones y la planificación académica. La aplicación de MDE demostró ser una estrategia efectiva en este contexto, ofreciendo una plataforma dinámica y eficiente para el seguimiento y gestión de los datos.

### REFERENCES

- [1] Sitio de la carrera Ing. en Computación: <https://eva.fing.edu.uy/course/view.php?id=800&section=3>
- [2] OpenAPI Initiative - <https://www.openapis.org>
- [3] Eclipse Modeling Framework (EMF) - <https://eclipse.dev/modeling/emf>