Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

T E S I S

“Diseño e Implementación de un Asistente Académico Basado en Autómatas Finitos”

Presenta: *Leonardo Fuentes Bueno*

Para obtener el grado de: Licenciado en ingeniería en Ciencias de la Computación

Director: *María Teresa Torrijos Muñoz*

Codirector: *Mireya Tovar Vidal*

Puebla, Pue, noviembre 2025

Agradecimientos

Para este espacio quisiera darle las gracias a mi madre por siempre seguirme apoyando a pesar de la distancia, siempre estuvo para mí en momentos de tensión y dificultades, sin ella este texto solo quedaría en una ilusión, a mi padre por todos los días que siempre se quitaba horas de sueño para permitirme descansar más aquellas noches de desvelo. También mencionar el gran apoyo de mi hermana y de mis tías que sin importar la situación siempre estaban para mí.

Al maestro Guillermo Marín Dorado que aprecio mucho y que siempre estaré agradecido por brindarme la oportunidad de trabajar para él, de compartirme sus conocimientos y de su valiosa recomendación para que yo realizara esta obra.

Agradezco enormemente a la Doctora María Teresa Torrijos Muñoz que siempre brindo de su valioso tiempo y paciencia para mí y este trabajo.

Por último, agradecer a todos los docentes por sus conocimientos impartidos y que por ellos obtuve la habilidad de desarrollar esta obra de la mejor manera.

Índice general

Tabla de contenido

Introducción 4
Estructura de la tesis 6
Capítulo 1: Objetivos y estado del campo del arte 8
Objetivos 8
Objetivo general 8
Objetivos especificos 8
Estado del arte 9
Capítulo 2: Modelado de procesos y requerimientos 13
Modelado de procesos 13
Requerimientos 19
Requerimientos generales 19
Descripción general 20
Requisitos específicos 23
Capítulo 3: Análisis y diseño 31
Metodología de desarrollo 32
Diagramas 33
Diagrama de casos de uso 33
Diagramas de secuencia 35
Diagrama de estado 41
Diseño de interfaces 41
Diseño de base de datos 51
Capítulo 4: Heurística 57
Capítulo 5: Pruebas 67
Caso ordinario 67
Caso regular 72
Caso semi irregular 76
Caso irregular 80
Conclusión y trabajo a futuro 83
Referencias 86

Introducción

En algunas instituciones públicas de nivel superior, uno de los principales problemas que enfrentan los estudiantes es el rezago académico. Según Cisneros, Rodríguez, Niño y Cuevas (2023) en un caso de estudio identificaron que la falta de orientación respecto a las materias que deben cursar o las alternativas disponibles para completar su formación genera retrasos significativos en la culminación de su plan de estudios, también mencionan que al no contar con una guía vocacional previa o durante la carrera universitaria puede generar una indecisión y eventualmente el abandono.

Por otro lado, está la problemática de la deserción estudiantil vinculada a la falta de orientación respecto a la trayectoria académica, lo que se evidencia en los datos proporcionados por el Instituto Nacional de Estadística y Geografía (INEGI, 2024), donde el porcentaje de deserción en el nivel educativo superior en Puebla ha fluctuado en los últimos años, pasando de un 7.8% en el ciclo 2020/2021 a un 8.0% en el ciclo 2023/2024.

Según Kamble, Salunke, Nalawade, Dedgaonkar, Mahadik y Shelke (2024) bajo este contexto, la implementación de autómatas finitos, máquinas de estados y otras técnicas provenientes de la teoría de autómatas podría ofrecer soluciones innovadoras para abordar esta problemática. Estas herramientas permiten modelar procesos educativos mediante la generación de grafos y trazado de rutas óptimas, facilitando la orientación de los estudiantes para que sigan trayectorias

académicas adecuadas, evitando la pérdida de materias o retrasos en su titulación.

Conscientes de esta problemática, en este proyecto se plantea el desarrollo de un sistema basado en autómatas finitos, que permite a los estudiantes trazar rutas académicas óptimas en función de su situación académica actual. Este busca ofrecer una solución dinámica y personalizada, considerando factores críticos como la carga académica, la disponibilidad de materias y los prerrequisitos establecidos, proporcionando así una herramienta eficaz para la toma de decisiones académicas informadas.

Estructura de la tesis

Capítulo 1

En el capítulo uno se describe el objetivo general y especifico del proyecto, además también se presentan algunos trabajos relevantes de diferentes investigadores los cuales tuvieron problemáticas acerca del rendimiento de un programa o funcionalidad y como su solución se basó en autómatas finitos deterministas o heurísticas.

Capítulo 2

En esta sección se presenta el cómo dado los requerimientos del proyecto se empieza a generar un diagrama de funcionamiento en el cual se describen los agentes y la funcionalidad que tiene cada uno de ellos para que el proyecto funcione de buena manera, esto con ayuda del modelo y notación de procesos de negocios (BPMN).

Capítulo 3

Para este capítulo se verá cómo se dividen de mejor manera los requerimientos para tener puntos críticos antes de empezar a desarrollar, para esto se aplican casos de uso donde se analizan los flujos que puede tener la aplicación, las especificaciones y validaciones que se requieren para tener un programa limpio.

Capítulo 4

En este apartado se realiza el diseño de la base de datos con ayuda de diagramas entidad relación donde se analiza el número de tablas y las relaciones entre ellas, dada una idea de cómo se puede construir la base de datos se muestra cómo se puede enriquecer de información a través de una interfaz web.

Capítulo 5

En este capítulo se muestra como con base en diagramas y en el análisis previamente realizado se empieza a desarrollar pantallas por parte del frontend y la heurística que se utilizará para la generación de una ruta académica que corresponderá al backend o lógica del proyecto.

Capítulo 6

Por último, se analiza el funcionamiento del sistema de manera local para comprobar que genere una ruta académica correcta, que no haya errores de validación y que si se guarde la información correctamente en la base de datos.

Capítulo 1: Objetivos y estado del campo del arte

Objetivos

Objetivo general

Implementación de un modelo de autómata finito determinista que permita orientar a los estudiantes de nivel superior para elegir una ruta académica adecuada por la cual puedan transitar de acuerdo con las condiciones personales que viven en cada periodo de reinscripción.

Objetivos especificos

1.- Modelar un autómata finito que represente el avance académico de un estudiante considerando estados como materias aprobadas, reprobadas, pendientes y alternativas para recuperación.

2.- Desarrollar un motor de reglas que evalué los requisitos previos y determiné si el estudiante puede o no inscribirse a una materia.

3.- Implementar algoritmos que analicen el camino más eficiente para que un estudiante logre culminar su carga de materias en el menor tiempo posible considerando su estado actual.

4.- Diseñar una interfaz gráfica que permita a los estudiantes visualizar su progreso, identificar bloqueos y ver las rutas recomendadas.

Estado del arte

Los avances tecnológicos han impulsado el desarrollo de algoritmos computacionales capaces de resolver problemas de mejor manera que el ser humano para ahorrar tiempo y esfuerzo, como ejemplo están los autómatas finitos que han desempeñado un papel importante en el procesamiento de lenguajes formales, trazado de rutas, optimización entre otras. A medida que las Ciencias Computación ha evolucionado, los autómatas han pasado de ser simples modelos teóricos a ser herramientas clave en la implementación de programas, inteligencia artificial y control de procesos que resuelven problemas y permiten la innovación. Su integración en aplicaciones modernas demuestra como la combinación de teoría y tecnología ha permitido optimizar el rendimiento y la eficiencia en múltiples áreas.

Un ejemplo de uso de autómatas finitos fue en el problema de búsqueda de patrones difusos de un texto, según Kostanyan (2017) los autómatas se ocuparon para modelar transiciones que permitieron identificar ocurrencias del patrón difuso del texto conforme a un umbral de similitud. Como ejemplo se muestra una definición de símbolos difusos como SMALL y LARGE que se les asigna unos valores de pertenencia, el autómata procesa el texto carácter por carácter de manera que actualiza los estados que representan grados de coincidencia, dando como resultado la identificación de posiciones validas donde el patrón ocurre con un umbral de similitud. Con esta implementación de autómatas se pudo reducir el costo computacional.

Esta investigación hace ver importante el uso de autómatas para optimizar un proceso de identificación y reducir la complejidad del algoritmo de búsqueda, además de que contribuye a la teoría de que un autómata finito determinista pueda ser útil para la identificación de materias o cursos similares de un estudiante y que además al generar una ruta o alternativa al usuario esta se genere de manera óptima ya que no se consideran las especificaciones del servidor que ejecute el algoritmo, por lo que es importante tomar en cuenta la complejidad del algoritmo.

Un trabajo realizado por Zare, Jampour, Arezoomand y Sabouri (2019) de Shiraz, Irán era reducir la incertidumbre en el reconocimiento de caracteres manuscritos, esto debido a que en movimientos en la mano al escribir letras diferentes como la “Q” y la “O” son difíciles de distinguir. La manera en que detectaban la escritura era en un programa donde al escribir la letra el sistema mandaba la letra escrita en un mensaje, pero no siempre era correcta, debido a este problema decidieron combinar 3 técnicas: La detección de bordes, lógica difusa (fuzzy) y autómatas finitos deterministas para modelar secuencia de movimientos y asignar letras específicas. Se empleo un DFA matemático el cual usa una máquina de estados finitos, Con esta implementación los DFA permitieron modelar el orden de movimientos para cada letra, esto fue importante ya que distinguieron caracteres con movimientos similares con un margen del 74.19% y lograron procesar secuencias de movimiento en tiempo real mejorando la eficiencia del sistema.

Este trabajo también deja en claro como el uso de los autómatas finitos determinísticos contribuyen a la detección de patrones y a la optimización

computacional, de igual manera se menciona la técnica fuzzy que será importante para la lógica del autómata a realizar para determinar si una materia se puede tomar o no dada la situación actual del estudiante (aprobado o no).

Otro caso relacionado con aplicaciones móviles fue presentado por la Facultad de Física e Ingeniería Electrónica de Chongqing Three Gorges University en china, los investigadores Guoping, Xiaowei, Minlu, Zefu, Hong (2011) encontraron un problema con el sistema CMMB (China Mobile Multimedia Broadcasting) el cual transmite audio, video y datos multimedia a dispositivos móviles mediante canales terrestres o satélites, durante la recepción de flujo de datos se presentaban errores debido a interferencias o ruidos dificultando la demultiplexación para recuperar información útil, el desafío era gestionar estos errores de manera eficiente para que así se garantizara una transmisión de datos continua y de calidad. Se implemento el uso de autómatas finitos deterministas para clasificar y manejar errores en tiempo real durante la demultiplexación donde definen varios estados para representar diferentes fases de proceso de demultiplexación y posibles errores permitiendo que el AF funcione en paralelo con el proceso de verificación CRC (decodificación CRC-32), si los errores están dentro del alcance de corrección de CRC el flujo se repara automáticamente, si están fuera del alcancel el autómata toma el control. De esta manera el lograron reducir la complejidad simplificando el manejo de errores heterogéneos y ayudaron a que el sistema tolerara mejores flujos corruptos minimizando interrupciones y garantizando una mejor fluides al usuario.

Este otro ejemplo impulsa la idea de que los autómatas finitos son buenos para la clasificación y control de errores, para el proyecto en mente esta teoría aporta a la idea de crear grupos de materias en común y así hacer eficiente la lógica del autómata finito a implementar.

Un trabajo realizado recientemente es el de los investigadores Dwinandana, Reska y Sulistyo (2022) los cuales en tiempos de pandemia por el COVID-19 se les encomendó mejorar la aplicación móvil PeduliLindungi, esta sirve para monitorear y registrar la actividad de los usuarios con el fin de evitar la propagación del virus, la tarea a realizar era hacer un testing eficiente para garantizar su calidad y evitar hacer testeos manuales ya que eran lentos y repetitivos.

Una mejora realizada fue el modelado de funciones CRUD con el autómata Extended Finite State Machine (EFSM), el cual incluye condiciones de guardia y acciones en las transiciones permitiendo modelar sistemas donde algunas acciones dependen de condiciones previstas. El algoritmo fue implementado en 3 funcionalidades de la aplicación (vacunación, apartado eHAC y perfiles), estos se modelaron como grafos con nodos y aristas, donde las transiciones incluían condiciones de guardia. Después para la estrategia del testing usaron el software GraphWalker el cual genera casos de pruebas basado en modelos EFSM. Con estas 2 implementaciones lograron reducir el tiempo de las pruebas y con los porcentajes obtenidos en cada funcionalidad lograron enfocar más esfuerzo en apartados críticos del CRUD. De esta manera demostraron que los autómatas

finitos extendidos son efectivos para simular y automatizar testeos en aplicaciones móviles con gestión de datos.

Esta investigación aporta mucho para la fase 4 de la metodología que se planea seguir para este proyecto ya que con un autómata finito extendido y con el uso del software GraphWalker se puede ver la eficiencia en cada área que se vaya a desarrollar como el registro de datos de un alumno, su carga académica o las materias faltantes por cursar, de este modo se podrá analizar en que funcionalidad se requiere más dedicación.

Estos fueron algunos casos como ejemplo a tomar para demostrar que los autómatas finitos deterministas pueden ser de gran utilidad para resolver problemas de optimización, errores o para considerar casos especiales y tomar una decisión. OBJ

Capítulo 2: Modelado de procesos y requerimientos

Modelado de procesos

Analizando el problema de elección de materias y de rezago que viven los estudiantes de la Facultad de Ciencias de la Computación se pensó en un proyecto el cual ayudara a la construcción de una ruta académica que servirá como orientación de las materias que se pueden tomar los semestres restantes, además de que esta recomendación fuera la más eficiente en tiempo para el caso de estudiantes que sufran un rezago en un área con el fin de que logre culminar su plan de estudios en el menor tiempo posible.

Se modeló el proceso de cómo podría un alumno interactuar con el servicio a futuro, para esto se optó por un diagrama de modelo de negocios BPMN (Business Process Model and Notation), el cual en su documentación oficial cuenta con una notación sencilla de aprender e implementar, además de que es entendible para el desarrollador como para el usuario, de acuerdo con la Object Management Group (OMG, 2013), el principal objetivo de BPMN es “proporcionar una notación que sea fácilmente entendible por todos los participantes del negocio”, lo que permite crear un lenguaje común entre las distintas partes involucradas en el desarrollo de un sistema o servicio.

En algunos problemas relacionados con el ámbito académico se contaban con problemas más grandes como el de optimización de procesos o gestión de alumnos y una base para su solución fue la implementación de diagramas BPMN. Un ejemplo es el de los investigadores Strîmbei, C., Dospinescu, O., Strainu, R. M., & Nistor, A. (2016), en su artículo mencionan la gran complejidad que presenta una universidad como organización ya que gestionan una gran red de procedimientos y procesos como por ejemplo el reclutamiento, la admisión de estudiantes, la evaluación académica y los servicios que la universidad ofrece para el bienestar de los alumnos entre otros, explican que cada proceso debe ser soportado por sistemas de información robustos y en sus hallazgos notaron que estos sistemas se desarrollan o implementan de manera fragmentada, lo que da como resultado sistemas que no se comuniquen o enlacen de manera correcta ocasionando ineficiencias y malas experiencias para el usuario. En su trabajo los autores mencionan que la base para un sistema de información exitoso no es la

tecnología que se utilice para desarrollarlo, importa más una comprensión profunda y clara de los procesos de negocios que el sistema o tecnología debe soportar, aquí es donde los investigadores proponen identificar bien los procesos y usar BPMN para modelarlos, como ejemplo muestran el proceso de intercambio de estudiantes que es implementado con BPMN.

Otro caso relacionado al tema es el de los investigadores Hedi Tebourbi, Soumaya Nouzri, Youssef Mualla, Mohammed El Fatimi, Ali Najjar, Ahmed Abbas-Turki y Mohamed Dridi (2025), recientemente analizaron una problemática relacionada a sistemas de tutoría lingüística impulsados con inteligencia artificial, encontraron que por la escasez de datos de entrenamiento al modelo de lenguaje grande se pueden dar errores, también que estos sistemas se centran más en el vocabulario o la conversación descuidando áreas importantes como la lectura, escritura y el habla, esto hace una gran confusión para el alumno y el docente.

Para dar una solución a esto Tebourbi et al. (2025) proponen una metodología que hace uso de BPMN para modelar los flujos pedagógicos, de este modo se implementan esos flujos en un sistema multi-agente y se asegura el contenido mediante una generación aumentada por recuperación, con esta implementación lograron una tutoría lingüística personalizada, modular y audita.

Este articulo refuerza las ideas de Strîmbei et al. (2016) donde mencionan que la base para un sistema de información exitoso no es la tecnología que se utilice para desarrollarlo, importa más una comprensión profunda y clara de los procesos de negocios que el sistema o tecnología debe soportares por ello que se optó por

modelar el comportamiento de un alumno de la facultad de ciencias de la computación a la hora de elegir sus materias para cargar su siguiente periodo.

Para este trabajo se utilizarán los siguientes componentes de BPMN bajo la Object Management Group (OMG, 2013) mostrados en las siguientes ilustraciones.

Figura 1

Circunferencia que indica el inicio de un evento.

Nota: *El componente marca inicio a un evento dentro del diagrama. Icono tomado de Bonita BPM, Versión 7.15.0 (Bonitasoft, 2024). Copyright 2024 por Bonitasoft.*

Figura 2

Rectángulo con icono de usuario que indica una actividad humana.

Nota: *El componente indica el trabajo que se va a realizar por una persona. Icono tomado de Bonita BPM, Versión 7.15.0 (Bonitasoft, 2024). Copyright 2024 por Bonitasoft.*

Figura 3

Rombo con el símbolo "+" que indica una compuerta lógica AND.

Compuerta AND

Nota: *El componente indica puntos donde el flujo de secuencia se divide o se une. Icono tomado de Bonita BPM, Versión 7.15.0 (Bonitasoft, 2024). Copyright 2024 por Bonitasoft.*

Figura 4

Rombo con el símbolo "x" que indica una compuerta lógica XOR.

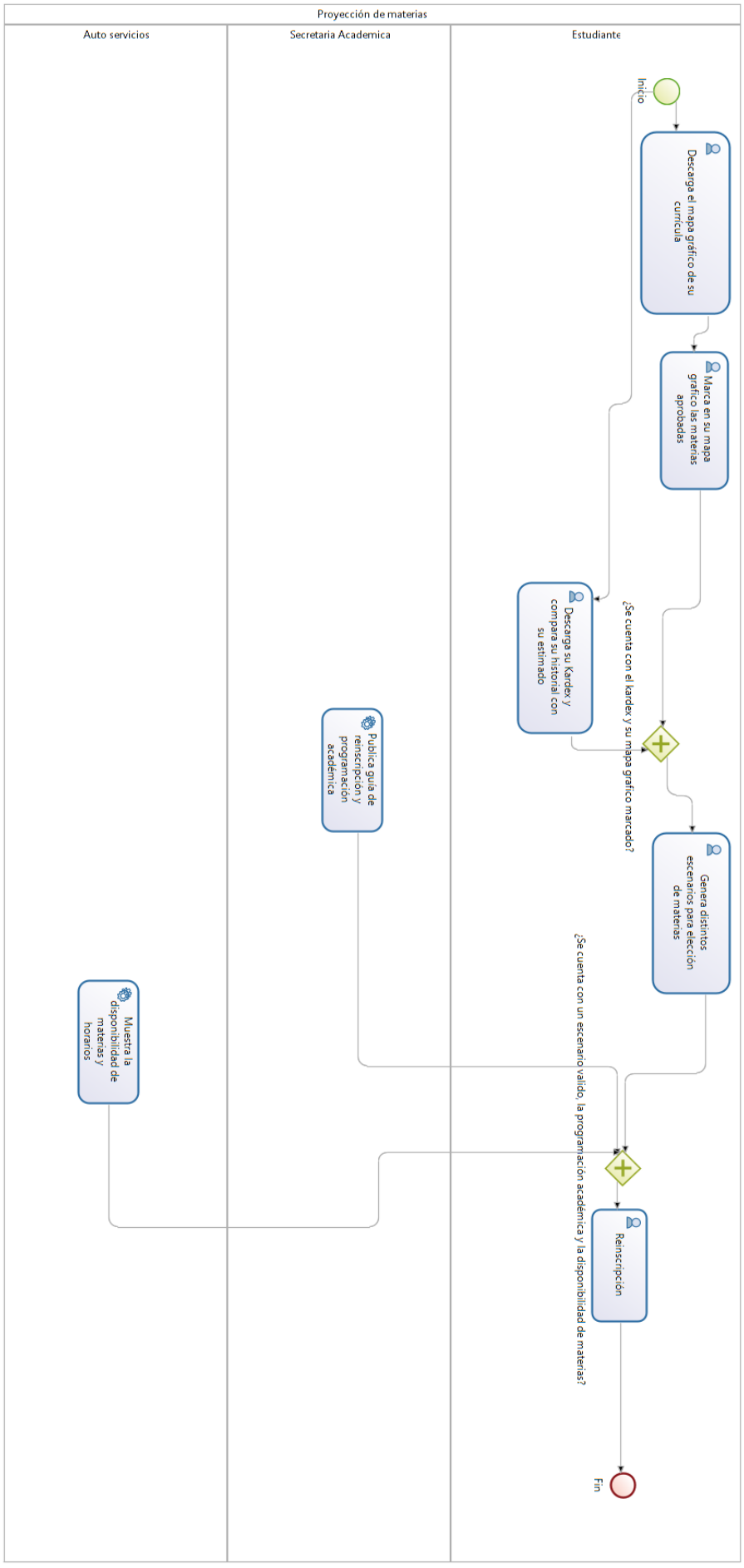
Compuerta XOR

Nota: *El componente indica puntos donde el flujo de secuencia se divide o se une. Icono tomado de Bonita BPM, Versión 7.15.0 (Bonitasoft, 2024). Copyright 2024 por Bonitasoft.*

Después de entender los procesos que sigue un alumno en su proyección de materias se llegó al diagrama de la figura 5.

Figura 5

Modelo de negocios BPMN que interpreta las acciones llevadas a cabo por un estudiante a la hora de inscribir materias para su próximo periodo.


Proyección de materias
Auto servicios
Secretaria Academica
Estudiante
Inicio
Descarga el mapa grafico de su curricula
Marca en su mapa grafico las materias aprobadas
Descarga su Kardex y compara su historial con su estimado
¿Se cuenta con el kardex y su mapa grafico marcado?
Publica guía de reinscripción y programación académica
Genera distintos escenarios para elección de materias
Muestra la disponibilidad de materias y horarios
¿Se cuenta con un escenario valido, la programación académica y la disponibilidad de materias?
Reinscripción
Fin

Nota: *Diagrama realizado en Bonita BPM, Versión 7.15.0 (Bonitasoft, 2024). Copyright 2024 por Bonitasoft.*

Con este modelo se vuelve más fácil una futura automatización con un sistema o dar alguna herramienta para facilitar el proceso de cada estudiante.

Requerimientos

En este capítulo se sentarán los requerimientos para la realización de un sistema que facilite a los estudiantes una elección de materias conforme a sus materias previamente aprobadas, esto conforme a la norma **IEEE Std 830-1998** la cual establece la estructura y contenido de una especificación de requerimientos de software, este estándar propone descripción formal de la funcionalidad de un sistema, con el objetivo de garantizar su veracidad, consistencia y trazabilidad (IEEE Computer Society, 1998).

Requerimientos generales

Propósito del sistema

Implementación de un algoritmo que permita orientar a los estudiantes de nivel superior para elegir una ruta académica adecuada por la cual puedan transitar de acuerdo con las condiciones personales que viven en cada periodo de reinscripción.

Alcance del sistema

Se espera que la herramienta permita consultar el avance académico del estudiante, genere una ruta académica personalizada y a su vez permita mostrar el tiempo en años que le tomara terminar su mapa curricular y el tiempo total

desde que ingreso hasta el último periodo recomendado por el sistema. Con estas funcionalidades se espera que el alcance llegue a cualquier usuario de la Facultad de Ciencias de la Computación de la BUAP siempre y cuando su carrera esté disponible.

Descripción general

Bajo la norma IEEE Std 830-1998 se da el contexto del proyecto que se quiere abordar para entender la funcionalidad, las restricciones, los usuarios previstos y las dependencias del proyecto que requerirá en una futura implementación (IEEE Computer Society, 1998).

Perspectiva del producto

La idea de implementar una herramienta la cual genere una ruta académica personalizada a los estudiantes con base en su programa educativo así como las materias que ha acreditado en el tránsito por el mismo surge en primera instancia por la necesidad de una herramienta de apoyo para los estudiantes de la Facultad de Ciencias de la Computación, a un futuro este proyecto ayudaría a estudiantes de cualquier carrera siempre que su carrera este registrada, además de que este sistema serviría para llevar un mejor control de los estudiantes.

Se prevé que el sistema este conformado de 3 componentes:

Interfaz de usuario: Para una mejor portabilidad se implementará en web considerando un diseño limpio y utilizar lenguajes que permitan la conexión con el algoritmo.

Servidor de aplicación: Consiste en que lenguaje se programará el autómata finito y además se encargue de todas las validaciones que se reciban del usuario (backend), para este componente se hará uso de Python.

Base de datos: Se usará MySQL por la facilidad de instalación, conexión y administración, aquí se almacenarán los datos del estudiante, su estado, ruta académica entre otros.

Funciones del producto

1.- *Gestión de usuarios:* Se lleva a cabo el registro, inicio de sesión y actualización de datos del estudiante.

2.- *Consulta académica:* Se muestra el listado de materias por carrera, las aprobadas y las disponibles por cursar siempre y cuando los prerrequisitos estén acreditados.

3.- *Generación de ruta académica:* se genera la recomendación del algoritmo considerando la elección de materias aprobadas por parte del usuario.

Características del usuario

1.- *Estudiante:* es el usuario principal, el que enriquecerá al sistema ya que tendrá las funciones de registrar sus datos como materias aprobadas, carrera entre otros. Además de que su información es indispensable para el funcionamiento del algoritmo.

Restricciones

1.- El sistema requerirá conexión a internet ya que los datos serán guardados en una base de datos, además de que el algoritmo también debe estar alojado en un servidor.

2.- Solo se podrá acceder con credenciales validas, se tiene pensado un módulo de registro por lo que un usuario debe registrar una matrícula no mayor al año actual.

3.- La información de materias y sus prerrequisitos por cada carrera debe estar actualizada y esta se debe hacer manualmente en la base de datos.

4.- La interfaz gráfica debe ser simple, sin complicar el uso para el usuario.

5.- Se deben proteger las credenciales de los usuarios.

Suposiciones y dependencias

Para un correcto funcionamiento del sistema se requiere información del departamento de secretaria académica, infraestructura, conocer la diversidad de dispositivos con los que cuentan los estudiantes de la facultad entre otros, se analizaron los más básicos y son los siguientes:

1.- Los datos de que materias tiene una carrera y como es su orden y jerarquía, esto para que el algoritmo pueda trazar una ruta valida.

2.- Se piensa que los estudiantes tienen una conexión a internet estable a la hora de querer utilizar la herramienta.

3.- Se contará con un servidor capaz de soportar la base de datos y la ejecución del algoritmo (es recomendable tener 1 servidor para cada proceso).

4.- El estudiante deberá marcar las materias verdaderamente acreditadas, de no ser así la recomendación será errónea y no cumplirá el propósito de ayudar al alumno.

Algunas suposiciones a futuro que impulsarían aún mejor el uso de este proyecto son las siguientes:

1.- Se podría evitar el módulo de registro si es que se cuenta con la información oficial de los estudiantes (nombre, matricula, carrera), además de que se evitan solicitudes falsas y una mejor protección. Esto depende de la facultad o de la universidad debido a los datos sensibles.

2.- La implementación de una interfaz administrativa para el tutor y la secretaria académica con el fin de tener un mejor control en cada estudiante.

Requisitos específicos

Requerimientos funcionales

Conforme a la norma IEEE Std 830-1998, los requerimientos funcionales deben describir como se comportará el proyecto a realizar de manera que el cliente o los usuarios entienda, para los requerimientos funcionales se describirá el comportamiento del sistema bajo condiciones normales, alternas y excepcionales (IEEE Computer Society, 1998), en cada caso de uso será dividido en un flujo principal, flujo alterno y flujo excepcional.

Caso de uso CU-01: Inicio de sesión

Actor principal: Estudiante.

Descripción: La sección permite al usuario autenticarse.

Precondición: El usuario debe ingresar las credenciales correctas.

Postcondición: El sistema muestra sus materias conforme a su carrera para que pueda actualizarlas en el CU-04.

Flujo básico

FB1.- El sistema valida las credenciales ingresadas con los registros en la base de datos.

FB2.- Si los datos son correctos se hace el inicio de sesión mostrando la página principal, si no, se dirige a excepción E2.

Flujo alterno

FA1.- El usuario trata de intentar a una URL que requiere autenticación, el sistema lo redirige a la URL original ósea a FB1.

Flujos de excepción

FE1.- Credenciales invalidas: El sistema muestra un error de autenticación y se regresa a FB1 para intentar nuevamente.

FE2.- Fallo de base de datos: Si existe el caso en que no responda la base de datos se notificara con un error personalizado que hay indisponibilidad hasta arreglar el error

Caso de uso CU-02: Registro de usuario

Actor principal: Estudiante

Descripción: La sección permite al usuario generar sus credenciales.

Precondición: El usuario no debe existir y debe ingresar datos reales.

Postcondición: El sistema muestra una sección donde el usuario debe elegir su carrera.

Flujo básico

FB1.- El usuario deberá ingresar una matrícula de 9 dígitos empezando con un año menor al actual y su contraseña.

FB2.- El sistema valida que la matricula cumpla con la longitud de la matricula y el año, además de que la contraseña debe ser mayor a 8 caracteres, si no, se manda a E1

FB3.- El sistema almacena y deja la sesión iniciada para que el estudiante pueda elegir su carrera en CU-03, en dado caso de que la matricula ya se haya registrado previamente se manda a FE2.

Flujos de excepción

FE1.- Datos inválidos: Sea cualquier caso de los flujos alternos se mostrará un mensaje de error y el usuario debe volver a FB1.

FE2.- Matricula ya registrada: El sistema muestra un mensaje de matrícula duplicada y no se sobrescribirá la información por lo que el usuario deberá volver a FB1.

FE3.- Fallo de base de datos: Si existe el caso en que no responda la base de datos se notificara con un error personalizado que hay indisponibilidad hasta arreglar el error

Caso de uso CU-03: Seleccionar carrera

Actor principal: Estudiante

Descripción: La sección permite seleccionar la carrera a la que pertenece el estudiante.

Precondición: El usuario debe tener su sesión iniciada.

Postcondición: El sistema guarda la carrera elegida y asigna las materias que le corresponden.

Flujo básico

FB1.- EL usuario debe desplegar el listado y seleccionar la carrera correspondiente.

FB2.- El sistema guarda su carrera seleccionada y asigna las materias conforme a la carrera al alumno, después se redirige a CU-04.

Flujo alterno

FA1.- Si el alumno selecciona otra carrera a la elegida por primera vez se guardan ambos historiales.

Flujos de excepción

FE2.- Fallo de base de datos: Si existe el caso en que no responda la base de datos se notificara con un error personalizado que hay indisponibilidad hasta arreglar el error

Caso de uso CU-04: Marcar materias acreditadas

Actor principal: Estudiante

Descripción: Se permite marcar y desmarcar las materias que el usuario haya acreditado.

Precondición: El usuario debe tener su sesión iniciada y una carrera seleccionada.

Postcondición: Si la selección es correcta se redirige a CU-05.

Flujo básico

FB1.- El estudiante marca o desmarca materias de la lista mostrada por el sistema con base a su carrera, si el alumno ya había capturado sus materias anteriormente puede recurrir a los flujos FA1 y FA3.

FB2.- El sistema valida los prerrequisitos de las materias seleccionadas.

FB3.- Si hay una o más materias seleccionadas sin su prerrequisito aprobado se redirige a E1.

FB4.-Si los prerrequisitos son válidos para tomar las nuevas materias se actualiza el historial del alumno y se hace un respaldo de cambios en CU-06.

FB5.- El sistema confirma los cambios y redirige a CU-05.

Flujo alterno

FA1.- Si el alumno no guarda cambios y solo quiere pasar a la siguiente sección el sistema informa que no hubo cambios y se permite el acceso a FB1 del CU-05.

FA2.- Si es la primera captura generada por un usuario nuevo en el historial aparecerá el registro de cuando se hizo esta captura inicial, después se pasa a FB5.

FA3.- Si el usuario se equivocó de carrera tiene la opción de cambiar de carrera y el sistema mostrara las materias actualizadas conforme a su nueva elección dejándolo en FB1.

Flujos de excepción

FE1.- No se cumplen los prerrequisitos: Si el alumno tiene una o más materias marcadas, pero sin haber marcado las asignaturas que desbloquean las nuevas se notifica al usuario con un mensaje que materias no se pueden guardar, no se guardan los cambios y debe volver a capturar nuevamente su historial en FB1.

FE2.- Fallo de base de datos: Si existe el caso en que no responda la base de datos se notificara con un error personalizado que hay indisponibilidad hasta arreglar el error

Caso de uso CU-05: Visualización de materias aprobadas y captura de periodos.

Actor principal: Estudiante

Descripción: Muestra el historial de materias previamente guardadas y un formulario donde debe desplegar el periodo en el que entro, el periodo actual y seleccionar si inscribirá materias en verano o no para la generación de su ruta académica recomendada.

Precondición: El usuario debe tener su sesión iniciada, una carrera seleccionada y su historial que debió pasar los prerrequisitos (el proceso de CU-04).

Postcondición: Si todos los campos necesarios fueron llenados se pasa a CU-06.

Flujo básico

FB1.- El estudiante visualiza sus materias aprobadas y faltantes por cursar.

FB2.- El sistema solicita al usuario seleccionar el periodo de ingreso, el periodo actual y seleccionar si cursara materias en verano o no, si no se contesta el mini formulario se redirige a E1.

FB3.-Si todos los campos fueron correctos se pasa a CU-06.

Flujo alterno

FA1.- Si el usuario se equivocó en la captura de materias, pero si se le concedió el acceso a esta sección puede regresar a CU-04 y corregir su selección.

FA2.- Si el usuario solo quería guardar su historial en CU-04 y no requiere visualizar su plan académico recomendado puede cerrar su sesión.

Flujos de excepción

FE1.- No selecciono uno o los dos periodos (ingreso y actual): El sistema le notifica que es necesario capturar los periodos de inicio y actual debido a que serán necesarios para el cálculo de tiempo en CU-06, no se puede redirigir al siguiente modulo y debe seleccionar las opciones en FB2.

FE2.- Fallo de base de datos: Si existe el caso en que no responda la base de datos se notificara con un error personalizado que hay indisponibilidad hasta arreglar el error

Caso de uso CU-06: Generación de ruta académica por el sistema.

Actor principal: Sistema

Descripción: Muestra la ruta académica recomendada por el algoritmo considerando su previo registro de materias aprobadas, además se muestra el cálculo de tiempo restante que le tomara al usuario en terminar su plan académico y el tiempo total desde que inicio hasta el último periodo que el algoritmo recomienda.

Precondición: El usuario debe tener su sesión iniciada, una carrera seleccionada, su historial que debió pasar los prerrequisitos (el proceso de CU-04) y sus periodos seleccionados en CU-05.

Postcondición: El sistema muestra en una tabla la recomendación de materias a cursar por periodo y año, además de que se muestra el cálculo de tiempo faltante y tiempo total desde inscripción hasta el último periodo recomendado.

Flujo básico

FB1.- El estudiante visualiza su recomendación generada.

FB2.- Si la recomendación es aceptada por el usuario la guarda y puede cerrar sesión y finalizar el uso del sistema, si no puede elegir FA1.

Flujo alterno

FA1.- Si el usuario siente que puede tener una mejor respuesta puede regresar a FB1 de CU-05.

FA2.- Si el año del servidor es anterior al actual se pasa a E1.

Flujos de excepción

FE1.- Año del servidor incorrecto: En caso de que el año del servidor donde está alojado el algoritmo es mayor al actual el cálculo de tiempo será incorrecto, si se

generara la recomendación académica, pero con tiempos incorrectos, si el año es menor al actual al hacer cálculos dará un tiempo negativo y no se pueden mostrar años negativos en una recomendación, por lo tanto, aparecerá una alerta al usuario por inconvenientes y tendrá que intentar generar su ruta más tarde.

FE2.- Fallo de base de datos: Si existe el caso en que no responda la base de datos se notificara con un error personalizado que hay indisponibilidad hasta arreglar el error.

Requerimientos no funcionales

A continuación, se muestra una tabla de los requerimientos no funcionales.

ID	Descripción	Tipo
RNF-01	El sistema debe responder a las solicitudes del usuario en menos de 2 segundos.	Rendimiento
RNF-02	Las contraseñas deben almacenarse de forma cifrada utilizando algoritmos de hash seguros.	Seguridad
RNF-03	La interfaz debe ser accesible desde dispositivos móviles y navegadores modernos.	Usabilidad

Capítulo 3: Análisis y diseño

En este capítulo se presenta el análisis y diseño del sistema a implementar con base a los requerimientos descritos del capítulo anterior. Se muestra cómo se

emplean técnicas de modelado orientado a objetos, arquitectura software y diagramación UML para definir la estructura lógica y técnica del sistema.

Metodología de desarrollo

Al ser un proyecto de programación se optó por la metodología RUP, la cual para este proyecto y en base a International Business Machines Corporation (IBM) se divide en 4 etapas que se desglosan de la siguiente manera:

1.- Fase de iniciación (Inception)

En este primer punto se verá la definición clara del proyecto, su alcance y su viabilidad, también se analizarán los riesgos iniciales que podrá llegar a tener este proyecto.

2.- Fase de elaboración (Elaboration)

Esta es una fase que toma más tiempo ya que se analizan puntos como el refinamiento de requisitos y casos de uso, el diseño de la aplicación, así como considerar el lenguaje que se ocupara para la interfaz, base de datos y el autómata finito. La metodología menciona el uso de iteraciones y estrategias de prueba para detallar cuantas veces se repetirá una sub-fase y que funcionalidad entregará cada una de ella.

3.- Fase de construcción (Construction)

En la construcción se realizan las iteraciones propuestas en la fase anterior, siendo esta la fase crítica ya que se hacen modificaciones al proyecto n veces necesarias hasta que se logre una aprobación o funcionamiento deseado.

4.- Fase de transición (Transition)

Esta fase es dedicada a la entrega del producto final y también a la estabilización o mantenimiento del sistema.

¿Por qué usar RUP para este proyecto?
RUP fue seleccionado como metodología base debido a su enfoque disciplinado y la capacidad de gestionar proyectos de tamaño significativo, proporcionando una estructura formal para la fase de Análisis y Diseño como lo plantean Jacobson, Booch y Rumbaugh (1999).

Diagramas

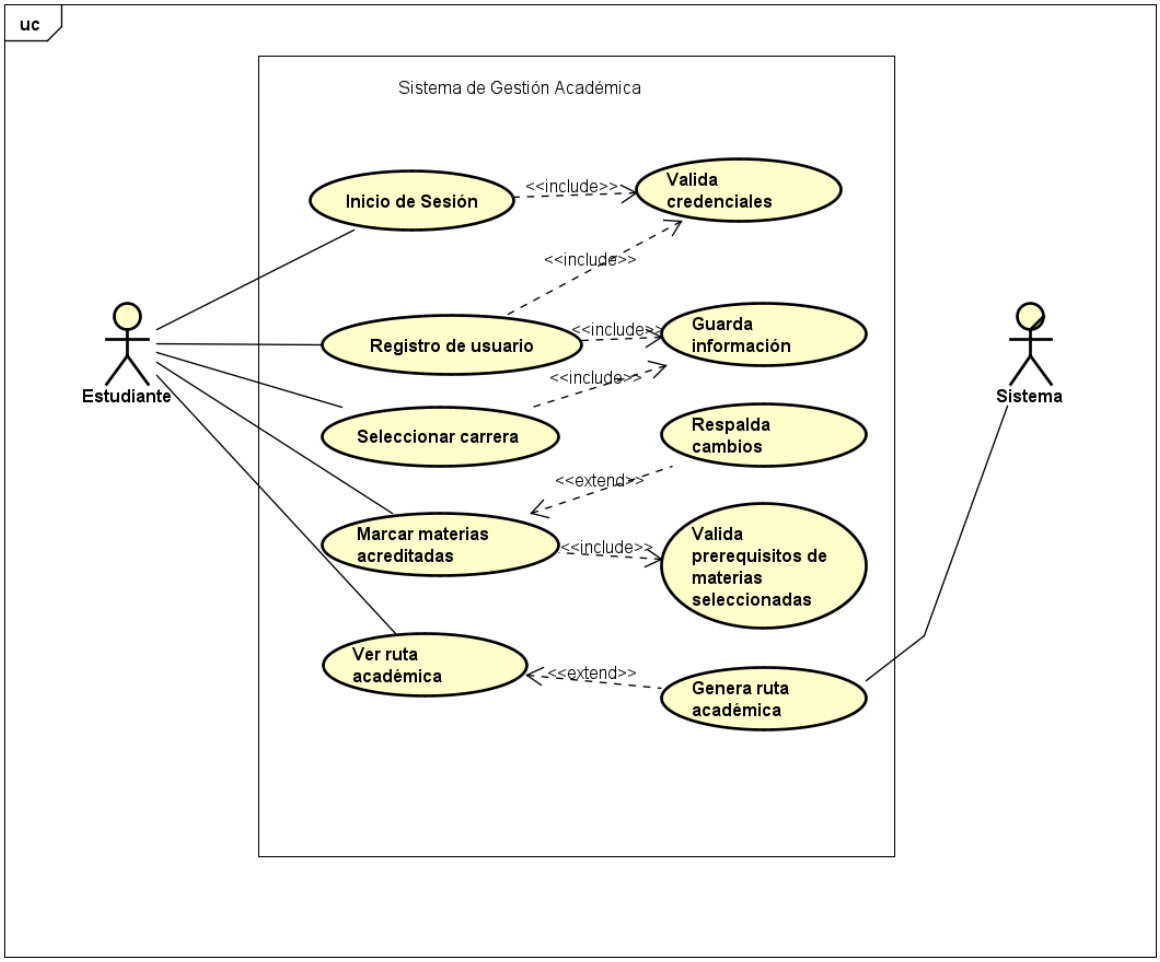
Para una programación limpia es necesario diseñar diagramas los cuales refleje el comportamiento del proyecto. En esta sección se plasma un diagrama de caso de uso general el cual describe las acciones que realiza el usuario con el sistema, también por cada caso de uso planteado en los requerimientos tendrá su diagrama de secuencia y por último un diagrama de eventos.

Diagrama de casos de uso

En la figura 7 se muestra un diagrama el cual representa las acciones que puede realizar un usuario (estudiante) en el sistema, además de que se puede ver como los procesos ofertados por el sistema se relacionan con las funciones que puede hacer el estudiante.

Figura 7

Diagrama general de casos de uso del sistema.



Nota: *La figura describe el papel que tiene el usuario con el sistema. Elaboración propia realizada con Astah (Change Vision Inc., versión 8.5.0).*

En el diagrama se aprecia los módulos que puede acceder un usuario, los cuales son el inicio de sesión, registro de usuario, selección de carrera, marcar materias acreditadas y ver su ruta académica recomendada así mismo se ven las relaciones que tienen estas acciones con los servicios más importantes del sistema.

Diagramas de secuencia

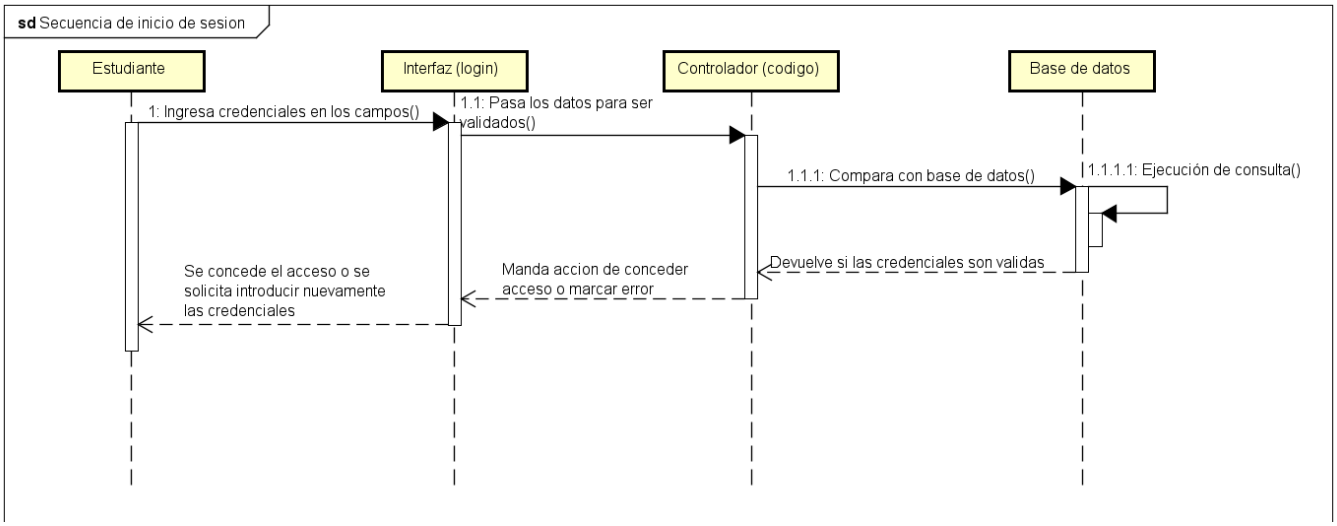
Con un diagrama general es necesario describir cada fase con diagramas de secuencia, de este modo se logra ver cuánto tiempo podría tardar un usuario en cada módulo, a continuación, se presenta un diagrama por cada caso de uso descrito en los requerimientos.

Secuencia de inicio de sesión

Para este módulo el usuario deberá introducir sus datos (matricula y contraseña), si son válidas se le concede el acceso, pero en caso contrario el sistema le marcara una alerta diciendo que el usuario es incorrecto y debe intentarlo de nuevo, en la figura 8 se muestra más a detalle los pasos donde también se muestran los pasos detrás de esta acción como es la comparación de las credenciales insertadas con la base de datos y como el código personaliza un mensaje de inicio de sesión correcto.

Figura 8

Diagrama de secuencia para el módulo de inicio de sesión.



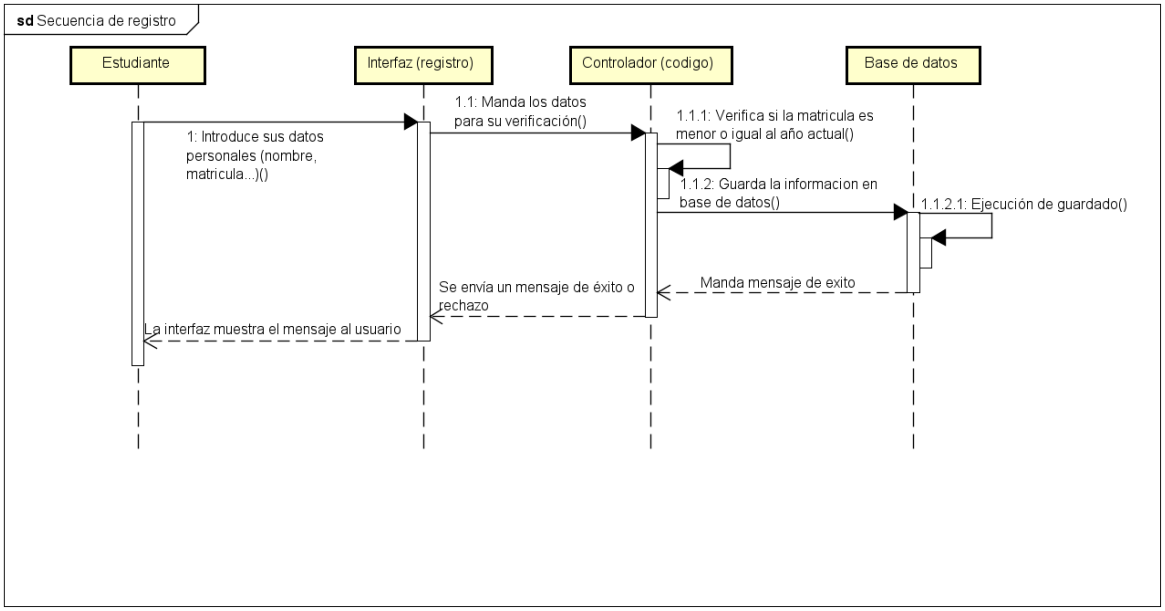
Nota: *La figura describe los pasos a seguir para iniciar sesión. Elaboración propia realizada con Astah (Change Vision Inc., versión 8.5.0).*

Secuencia de registro

En la sección de registro es parecida a la de inicio de sesión ya que el usuario debe introducir sus datos personales para la creación de su cuenta en el sistema, sin embargo en la figura 9 se puede apreciar como el controlador es el que demora un poco más de tiempo en completar el proceso ya que debe de validar que la matricula del alumno tenga 9 caracteres y sea menor o igual al año actual, esto debido a que es un requerimiento fundamental para que a la hora de determinar el tiempo total transcurrido en la carrera no de números incorrectos.

Figura 9

Diagrama de secuencia para el módulo de registro.



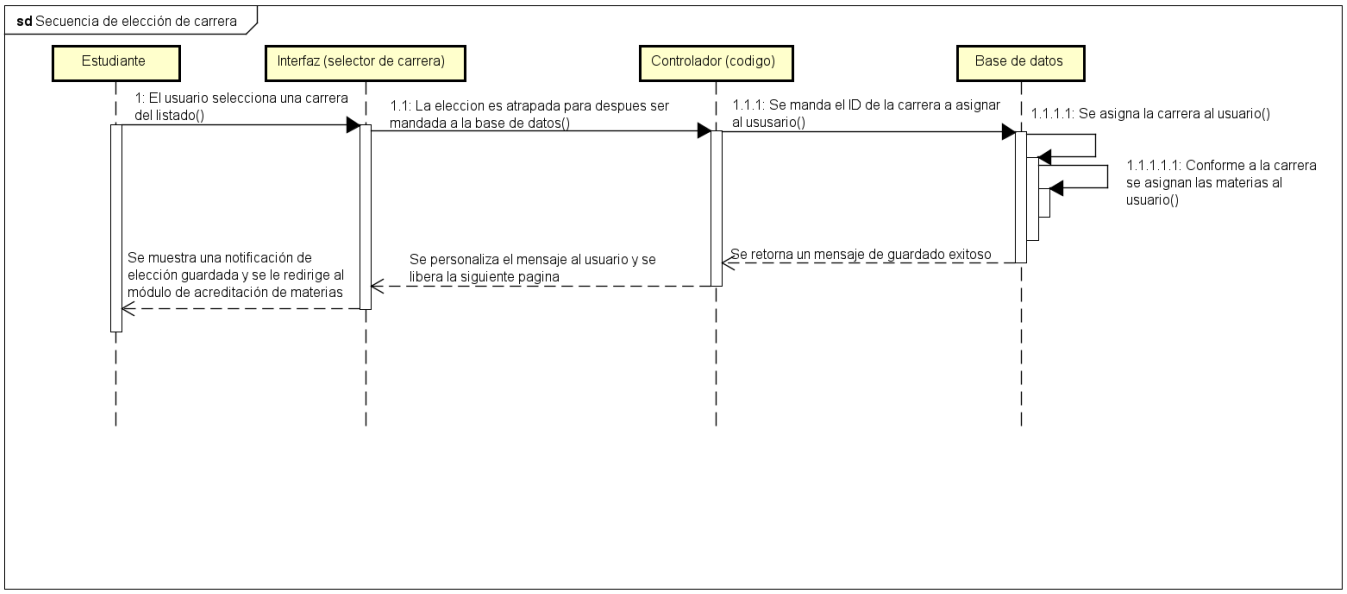
Nota: *La figura describe los pasos a seguir para registrarse. Elaboración propia realizada con Astah (Change Vision Inc., versión 8.5.0).*

Secuencia de selección de carrera

Este módulo es el más sencillo para el usuario ya que solo debe seleccionar del listado disponible la carrera la cual quiera generar una ruta académica conforme a sus avances, aunque si se aprecia en la siguiente figura como la base de datos tiende a ocupar más tiempo ya que debe completar 2 procesos a la vez, primero debe guardar la selección de carrera del estudiante y después debe asignar las materias correspondientes al alumno.

Figura 10

Diagrama de secuencia para la selección de carrera.



Nota: *La figura describe los pasos a seguir para seleccionar y guardar una carrera.*

Elaboración propia realizada con Astah (Change Vision Inc., versión 8.5.0).

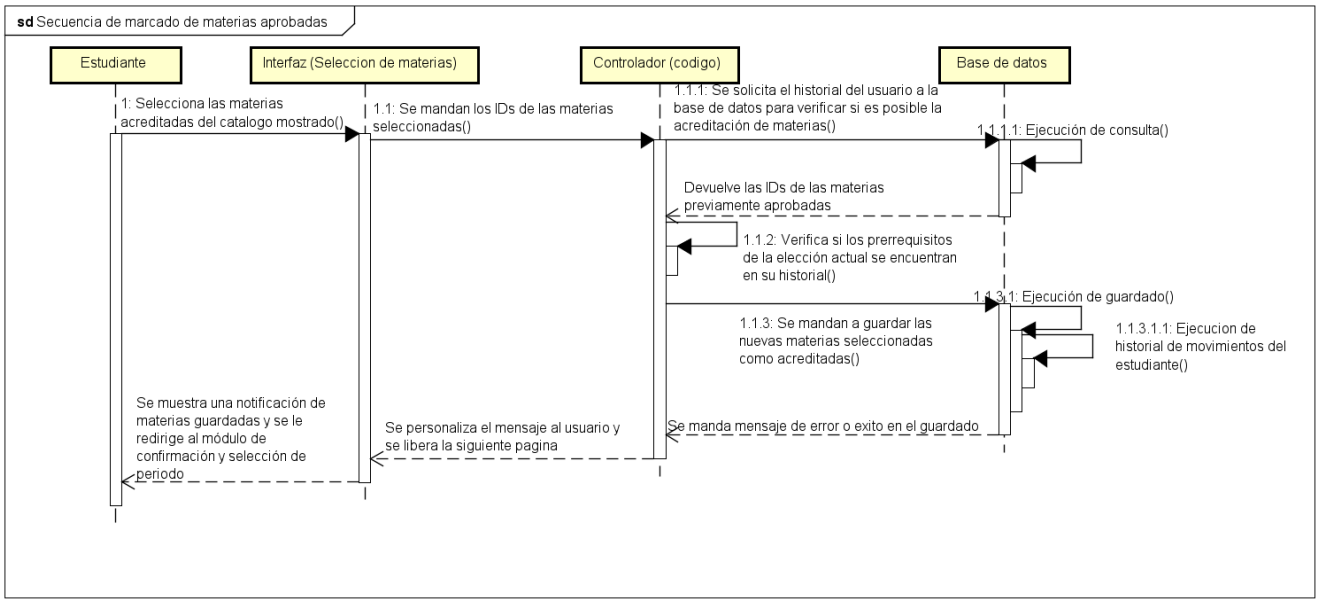
Secuencia de marcado de materias aprobadas

Para esta sección el alumno debe marcar en una lista las materias que haya aprobado para guardar su nuevo historial, en la figura 10 se aprecia como la carga

de tiempo la tienen el controlador y la base de datos ya que en el momento que el usuario manda su selección de materias el código debe solicitar su último historial guardado y comparar los prerrequisitos de las nuevas materias seleccionadas, si los prerrequisitos se encuentran aprobados en el anterior historial la nueva selección se guarda y se hace un respaldo de movimientos en una tabla independiente, esto para llevar control, en caso contrario se notifica al estudiante que no es posible guardar la selección porque hay conflicto en ciertas materias.

Figura 10

Diagrama de secuencia para la actualización de materias aprobadas.



Nota: *La figura describe los pasos a seguir para actualizar las materias acreditadas del usuario. Elaboración propia realizada con Astah (Change Vision Inc., versión 8.5.0).*

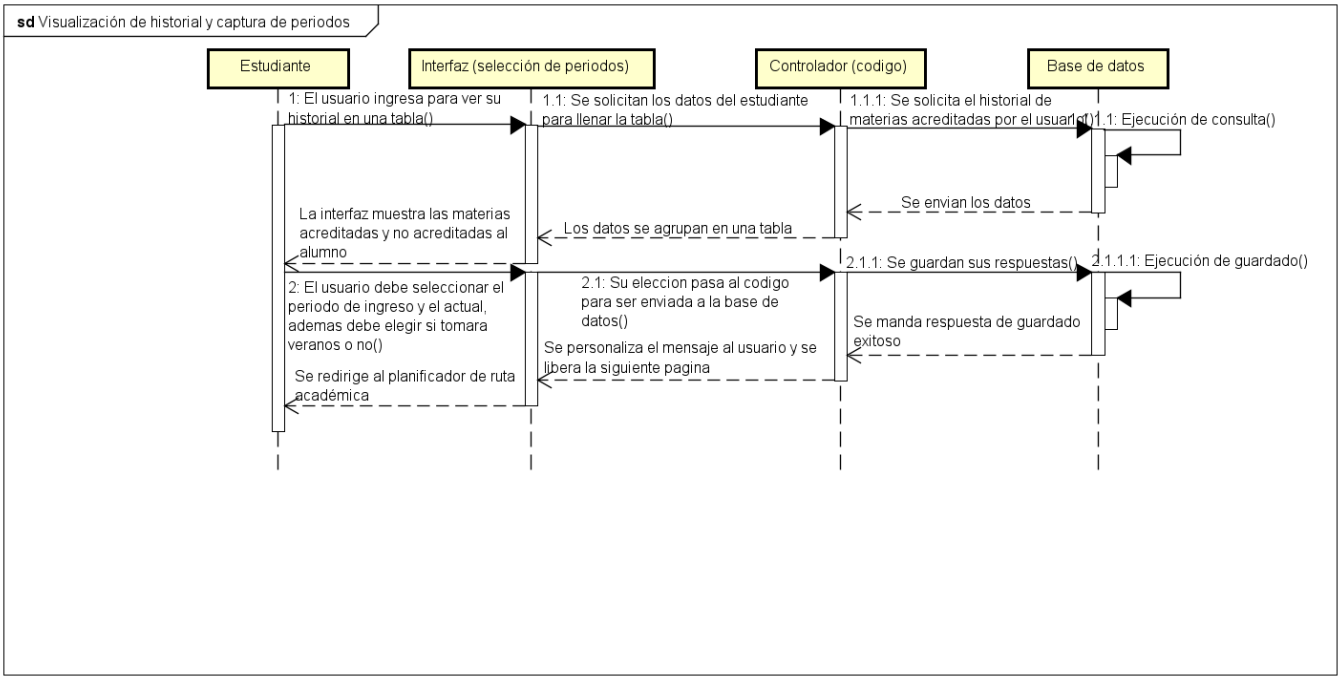
Secuencia de visualización de materias aprobadas y captura de periodos

Aquí al igual que en la elección de carrera esta interfaz es sencilla para el estudiante ya que solo debe ver si su historial de materias es correcto conforme a

lo seleccionado en el módulo anterior y debe seleccionar los periodos de ingreso a la carrera y periodo actual, además de que debe seleccionar si para la generación de su recomendación académica incluirá el periodo de verano o no. En la figura 11 se puede ver cómo los 4 actores (estudiante, interfaz, código y base de datos) tienen casi el mismo número de eventos ya que en esta sección solo es de guardar y mostrar información.

Figura 11

Diagrama de secuencia para la visualización del historial guardado y captura de periodos.



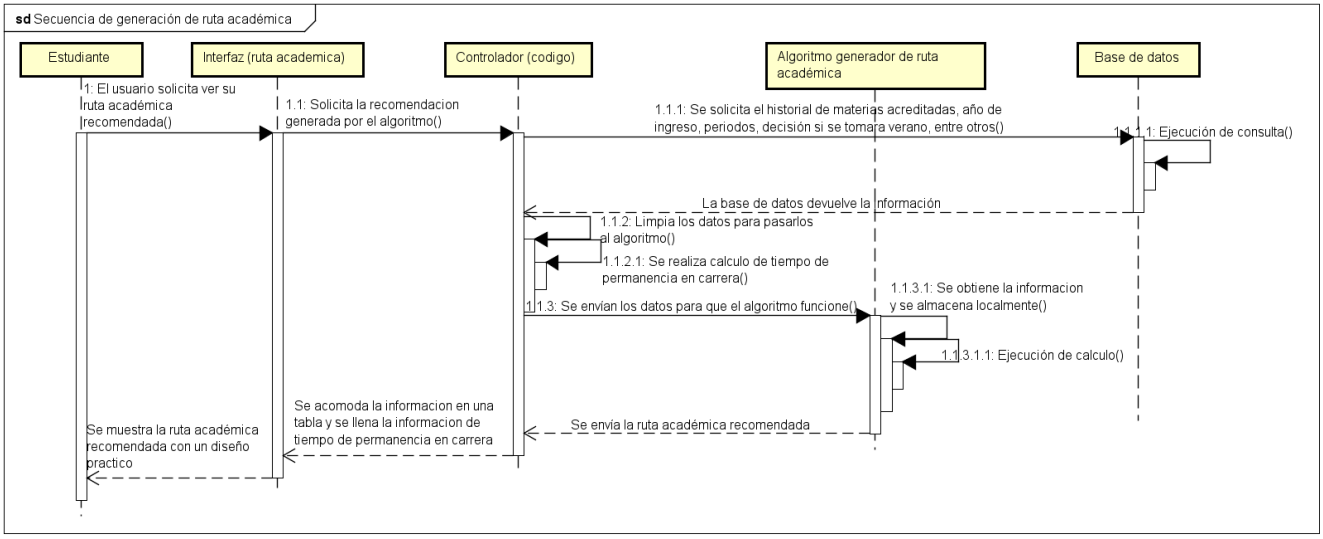
Nota: *La figura describe los pasos a seguir para visualizar las materias acreditadas y para la captura de periodos. Elaboración propia realizada con Astah (Change Vision Inc., versión 8.5.0).*

Secuencia de generación de ruta académica recomendada

Por ultimo en la figura 12 se muestra la interacción del usuario con el módulo principal el cual es la generación de ruta académica, para este módulo interviene el actor algoritmo ya que es el que tiene más procesos a cumplir y por ende es mayor su tiempo de ejecución debido a que debe recopilar la información del estudiante con ayuda del controlador, este hace la consulta a la base de datos y pasa los datos al algoritmo, después analiza el mejor camino a elegir y envía la recomendación al controlador junto con el cálculo de tiempo restante en terminar la carrera y tiempo total de permanencia, por último el controlador se encarga de mostrar los datos al usuario de manera entendible y simple.

Figura 12

Diagrama de secuencia para la generación y visualización de la ruta académica.



Nota: *La figura describe los pasos a seguir visualizar la ruta académica recomendada.*

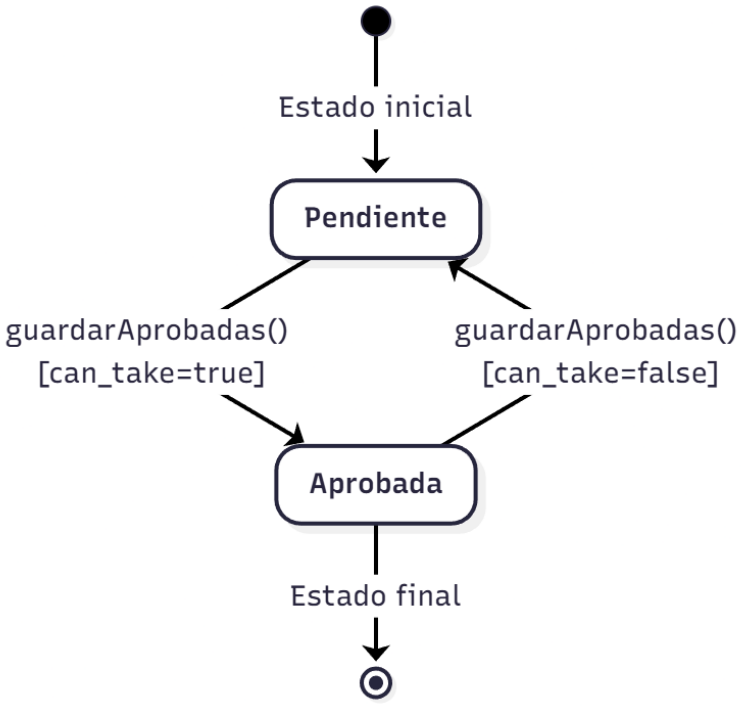
Elaboración propia realizada con Astah (Change Vision Inc., versión 8.5.0).

Diagrama de estado

Para este diagrama el proceso más importante es el estatus de una materia ya que dependiendo de su estado es el resultado que dará para guardar o no su historial.

Figura 13

Diagrama de estado.



Nota: *La figura describe los estados que cuenta una materia. Elaboración propia.*

Diseño de interfaces

Ya con una visión más clara del comportamiento que tendrá el usuario con el sistema se procede a realizar los diseños de interfaz, para esta sección se

muestran los bocetos planteados para su futura programación, cada interfaz se diseñó de manera que el estudiante tenga que realizar el menor número de acciones posible para que no se le dificulte la interacción con el sistema.

Diseño de login

El primer interfaz que se presentara al ingresar al sistema es el login, ya que un usuario siempre deberá estar iniciada su sesión para acceder a las demás funciones del sistema como se menciona en los casos de uso, para esta pantalla se realizó un diseño simple ya que es lo más convencional, solicita el usuario (matricula) y la contraseña, los componentes estarán centrados y se hará uso del framework Bootstrap para darle estilo y hacerlo responsivo.

Figura 14

Interfaz de inicio de sesión.

Nota: *La figura muestra el diseño del login. Elaboración propia.*

Diseño de módulo de registro

Como se planteó en los casos de uso si una persona no tiene una cuenta tendrá la opción de crear una con matrícula y contraseña, el nombre no se pensó como requisito puesto que en los alcances a futuro se espera que la página cuente con la base de datos de todos los alumnos en la facultad, de este modo con la matricula se identifica el nombre del usuario que está ingresando.

Figura 15

Interfaz de registro para nuevos usuarios.

Nota: *La figura muestra el diseño del registro. Elaboración propia.*

Diseño del módulo de selección de carrera

Esta sección es la más sencilla ya que cumple con lo que se plantea en los casos de uso, habrá un desplegable el cual contiene las carreras a elegir para el usuario, solo se debe seleccionar una y guardarla haciendo click al botón.

Figura 16

Interfaz de selección y guardado de carrera.

Nota: *La figura muestra el diseño del apartado donde se selecciona la carrera.*

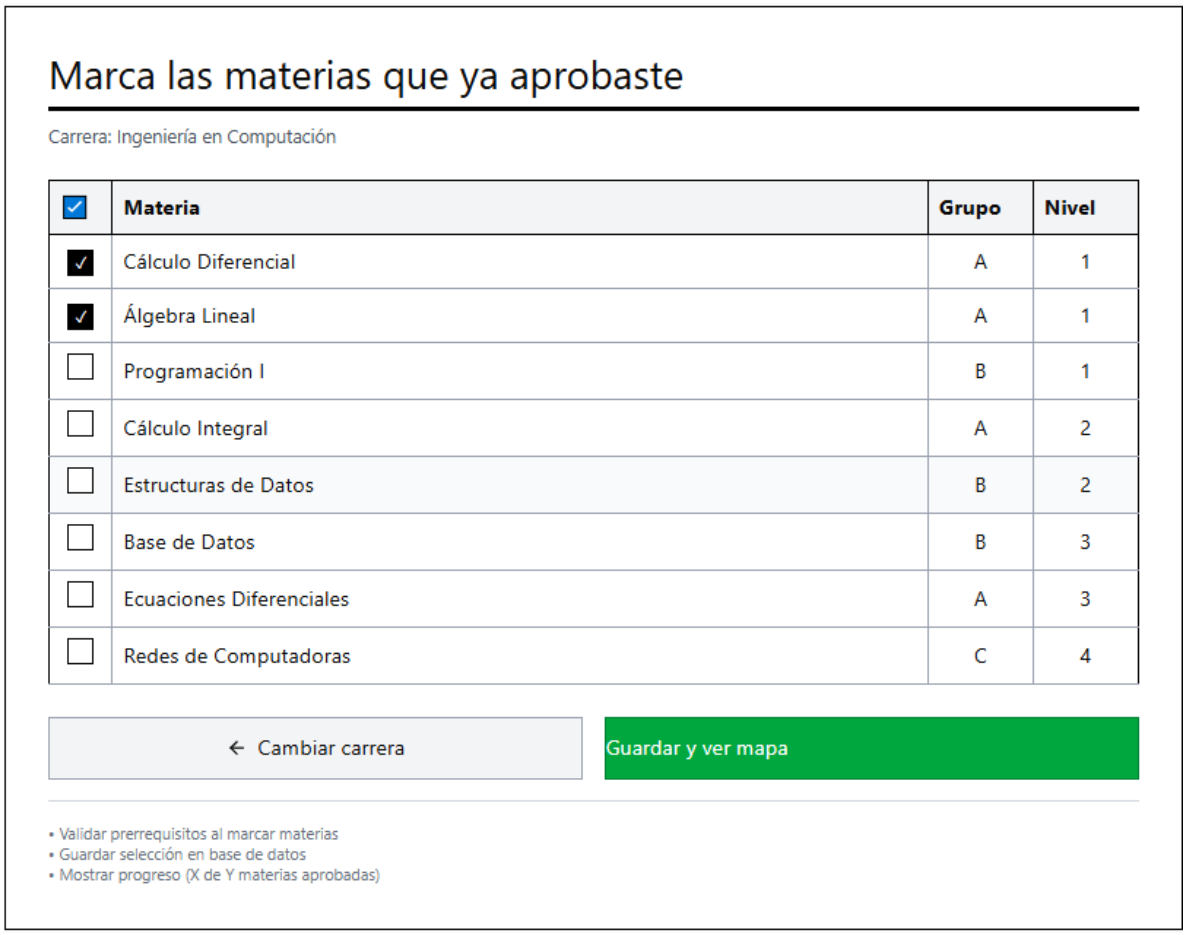
Elaboración propia.

Diseño del módulo de acreditación de materias

En los diagramas de secuencia se mostró que esta sección es la más pesada e importante ya que el usuario debe ingresar correctamente sus materias aprobadas para el funcionamiento del algoritmo, además de que este apartado cuenta con más procesos de validación debido a que la materia a guardar debe de tener su prerrequisito aprobado, de lo contrario al generar la ruta académica ocasionaría un error tanto para el usuario como para el servicio. Dado este problema la mejor solución fue colocar las materias en una tabla la cual tuviera unos checklist con los que el alumno marcario la materia a acreditar, también se muestran datos de esa materia como el nivel de semestre en el que se encuentra y el grupo al que pertenece.

Figura 17

Interfaz de selección de materias.



Nota: *La figura muestra el diseño del apartado donde se seleccionan las materias aprobadas. Elaboración propia*

Diseño del módulo de captura de periodo

Para el módulo previo a la generación de la ruta académica es necesario la captura de los periodos (inscripción y actual) del estudiante, sin embargo el dejar esas únicas funcionalidades verían muy simple a esa sección y haría ver feo al sistema por lo que se implementó la visualización de las materias aprobadas y faltantes de acreditar en una tabla, este elemento sirve al estudiante verificar que sus materias acreditadas estén correctas, si es el caso contrario la página tiene un enlace que redirecciona al módulo anterior para hacer correcciones.

Figura 18

Interfaz de selección de verificación de guardado de historial y captura de periodos.

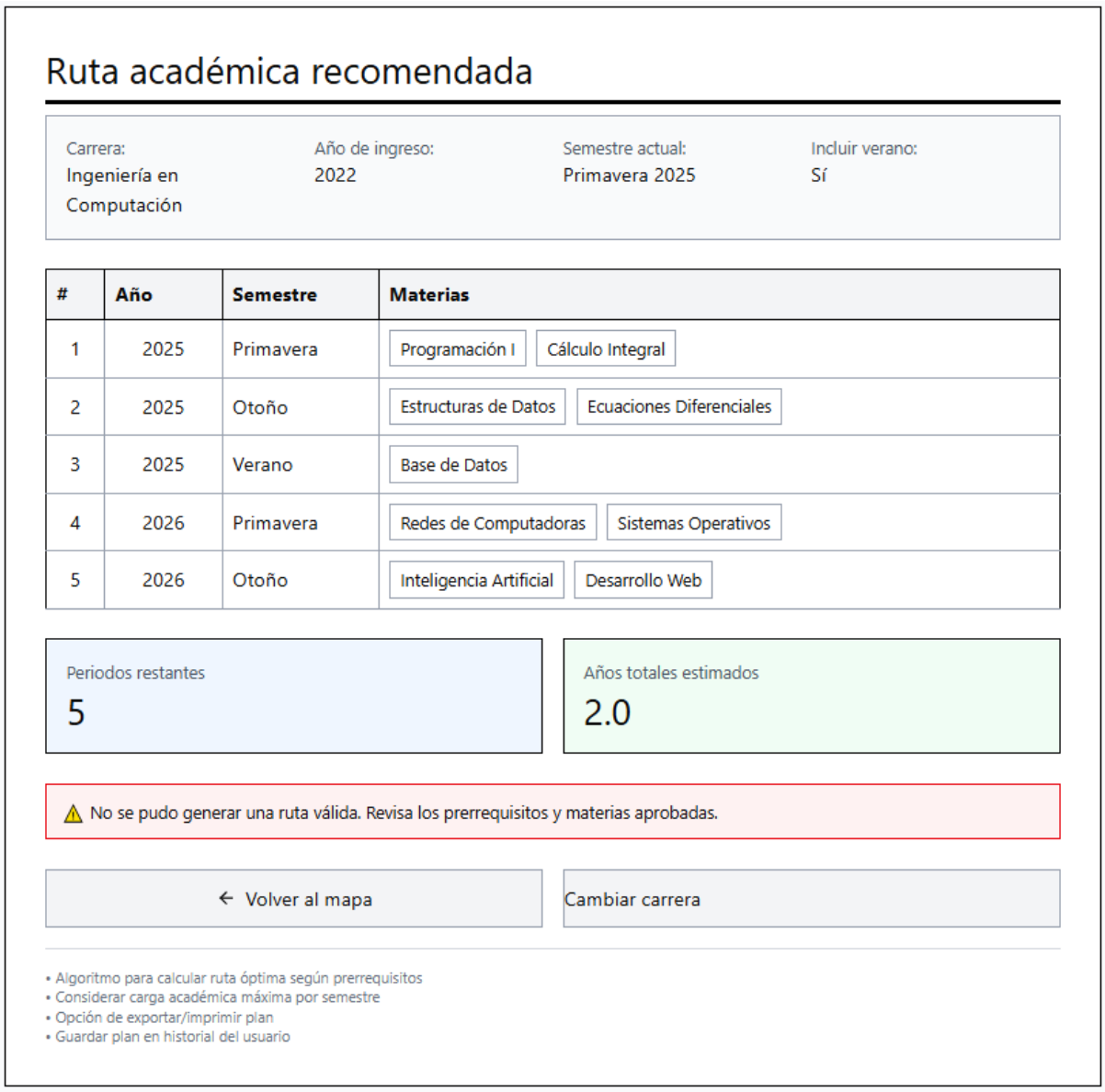
Nota: *La figura muestra el diseño del apartado donde se visualiza el historial guardado y se captura el periodo de inscripción y periodo actual. Elaboración propia.*

Diseño del módulo de generación de ruta académica

Por último se tiene la pantalla donde se mostrara la recomendación realizada por el algoritmo, esta sección se pensó cuidadosamente ya que este es el entregable al usuario y debe ser entendible y ordenado, se pensaba colocar en forma de diagrama su recomendación pero al querer programar la generación resultaba difícil de entender por el usuario además de que las líneas se podrían trazar de manera incorrecta causando confusión, es por esto que se prefirió mostrar las recomendaciones en forma de tabla, donde las columnas son los semestres y en la última fila se encuentra las materias a elegir por periodo, después de la tabla se muestran dos contenedores donde aparece el tiempo faltante para terminar de cursar su curricular académica y el tiempo total desde su ingreso a la carrera hasta el último periodo que contempla el algoritmo, este último contenedor es dinámico ya que si el tiempo rebasa el periodo de culminación normal (6 años) se cambia el color a amarillo o en rojo dependiendo su caso.

Figura 19

Interfaz de selección de verificación de guardado de historial y captura de periodos.



Nota: *La figura muestra el diseño del apartado donde muestra la ruta académica recomendada por el algoritmo. Elaboración propia.*

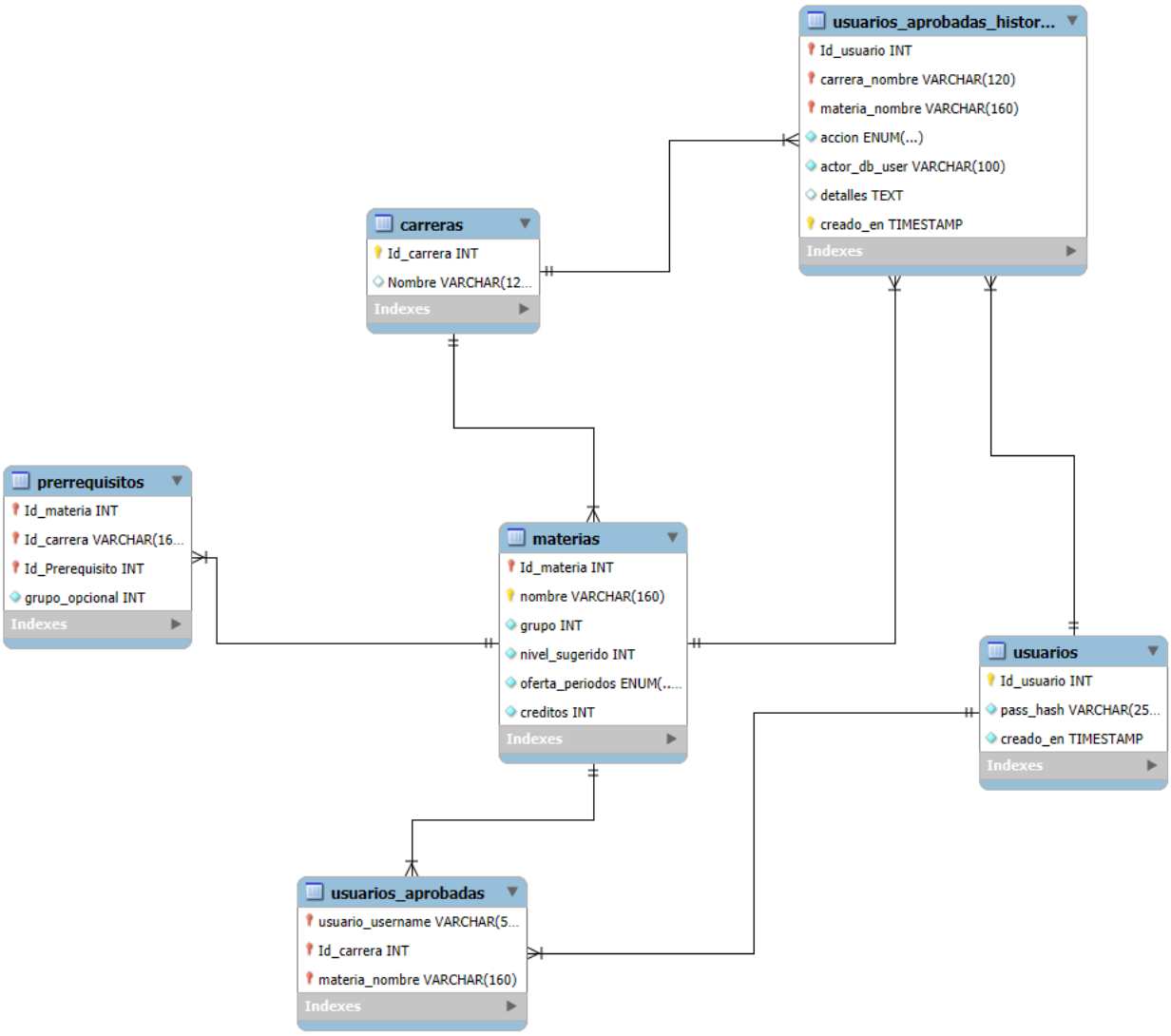
Diseño de base de datos

Para terminar la sección de análisis y diseño es importante dejar en claro cómo funcionará la base de datos, esta será el corazón del sistema ya que aquí se almacenarán todo tipo de datos como las materias, carreras, usuarios entre otros.

Las tablas deben estar relacionadas para tener un mejor control de los datos y a la hora de obtenerlos mediante código sea más fácil realizar consultas gracias a las conexiones que hay entre ellas. Para plasmar el funcionamiento de la base de datos se muestra en la figura 19 un diagrama entidad relación el cual se pueden ver como cada tabla se encuentra unida con otra.

Figura 20

Diagrama entidad relación de la base de datos.



Nota: *La figura muestra el modelo entidad relación de la base de datos. Elaboración propia.*

La tabla *usuarios* es utilizada para almacenar los datos básicos del estudiante para su futuro acceso, esta tabla está relacionada de muchos a muchos con *usuarios_aprobados* y uno a muchos con *usuarios_aprobadas_historial*.

En la tabla *carreras* se almacena el id y los nombres de las carreras disponibles, esta se relaciona mucho a muchos con *materias*, con

usuarios_aporbados_historial uno a muchos y con *usuario_aprobadas* muchos a muchos.

La tabla *materias* es una de las más importantes ya que ahí se encuentra el catálogo de todas las asignaturas existentes por carrera, esta es central ya que se relaciona con todas las tablas existentes (prerrequisitos, carreras, usuarios_aprobadas y usuario_aprobadas_historial), esto debido a que en cada tabla es requerido algún dato de *materias.*

Prerrequisitos es una tabla pivote donde define que materias son prerrequisitos de otras, aquí se almacenan las llaves primarias de carrera, materia y el id del prerrequisito, esta tabla hace una llave compuesta útil para le historial del alumno y la funcionalidad del algoritmo.

usuarios_aprobadas es una tabla tipo bitácora donde se registra las materias aprobadas por el alumno. Se relaciona con *usuarios, carrera* y *materias.*

usuarios_aprobadas_historial es parecida a la tabla usuarios_aprobadas en cuestión de relaciones solo que la función principal de esta es tener un registro de cambios realizados por el usuario ya que se utilizan funciones trigger para capturar el día y la hora cuando un usuario actualizo su información, esta tabla servirá para consultas a futuro.

Diccionario de datos

A continuación, se presentan tablas diccionario de cada tabla presentada en el diagrama entidad relación:

Tabla *usuarios*

Atributo	Tipo de Dato	Descripción
Id_usuario	INT	**Llave Primaria (PK)**. Identificador único del usuario.
pass_hash	VARCHAR(255)	Hash seguro de la contraseña del usuario.
creado_en	TIMESTAMP	Fecha y hora de creación del registro del usuario.

Tabla *carrera*

Atributo	Tipo de Dato	Descripción
Id_carrera	INT	**Llave Primaria (PK)**. Identificador único de la carrera.
nombre	VARCHAR(120)	Nombre completo de la carrera (e.g., Ingeniería de Software).

Tabla *materias*

Atributo	Tipo de Dato	Descripción
Id_materia	INT	**Llave Primaria (PK)**. Identificador único de la materia.
nombre	VARCHAR(160)	Nombre de la asignatura.
grupo	INT	Número de grupo al que pertenece la materia

nivel_sugerido	INT	Número de semestre o nivel académico recomendado para cursar la materia.
oferta_periodos	ENUM(...)	Define en qué periodos (ej. Primavera, Otoño) se ofrece la materia.
creditos	INT	Número de créditos otorgados por la materia.

Tabla *prerrequisitos*

Atributo	Tipo de Dato	Descripción
Id_materia	INT	**Llave Foránea (FK)**. Identificador de la materia que *requiere* un prerrequisito.
Id_prerrequisito	INT	**Llave Foránea (FK)**. Identificador de la materia que *es* el prerrequisito.
Id_carrera	VARCHAR(16)	**Llave Foránea (FK)**. Identificador de la carrera al que aplica el prerrequisito.
grupo_opcional	INT	Número de grupo al que pertenece la materia

Tabla *usuarios_aprobadas*

Atributo	Tipo de Dato	Descripción

usuario_username	VARCHAR(50)	**Llave Foránea (FK)**. del usuario que aprobó la materia.
Id_carrera	INT	**Llave Foránea (FK)**. de la carrera a la que pertenece la aprobación.
materia_nombre	VARCHAR(160)	**Llave Foránea (FK)**. de la materia aprobada.

Tabla *usuarios_aprobadas_historial*

Atributo	Tipo de Dato	Descripción
Id_usuario	INT	**Llave Foránea (FK)**. del usuario cuyo registro fue modificado.
carrera_nombre	VARCHAR(120)	**FK** del nombre de la carrera afectada.
materia_nombre	VARCHAR(160)	**FK** del nombre de la materia afectada.
accion	ENUM(...)	Tipo de acción registrada (ej. INSERT, UPDATE, DELETE).
actor_db_user	VARCHAR(100)	El usuario de la base de datos realizó la acción.
detalles	TEXT	Descripción o información adicional sobre el cambio realizado.
creado_en	TIMESTAMP	Fecha y hora en que se registró la acción de auditoría.

Capítulo 4: Heurística

Se probaron distintas técnicas, la primera de ellas basada en un autómata finito determinista para el cálculo de posibles rutas académicas a tomar por el estudiante, conforme a lo investigado un autómata finito determinista requiere de 5 elementos y uno de ellos es el conjunto de estados donde se debe colocar todas las posibles soluciones cuando el autómata procese una cadena, esto es eficiente cuando se cuenta con un grafo o posibles caminos ya definidos en un problema, pero para el caso de un generador de ruta académica, se presenta el caso en que encontramos materias, las cuales y no siguen un camino; esto se puede observar en las mallas curriculares donde algunas materias ya no desbloquean otra materia y en su línea de tiempo no se encuentra otra materia por tomar, en estos casos un autómata pensaría que ahí terminaría el camino y desperdiciaría ese espacio en vez de aprovecharlo para cursar una materia libre o sin prerrequisito.

Otro reto fue que se tendría que realizar un grafo el cual contenga todos los posibles caminos para que el autómata decida cual transitar con base en los estados liberados (materias aprobadas), pero sucede que no solo existe un solo camino a seguir, se pueden hacer distintos grafos con diferentes conexiones entre estados lo cual limitaría el sistema a una sola lógica en vez de buscar una opción más personalizada para el caso de un estudiante, además de que sería más tardado realizar el mejor grafo para cada carrera.

De lo anterior, se optó por usar un algoritmo voraz (Greedy Algorithm) con una heurística de priorización por nivel. Si bien un algoritmo voraz tiene un consumo

computacional elevado para el problema que se presenta puede ser aprovechado de manera más eficiente.

Un ejemplo de esto es el trabajo de Shi (2006) que encontró un problema de asignación a un conjunto de tareas interdependientes a un conjunto de procesadores para minimizar makespan (tiempo total de finalizacion), el autor hace uso de un algoritmo voraz conocido como list scheduling (planificación por lista) con ayuda de la heurística de priorización de nivel longest path first. Con el algoritmo calculó el nivel de cada tarea como la longitud de la ruta más larga, con esto cuando un procesador quedaba libre se le asignaba una tarea con una longitud mayor para así reducir el cuello de botella, de esta manera Shi logro minimizar el tiempo total de finalización de tareas al asignar primero las críticas.

Con este ejemplo se vió la utilidad a el problema de selección de materias ya que si un estudiante tiene ciertas materias acreditadas pero puede tomar otras materias que estén en semestres avanzados (ejemplo 6to semestre) su recomendación tome en cuenta tomar materias adelantadas conforme a su nivel actual, además de que se evita limitar a un estudiante a seguir un camino determinado, con esta implementación la ruta académica recomendada se personalizaría a la situación de cada persona, cumpliendo con el objetivo de este sistema.

Para la implementación de este algoritmo se utilizó Python, se generaron 2 códigos, la lógica del algoritmo se realizó en un archivo llamado *planner.py* y el código que consume los resultados del algoritmo que es *app.py*.

Por ejemplo, para la verificación de selección de materias que realiza un estudiante *app.py* solicita a *planner.py* que compare la selección actual con el historial previamente guardado para saber si el alumno puede tomar o no la nueva selección.

Figura 21

Fragmento de código en app.py.

```
if request.method == 'POST':
    # Selección enviada por el usuario
    sel = request.form.getlist('aprobada')
    nuevas_aprob = {int(x) for x in sel if x.isdigit()}

    # --- VALIDACIÓN de prerrequisitos (contra el conjunto completo resultante) ---
    materias_dict, prereqs = load_data(carrera_id)
    invalidas = []
    for mid in nuevas_aprob:
        if not can_take(mid, nuevas_aprob, prereqs):
            nombre = materias_dict.get(mid, {}).get('nombre', f'ID {mid}')
            invalidas.append(nombre)
    if invalidas:
        flash("No puedes guardar estas materias porque faltan prerrequisitos: " + ", ".join(invalidas), 'warning')
        cur.close(); cnx.close()
        return render_template(
            'select_materias.html',
            carrera_id=carrera_id,
            materias=materias,
            aprobadas=sorted(nuevas_aprob)
        )
```

Nota: *La figura muestra una funcionalidad en la clase app.py. Elaboración propia.*

Por ejemplo, en la línea *materias_dict, prereqs = load_data(carrera_id)* se puede apreciar como se manda a llamar el método *load_data* pasando como parámetro la id de la carrera, este método está en *planner.py.*

Figura 22 y 23

Fragmentos de código en planner.py (función load_data).

```python
def load_data(carrera_id):
    cnx = None
    cursor = None
    materias = {}
    prereqs = {}

    try:
        cnx = mysql.connector.connect(**DB_CONFIG)
        cursor = cnx.cursor(dictionary=True, buffered=True)

        # 1) Materias de la carrera
        cursor.execute("""
            SELECT id, nombre, grupo, nivel_sugerido AS nivel, oferta_periodos
            FROM materias
            WHERE carrera_id = %s
        """, (carrera_id,))
        rows = cursor.fetchall() or []
        for r in rows:
            raw = r['oferta_periodos']
            # Normaliza oferta de periodos a set {'Primavera','Verano','Otoño'}
            if raw is None or (isinstance(raw, str) and not raw.strip()):
                oferta = set(SEMESTRES)  # si prefieres que None signifique "ninguna", usa: set()
            elif isinstance(raw, str):
                oferta = {s.strip().title() for s in raw.split(',') if s.strip()}
            elif isinstance(raw, (list, tuple, set)):
                oferta = {str(s).strip().title() for s in raw}
            else:
                oferta = {s.strip().title() for s in str(raw).split(',') if s.strip()}
            oferta &= set(SEMESTRES)

            materias[int(r['id'])] = {
                'nombre': r['nombre'],
                'grupo':  int(r['grupo']),
                'nivel':  int(r['nivel']),
                'oferta': oferta
            }

        # 2) Detecta si existe grupo_opcional
        has_opt = False
        try:
            cursor.execute("SELECT materia_id, prerequisito_id, grupo_opcional FROM prerrequisitos LIMIT 1")
            _ = cursor.fetchall()  # si la tabla existe con esa columna, no lanza
            has_opt = True
        except mysql.connector.Error:
            has_opt = False  # columna no existe
```

```
    if has_opt:
        cursor.execute("SELECT materia_id, prerequisito_id, grupo_opcional FROM prerrequisitos")
        rows = cursor.fetchall() or []
        for row in rows:
            m = int(row['materia_id'])
            p = int(row['prerequisito_id'])
            g = int(row['grupo_opcional'] or 0)
            prereqs.setdefault(m, []).append((p, g))
    else:
        cursor.execute("SELECT materia_id, prerequisito_id FROM prerrequisitos")
        rows = cursor.fetchall() or []
        for row in rows:
            m = int(row['materia_id'])
            p = int(row['prerequisito_id'])
            prereqs.setdefault(m, []).append((p, 0))

    # Asegura retorno aunque no haya registros
    return materias, prereqs

finally:
    try:
        if cursor is not None: cursor.close()
    finally:
        if cnx is not None: cnx.close()
```

Nota: *La figura muestra la función load_data en la clase planner.py. Elaboración propia.*

Se puede ver como planner hace todas las consultas a la base de datos, consulta prerrequisitos y periodos, cuando *app.py* manda a llamar a *can_take (if not can_take(mid, nuevas_aprob, prereqs):),* si el proceso llega a encontrar un error, en automático el bucle en *app.py* se detiene y manda un mensaje de error el cual al usuario se devuelve como ***flash("No puedes guardar estas materias porque faltan prerrequisitos: " + ", ".join(invalidas), 'warning').***

Para la aplicación del algoritmo y de la heurística es parecido. Primero se aplica la heurística, lo que realiza es ordenar las materias de menor a mayor rango, esto depende del nivel que sea la materia (semestre), esto se hace en este método de la figura 23 y 24:

Figura 24 y 25

Fragmentos de código en planner.py (función load_data).

```
def plan_carrera(carrera_id,
                 aprobadas_ini,
                 entry_year, entry_sem_idx,
                 current_year, current_sem_idx,
                 include_verano):
    materias, prereqs = load_data(carrera_id)

    # 0) Cierre hacia atrás: si marcó avanzada, arrastra prereqs como aprobados
    aprobadas = set(int(x) for x in aprobadas_ini)
    stack = list(aprobadas)
    while stack:
        m = stack.pop()
        for p, _ in prereqs.get(m, []):
            if p not in aprobadas:
                aprobadas.add(p)
                stack.append(p)

    pendientes = set(materias) - aprobadas
    plan = []

    # offset de periodos desde ingreso hasta "hoy"
    offset = (current_year - entry_year)*3 + (current_sem_idx - entry_sem_idx)
    abs_p  = offset
    year   = current_year
    sem_idx= current_sem_idx

    def avanza(y, si):
        # siguiente índice
        si = (si + 1) % 3
        # si es Verano y no se incluye, salta a Otoño
        if si == 1 and not include_verano:
            si = 2
        # si volvimos a Primavera, incrementa año
        if si == 0:
            y += 1
        return y, si

    while pendientes:
        abs_p += 1
        year, sem_idx = avanza(year, sem_idx)
        sem = SEMESTRES[sem_idx]
        cup = CUPOS[sem]
```

```
    # regla de grupos por periodo absoluto
    if abs_p <= 4:      g_act = 1
    elif abs_p <= 10:   g_act = 2
    else:               g_act = 3

    # Conjunto congelado de aprobadas al inicio del periodo
    aprobadas_inicio = set(aprobadas)

    # Candidatas evaluadas contra 'aprobadas_inicio'
    candidatas = [
        m for m in pendientes
        if materias[m]['grupo'] <= g_act
        and sem in materias[m]['oferta']
        and can_take(m, aprobadas_inicio, prereqs)
    ]
    candidatas.sort(key=lambda m: (materias[m]['nivel'], materias[m]['nombre']))

    # Selección sin cadenas
    tomar = select_for_term(candidatas, aprobadas_inicio, prereqs, cup)

    if tomar:
        plan.append({
            'periodo': abs_p,
            'year':    year,
            'sem':     sem,
            'materias':[materias[m]['nombre'] for m in tomar]
        })
        # se aprueban al cerrar el periodo
        aprobadas.update(tomar)
        pendientes -= set(tomar)
    else:
        # no se pudo cursar nada este periodo (oferta/grupo); continúa
        continue

# Métricas de tiempo
periodos_restantes = len(plan)
anios_rest = periodos_restantes // 3
per_rest   = periodos_restantes % 3

final_abs = plan[-1]['periodo'] if plan else offset
total_desde_ing = final_abs
anios_tot = total_desde_ing // 3
per_tot   = total_desde_ing % 3

return plan, (anios_rest, per_rest), (anios_tot, per_tot)
```

Nota: *La figura muestra la función en donde es aplicada la heuristica en la clase planner.py. Elaboración propia.*

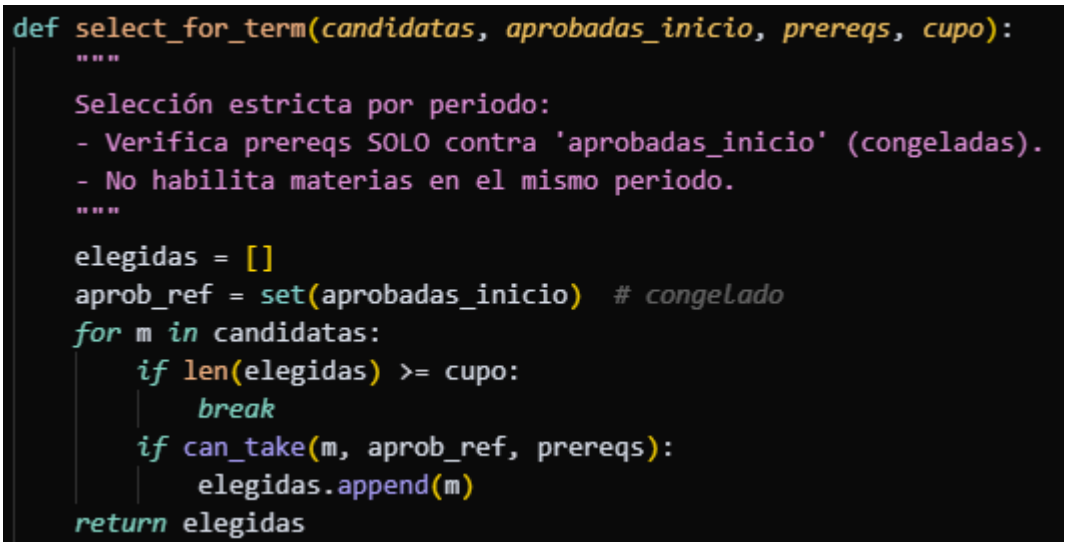
Este método es el corazón del algoritmo ya que realiza la lógica antes de formar la ruta académica y calcula el tiempo total que tomara terminar esta recomendación y el tiempo total de estadía (desde que entró a la carrera hasta el último periodo recomendado), la heurística se encuentra en la línea de código:

“candidatas.sort(key=lambda m: (materias[m]['nivel'], materias[m]['nombre']))”

su función es ordenar la lista de materias que podrían cursarse (candidatas) , con esto el algoritmo voraz toma las más importantes primero. El algoritmo voraz se ve aplicado en la figura 26:

Figura 26

Fragmento de código en planner.py (función select_for_term).



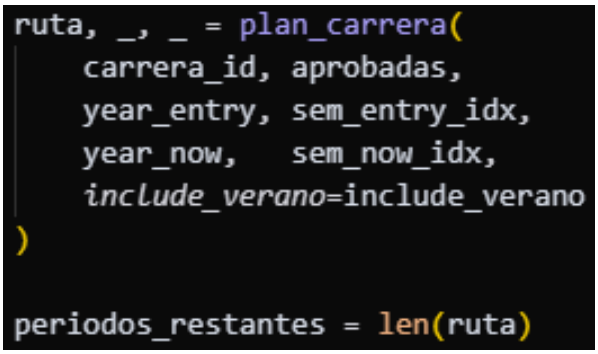
Nota: *La figura muestra la función en donde es aplicado el algoritmo voraz en la clase planner.py. Elaboración propia.*

Después de ordenar la lista de materias se invoca a esta función que implementa la voracidad. Simplemente va tomando las materias que van apareciendo del listado pasado por la heurística. En *for m in candidatas:* se encarga de recorrer el listado del nivel 1 a nivel n, después *if len(elegidas) >= cupo:* aplica la técnica voraz de ir tomando hasta que se terminen las materias o se alcance el cupo.

Por último, en *app.py* se manda la recomendación como la variable *ruta* en el metodo *calculate_route:*

Figura 27

Fragmento de código en app.py.

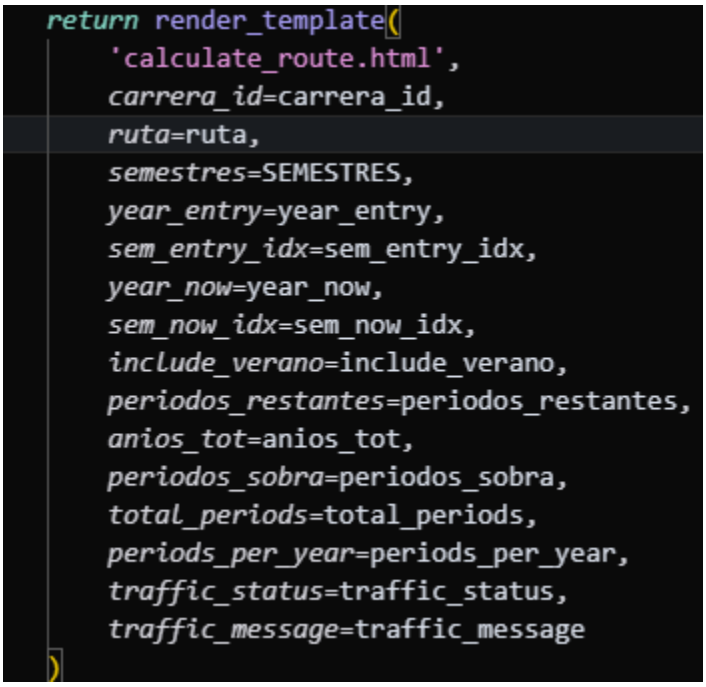


Nota: *La figura muestra como app.py manda los datos al algoritmo. Elaboración propia.*

pasa los datos a *planner.py* para hacer el calculo dicho anteriormente y después la ruta se retorna en esa variable para que se mande al frontend junto con todos los demás datos necesarios.

Figura 28

Fragmento de código en app.py.



```
return render_template(
    'calculate_route.html',
    carrera_id=carrera_id,
    ruta=ruta,
    semestres=SEMESTRES,
    year_entry=year_entry,
    sem_entry_idx=sem_entry_idx,
    year_now=year_now,
    sem_now_idx=sem_now_idx,
    include_verano=include_verano,
    periodos_restantes=periodos_restantes,
    anios_tot=anios_tot,
    periodos_sobra=periodos_sobra,
    total_periods=total_periods,
    periods_per_year=periods_per_year,
    traffic_status=traffic_status,
    traffic_message=traffic_message
)
```

Nota: *La figura muestra como app.py retorna los datos del algoritmo y los manda al frontend (calculate_route.html). Elaboración propia.*

El algoritmo junto con la heurística hace que la ruta académica recomendada sea optima ya que al tomar una materia de bajo nivel es lógico que se desbloquean más materias del siguiente nivel, evita el peor escenario que muchos alumnos han vivido, no poder tomar una materia de octavo semestre porque aun debe una de sexto semestre. El algoritmo siempre intentará tomar la materia de menor rango hasta que termine con esa línea pendiente, después puede hacer saltos.

Capítulo 5: Pruebas

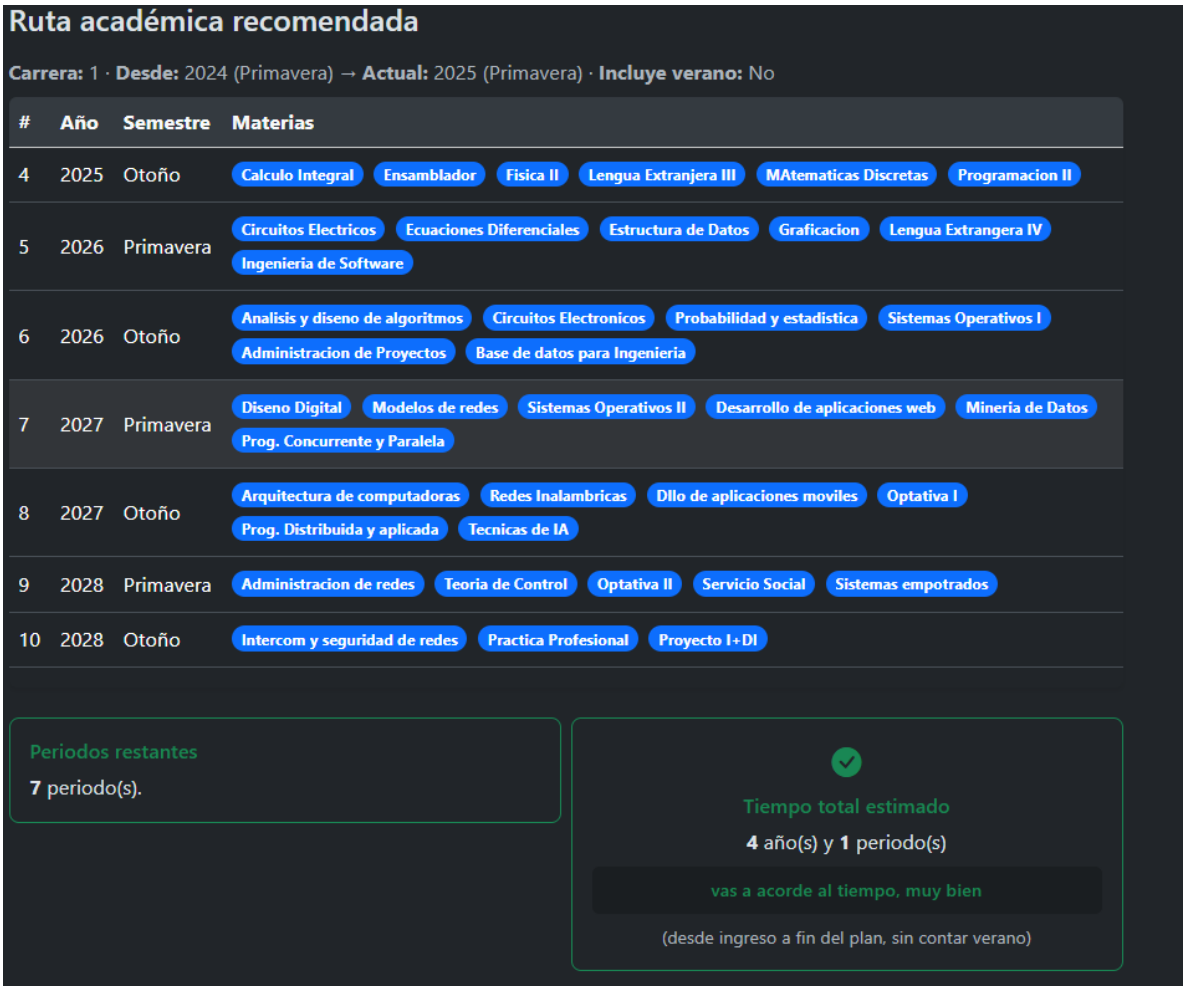
Para este último capítulo se harán pruebas de funcionalidad de la página, casos de alumnos los cuales tendrían un buen desempeño, los que irían regulares y los que estarían en dificultades, se hará uso de los mapas curriculares que ofrece la Facultad de Ciencias de la Computación de la BUAP (BUAP, FCC, 2017).

Caso ordinario

Para este ejemplo se hará la recomendación de un estudiante que va a la par con su mapa gráfico, es decir, en su semestre actual no cuenta con materias reprobadas, este mismo usuario será utilizado para las 3 carreras disponibles en el sistema.

Figura 29

Ejemplo de un estudiante de Ingeniería en Ciencias de la Computación.

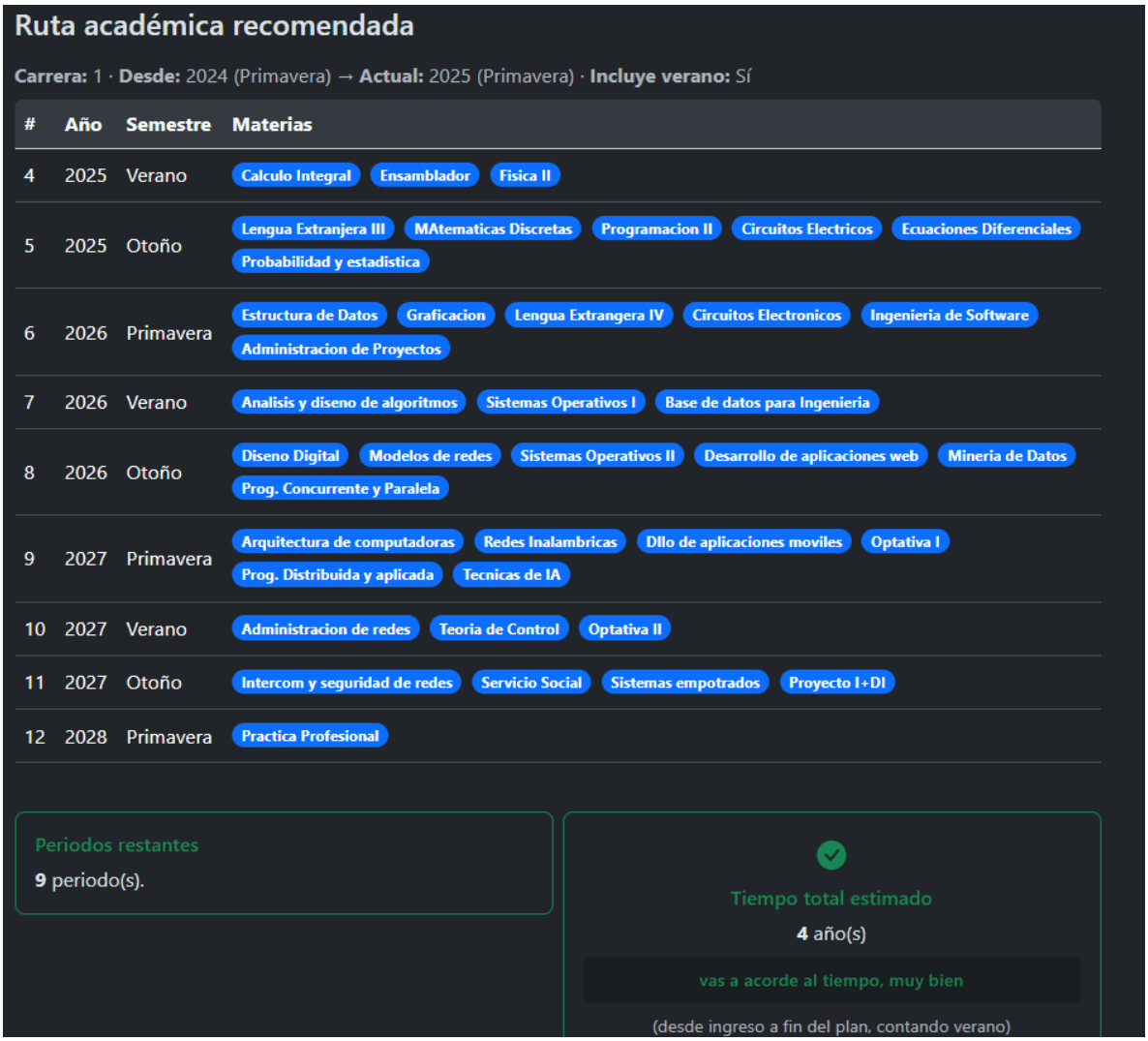


Nota: *La figura muestra la ruta académica del estudiante. Elaboración propia.*

Como se puede observar para el caso de este estudiante su ruta académica es óptima ya que le ahorra un semestre, además de que el semáforo indica que está dentro del tiempo establecido. Si se hace uso del periodo de verano su ruta académica se vuelve más eficiente como se muestra en la figura 21.

Figura 30

Ejemplo del mismo estudiante, pero con verano.

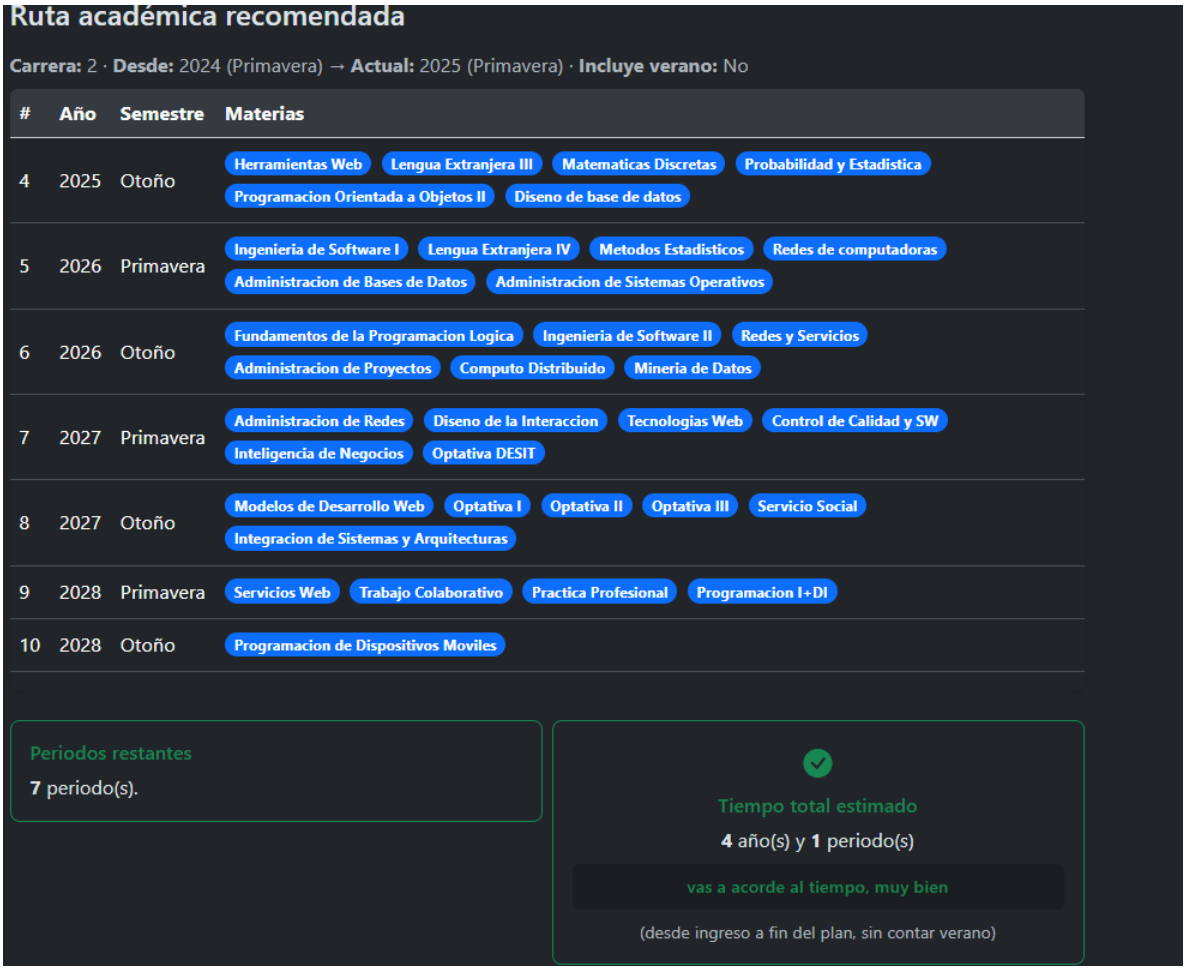


Nota: *La figura muestra la ruta académica del estudiante. Elaboración propia.*

Para este caso muestra el ahorro de 1 periodo, haciendo una carrera de 5 años a 4. Ahora el caso de un estudiante de Ingeniería en Tecnologías de la Información tiene distinta distribución de materias, se ve como se genera su mapa gráfico en la figura 22.

Figura 31

Ejemplo de un estudiante de Ingeniería en Tecnologías de la Información.



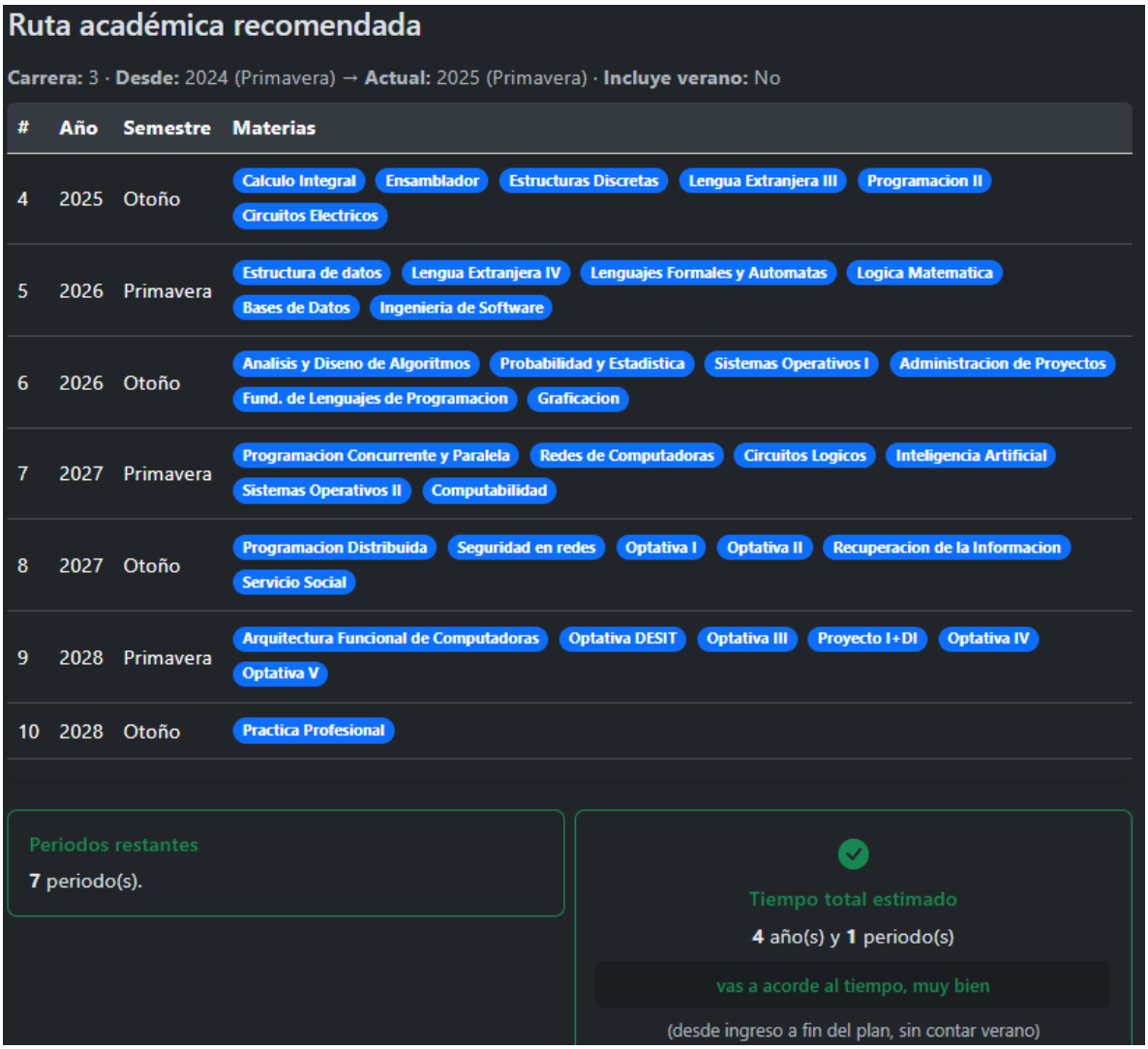
Nota: *La figura muestra la ruta académica del estudiante. Elaboración propia.*

Sucede lo mismo que el de Ingeniería en Ciencias de la Computación, su tiempo está dentro de lo normal.

Por último, se muestra el caso de un estudiante de Licenciatura en Ciencias de la Computación.

Figura 32

Ejemplo de un estudiante de Licenciatura en Ciencias de la Computación.



Nota: *La figura muestra la ruta académica del estudiante. Elaboración propia.*

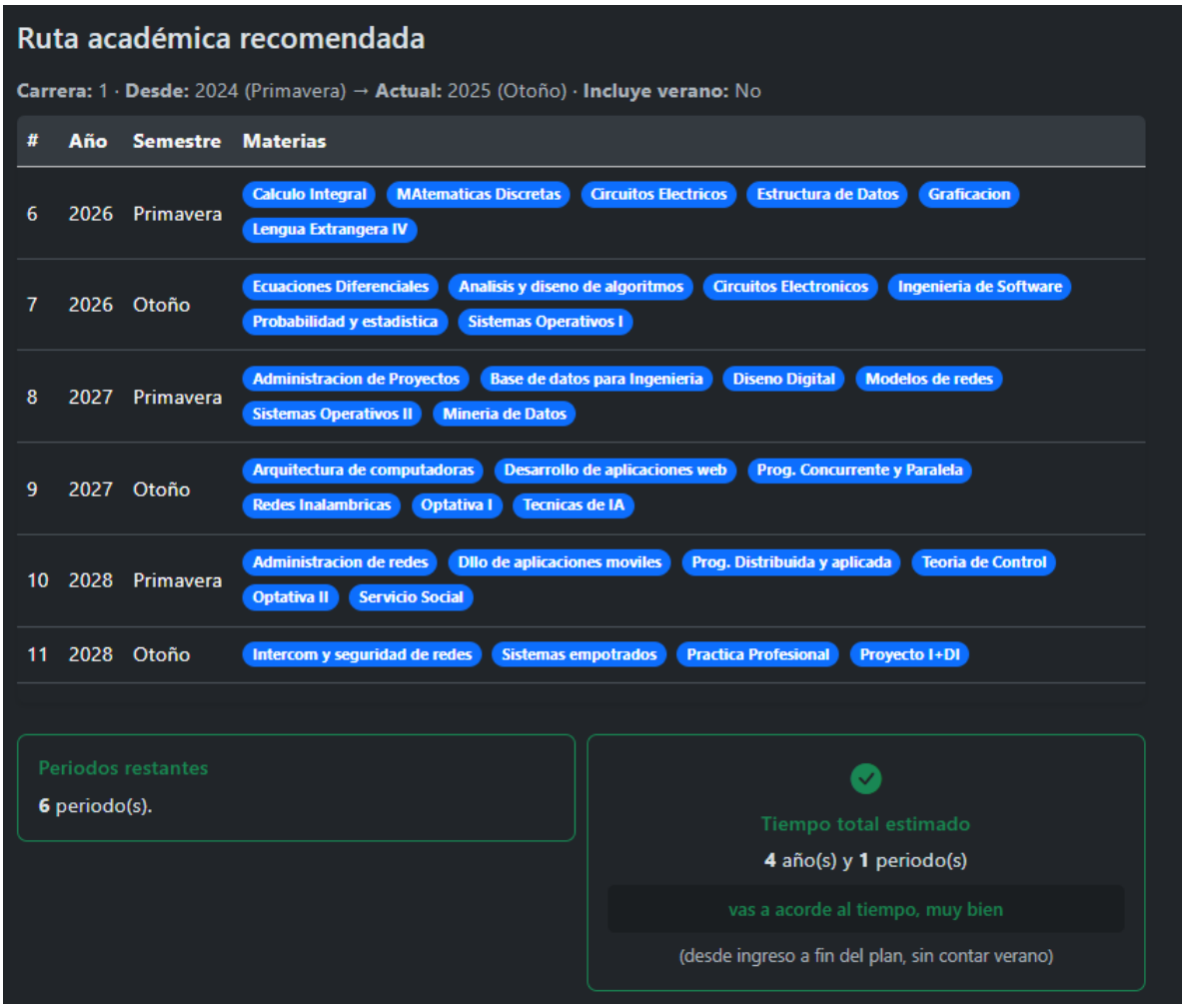
Tiende al mismo comportamiento, la carrera es de 5 años y sin incluir veranos el algoritmo le encontró un camino para terminar su carrera en 4 años y 1 periodo.

Caso regular

Para este caso se utilizará el mismo año de ingreso (2024) pero con un periodo más, además de que este alumno simularemos que reprobó 2 materias y se rezago un poco. Al igual se verá el comportamiento en las 3 carreras.

Figura 33

Ejemplo de un estudiante de Ingeniería en Ciencias de la Computación.

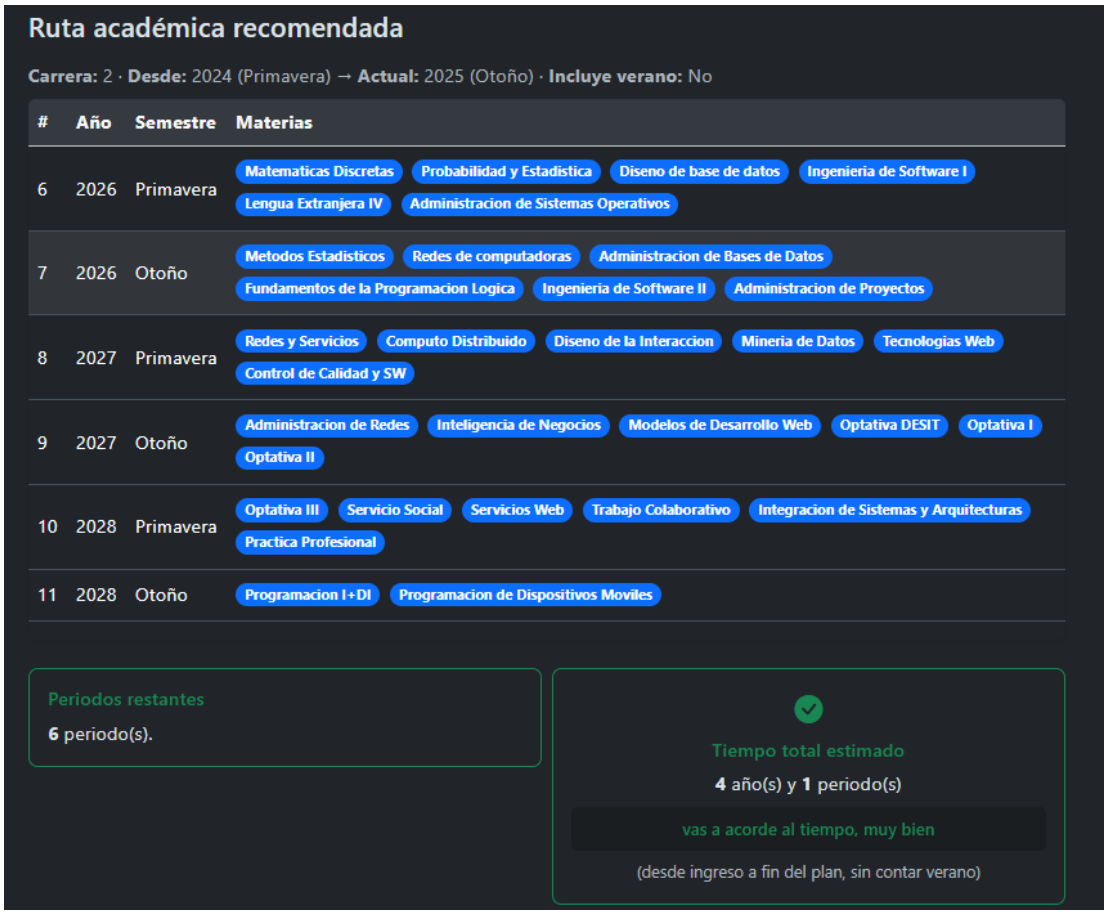


Nota: *La figura muestra la ruta académica del estudiante. Elaboración propia.*

Para el ejemplo anterior se marcó como no acreditadas 2 materias (calculo integral y matemáticas discretas), el algoritmo recomendó tomarlas el siguiente semestre que sería primavera 2026 como se ve en el ejemplo y se percibe que el sistema recomienda hacer un salto en otoño 2026, recomienda tomar ecuaciones diferenciales junto con probabilidad y estadística, logrando liberar minería de datos para su siguiente semestre y agilizar su ruta académica. A pesar de rezagar al alumno con 2 materias el sistema logro hacer una ruta con el mismo tiempo de un alumno ordinario. Ahora para el caso de un estudiante de Ingeniería en Tecnologías de la Información.

Figura 34

Ejemplo de un estudiante de Ingeniería en Tecnologías de la Información.

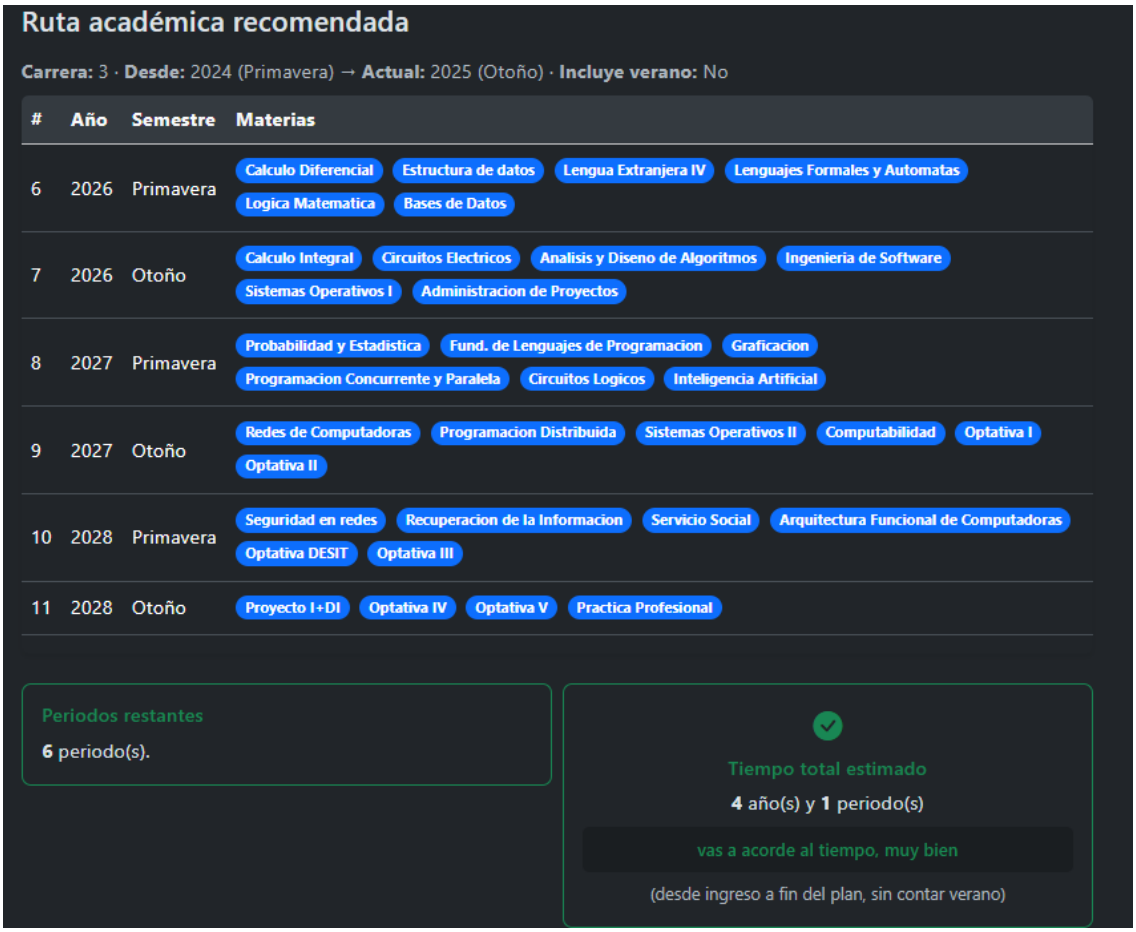


Nota: *La figura muestra la ruta académica del estudiante. Elaboración propia.*

En este caso se simulo el rezago en 2 materias del área matemática (matemáticas discretas y probabilidad y estadística), a pesar de esto el sistema recomienda retomar esas materias en el siguiente semestre y ahora hace un salto, toma la materia de administración de sistemas operativos que es 1 semestre adelantado al simulado y con esto logra ahorrar 1 periodo del cual se podía rezagar con el área matemática ya que esta materia es un nodo fuerte, es decir, desbloquea más materias por delante. También se aprecia que el tiempo es igual al de un alumno ordinario. Por último, toca el caso de licenciatura.

Figura 35

Ejemplo de un estudiante de Licenciatura en Ciencias de la Computación.



Nota: *La figura muestra la ruta académica del estudiante. Elaboración propia.*

Para este caso se desacreditaron 2 materias del área matemática, en el área de licenciatura calculo diferencial y calculo integral están unidas (una desbloquea a la otra) y siguen desbloqueando más por delante, si un alumno no acredita estas 2 materias o 1 habría un rezago notorio pero el algoritmo a pesar de esto recomienda tomar una un semestre después ya que no se pueden tomar al mismo tiempo pero se puede notar como hace un salto en la materia base de datos ya que es de un semestre más adelante, con este salto desbloquea más materias y

recomienda hacer más saltos en lo que el alumno se recupera en el área matemática, en cuestión al tiempo se nota el mismo comportamiento.

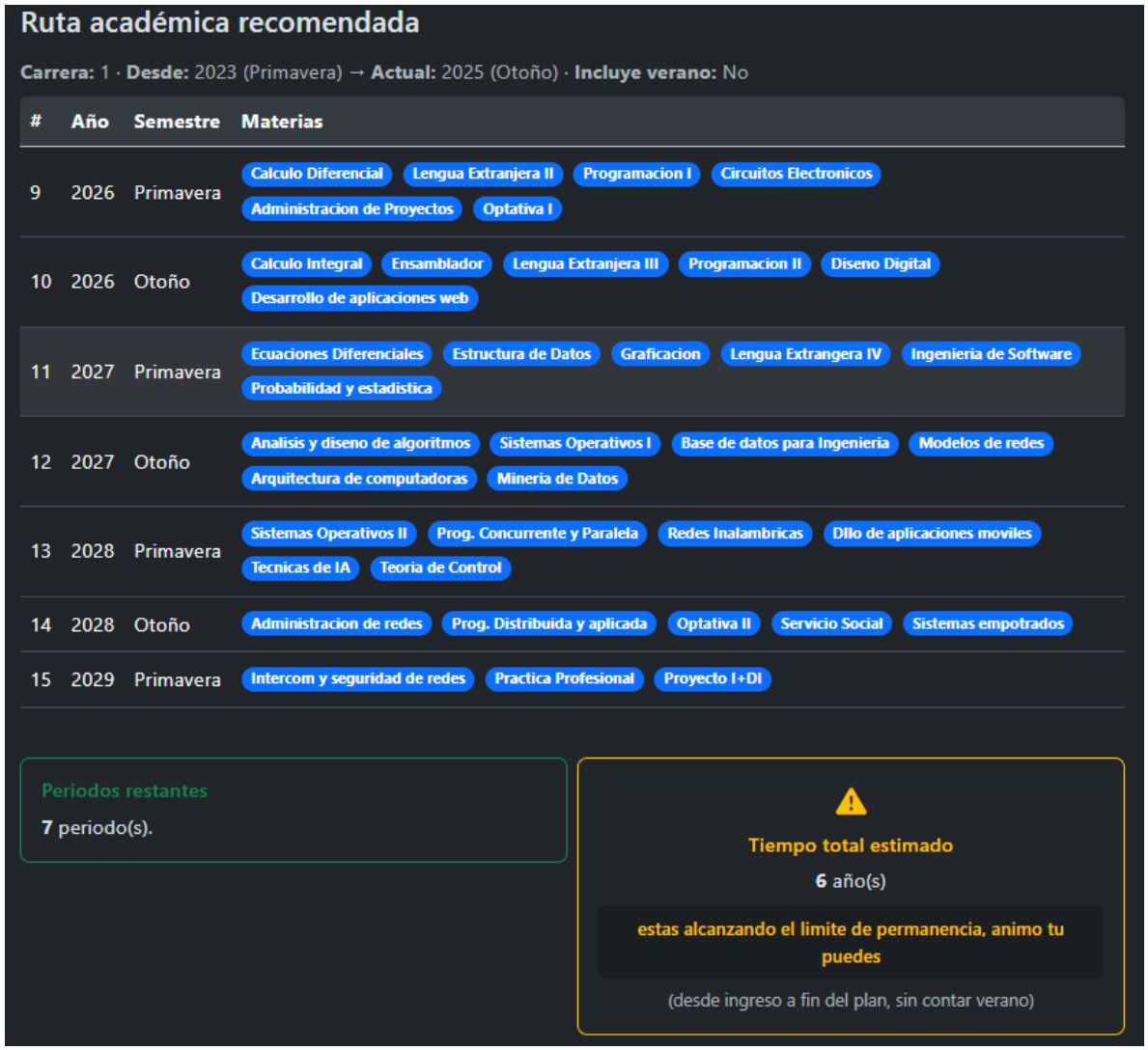
Como se vieron en los 3 casos los saltos que recomienda el algoritmo son muy buenos para que se evite un rezago significativo al alumno, además de que en los ejemplos no se incluyó verano, estas recomendaciones podrían tener el mismo comportamiento de la figura 21 donde se aprecia como se pudo ahorrar 1 periodo más.

Caso semi irregular

Para este caso se utilizará 1 año menos a los anteriores (2023) y también el alumno estará atrasado completamente en una línea de su área con más carga y además tendrá 2 materias sin acreditar en cualquier otra área.

Figura 36

Ejemplo de un estudiante de Ingeniería en Ciencias de la Computación.

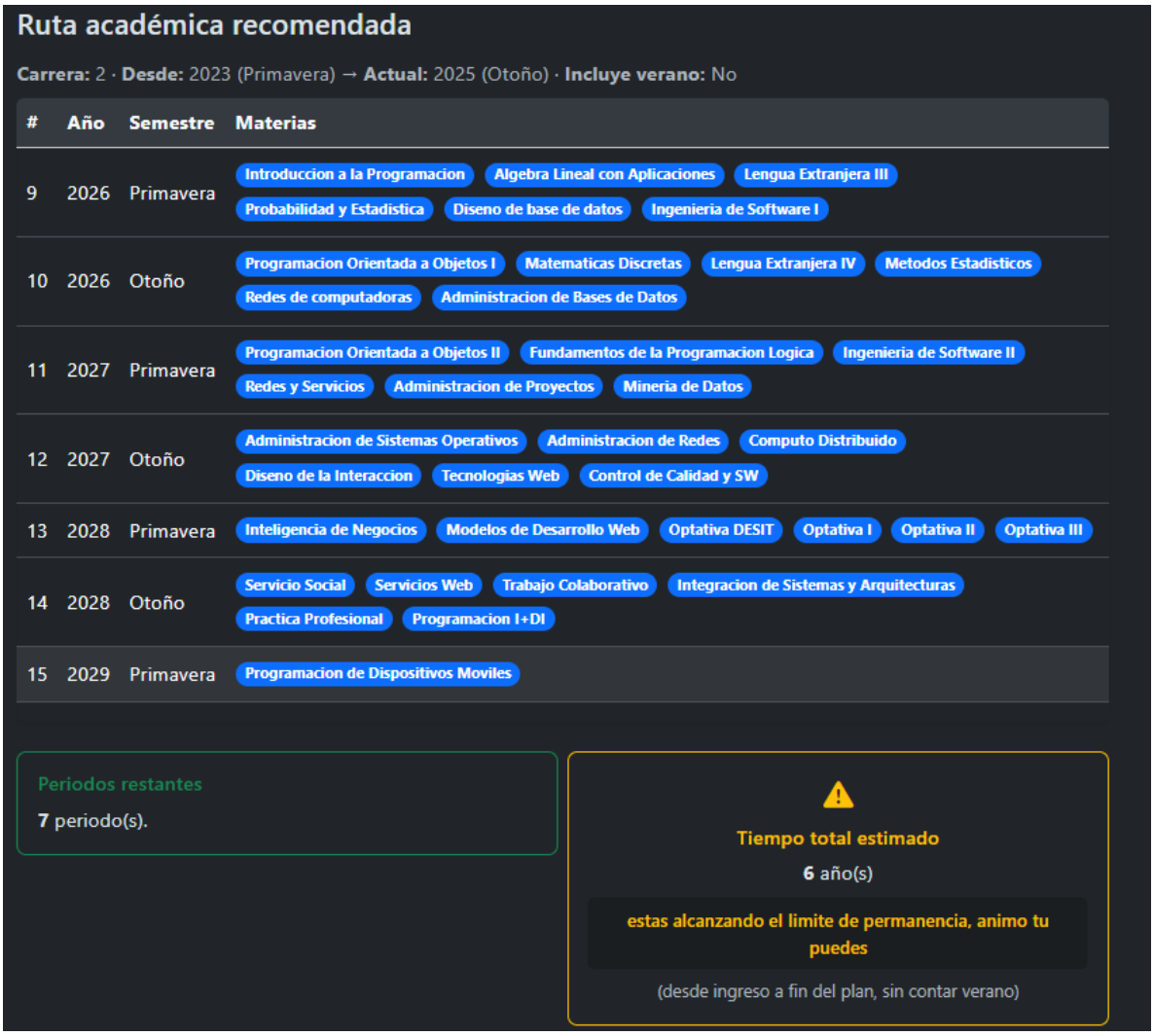


Nota: *La figura muestra la ruta académica del estudiante. Elaboración propia.*

Para este caso se simulo un rezago en su área formativa de ingeniería y obtuvo una demora en las materias de inglés y matemáticas, ahora vemos el funcionamiento del semáforo ya que el estudiante se está saliendo del tiempo normal establecido. Ahora para un estudiante de Tecnologías de la Información el comportamiento es el siguiente bajo la misma dinámica.

Figura 37

Ejemplo de un estudiante de Ingeniería en Tecnologías de la Información.



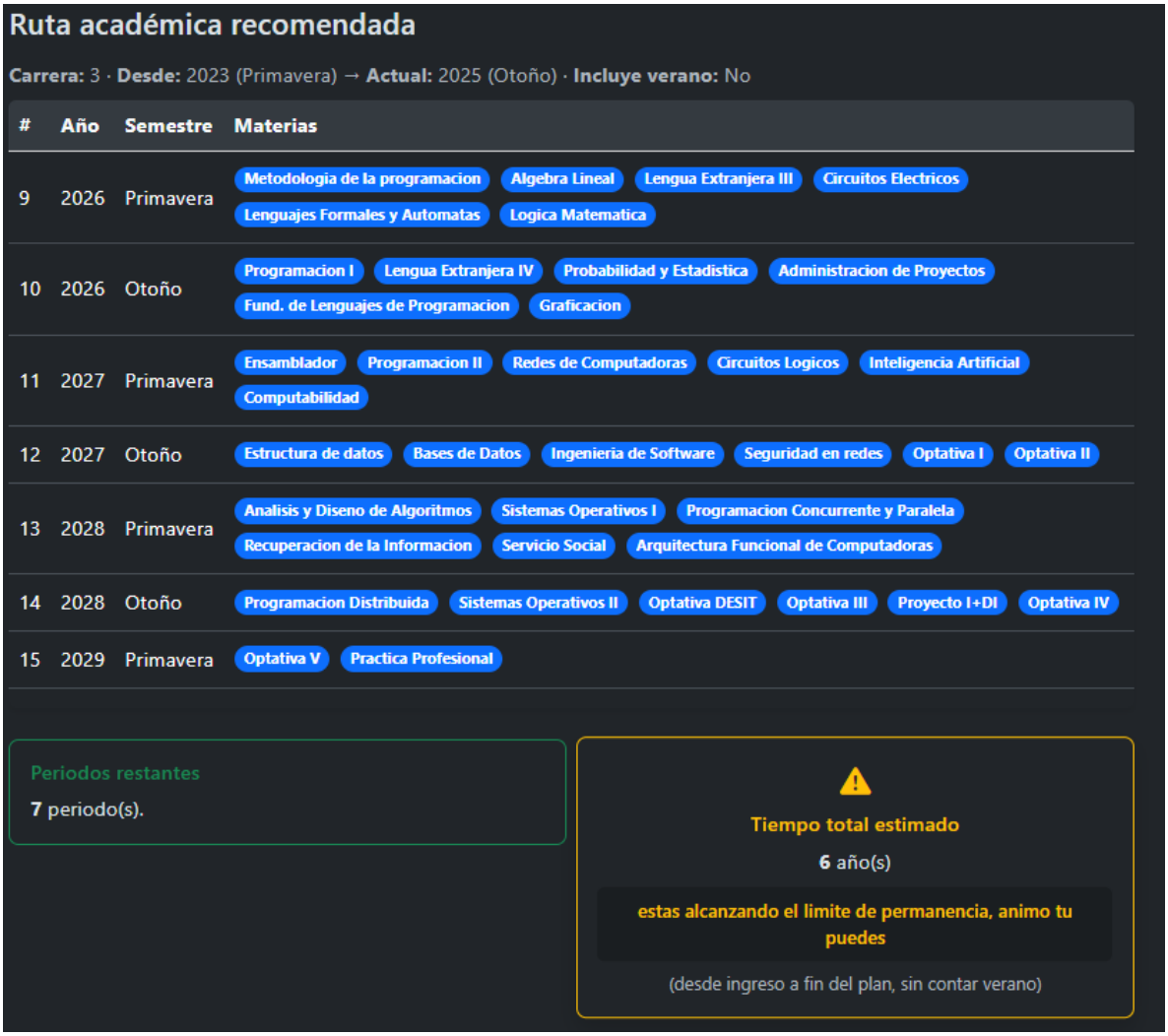
Nota: *La figura muestra la ruta académica del estudiante. Elaboración propia.*

Se nota un mismo comportamiento al anterior, se le afecto en su área de modelado de sistemas, en la lengua extranjera y en el área matemática, a pesar de que sus materias están muy relacionadas el algoritmo logro acomodar sus materias rezagadas junto con las de su semestre actual y logro determinar una ruta algo favorable en cuestión de tiempo.

Por último, toca analizar el comportamiento de un estudiante de Licenciatura en Ciencias de la Computación.

Figura 38

Ejemplo de un estudiante de Licenciatura en Ciencias de la Computación.



Nota: *La figura muestra la ruta académica del estudiante. Elaboración propia.*

Se puede apreciar como es el mismo caso, igual se simulo un rezago en el área de ciencias computacionales, lengua extranjera y matemáticas. En las 3 carreras el semáforo y el tiempo se mantuvieron iguales pero la carga en cada semestre

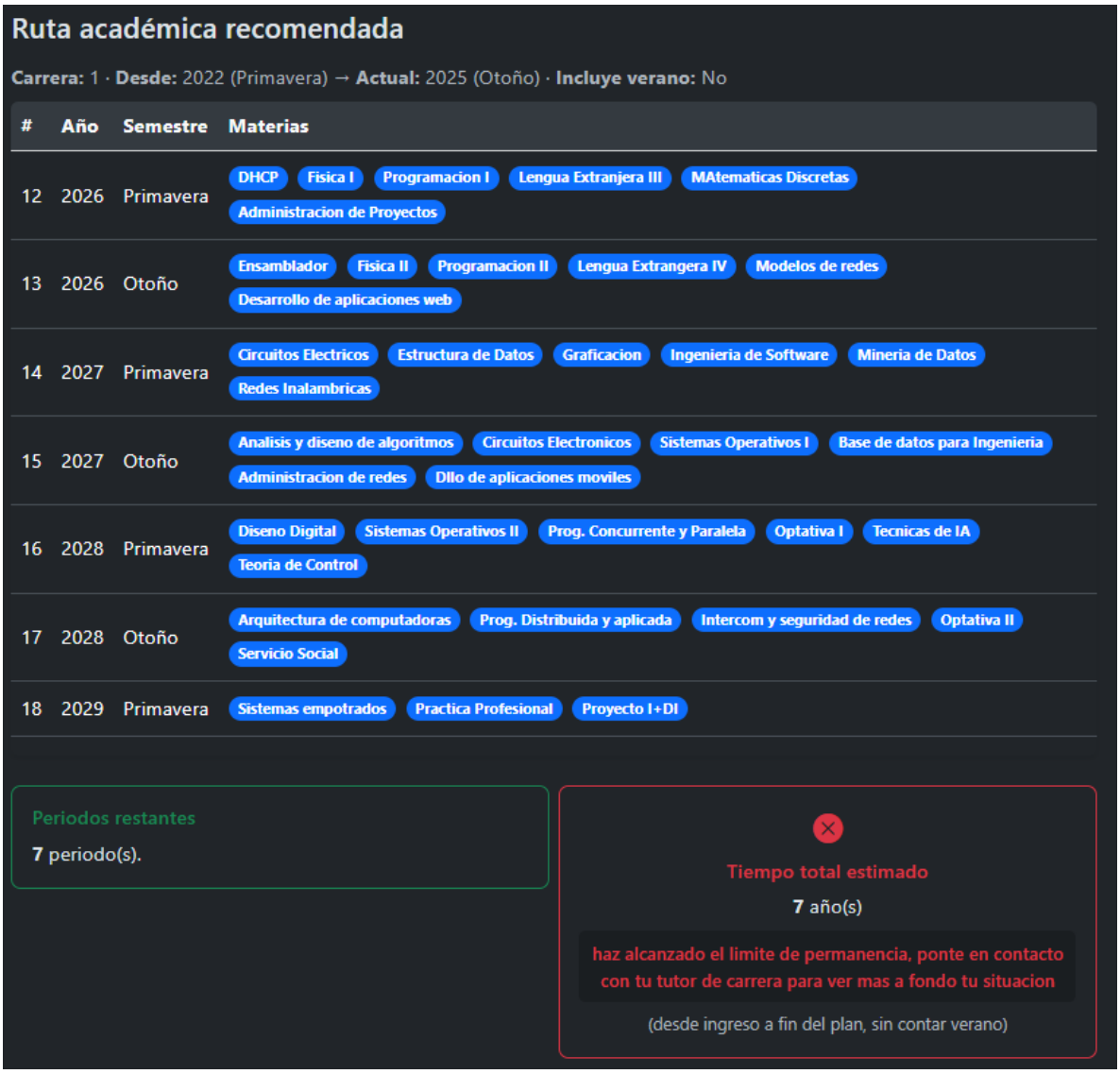
fue distinta, en especial Ingeniería en Ciencias de la Computación ya que en su último periodo recomendado tuvo 3 materias.

Caso irregular

Para este caso se utilizará una matrícula del año 2022, para esta simulación el estudiante ya debió haber completado el nivel formativo, pero se presentará el caso donde este muy rezagado en un área y tenga 3 materas pendientes que no le dejen desbloquear más su trayecto.

Figura 39

Ejemplo de un estudiante de Ingeniería en Ciencias de la Computación.



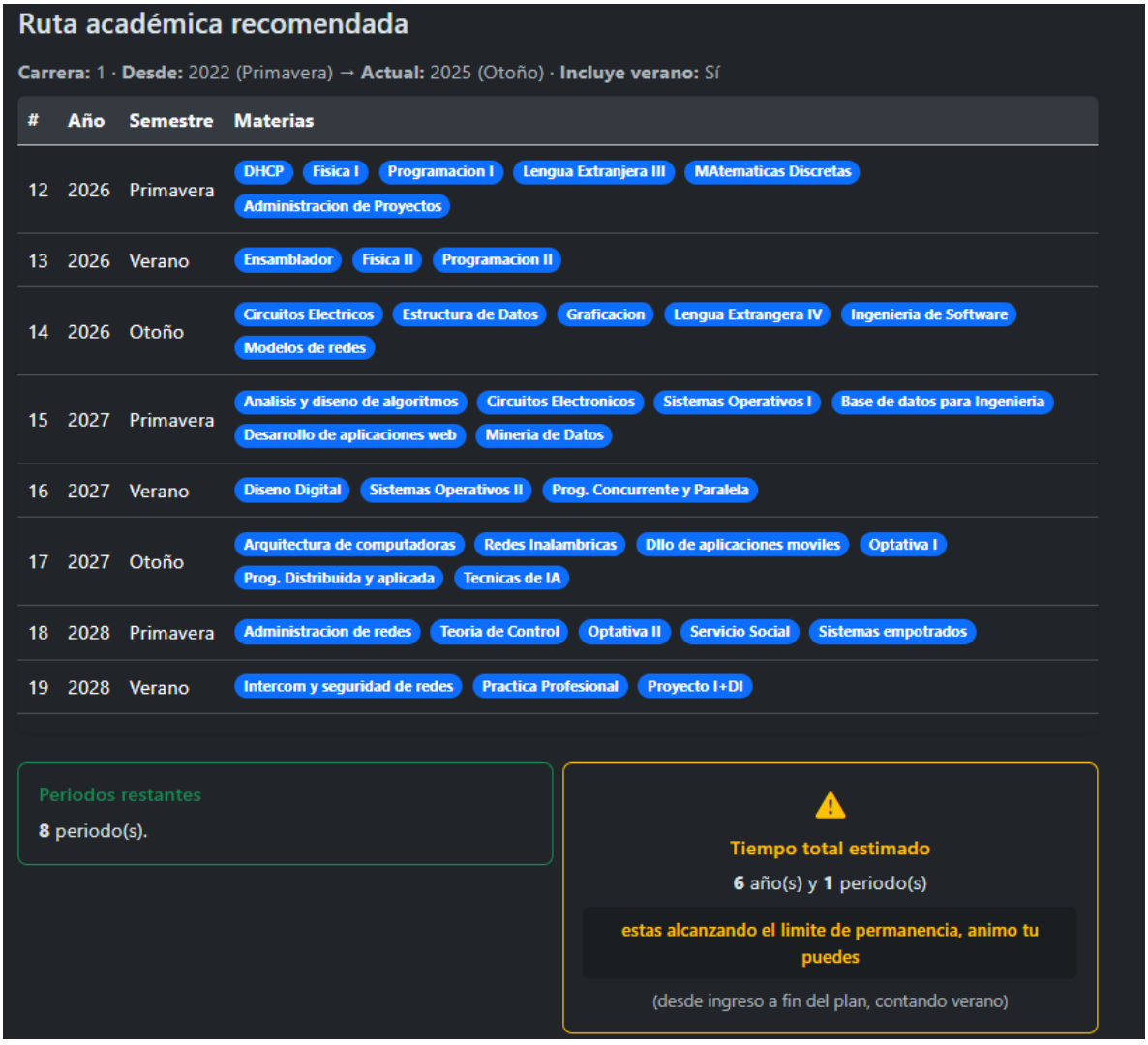
Nota: *La figura muestra la ruta académica del estudiante. Elaboración propia.*

Para este caso ya se presenta un nuevo estado del semáforo, el cual es cuando supera el tiempo límite de permanencia, en esta simulación el estudiante se rezago en materias de 2do. y 3er. semestre ocasionando que a su semestre normal (sexto) le faltaran materias. Del mismo modo en las otras 2 carreras se presentó esta situación, con el mismo tiempo total estimado y la misma situación, siguiendo este ejemplo en la figura 31 se muestra como hay un cambio si se

incluye el verano en cada semestre, se ahorra 1 periodo por lo que vuelve al semáforo en color naranja significando que el estudiante tiene salvación de no ser dado de baja.

Figura 40

Ejemplo estudiante de Ingeniería en Ciencias de la Computación con verano.



Nota: *La figura muestra la ruta académica del estudiante. Elaboración propia.*

Conclusión y trabajo a futuro

En el proyecto se aprecia como haciendo uso de un algoritmo voraz con una heurística se logra solucionar el problema de generar rutas académicas personalizadas, la combinación de estos 2 lograron cumplir el objetivo de generar una ruta óptima a pesar de simular rezagos en estudiantes, esto deja un aporte a las ciencias computacionales de como una técnica que no se considera óptima en cuestión al costo computacional logra resolver este tipo de situaciones en un buen tiempo. Gracias a esto el sistema es capaz de brindar una herramienta de apoyo a la comunidad estudiantil y un mejor acercamiento de tutores hacia los avances de cada uno de sus alumnos.

Trabajo a futuro

El proyecto cumple con el objetivo principal de generar rutas académicas eficientes conforme la situación de un alumno, con esto se deja una gran base funcional para ser implementada a futuro, el proyecto se puede volver más robusto y formal con las siguientes aportaciones:

- Tomar en cuenta los créditos para la recomendación de una ruta académica.
- Agregar las nuevas carreras que se encuentran en CU2 y actualizar los mapas curriculares para las 3 carreras existentes.
- Implementación de una interfaz de administrador para tener un control de los cambios realizados por los estudiantes en su historial.

- Implementar una función la cual permita filtrar estudiantes por carrera y determinar estadísticas como por ejemplo el número de alumnos que solicitaran una materia, número de alumnos que van rezagados en un área, ver cuántos alumnos están en una situación crítica, entre otros.
- Implementar una sección donde los tutores de cada grupo de estudiantes visualicen el avance del estudiante y validar que la ruta académica generada es considerada buena a seguir o no.

Con los puntos descritos anteriormente el proyecto podrá abarcar más usuarios como tutores y secretaria académica, no solamente será utilizado para generar rutas, sino que también se podrá utilizar para gestionar los avances de los alumnos, sacar estadísticas como número de materias a ofertar, cantidad de alumnos en riesgo entre otros, además de que se fortalece la comunicación entre alumno y tutor ya que habría una mejor gestión en el progreso del alumno.

Otros factores externos que aumentarían la capacidad del proyecto son los siguientes:

- Hacer uso de servidores para que el algoritmo tenga poder y permita la generación de rutas académicas a más estudiantes.
- Tener vinculada la base de datos donde se encuentre la información de los alumnos que se encuentran activos en la universidad, de este modo se evita el modulo de registro haciendo al sistema mas seguro y evitando la generación de usuarios ficticios que puedan afectar el rendimiento del sistema.

- Implementar la malla curricular de más carreras para mejorar el alcance y no dejarlo solo a la Facultad de Ciencias de la Computación.
- Ofrecer mayor divulgación a la comunidad estudiantil sobre esta herramienta.

Referencias

INEGI. (2024). Tasa de deserción escolar en educación superior, Puebla. Recuperado de https://www.inegi.org.mx/app/tabulados/interactivos/?pxq=9171df60-8e9e-4417-932e-9b80593216ee

Dirección General de Educación Superior Universitaria e Intercultural (DGESUI). (2023). Subsidio ordinario a la BUAP. Recuperado de https://subsidioentransparencia.dgesui.ses.sep.gob.mx/2023/subsidio-ordinario/universidad/BUAP

Kamble, U., Nalawade, S., Mahadik, T., Salunke, Y., Dedgaonkar, S., & Shelke, P. (2024). Survey of application of automata theory in natural language processing. In 2024 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems (ICETSIS) (pp. 1-4). Manama, Bahrain. https://doi.org/10.1109/ICETSIS61505.2024.10459556

Kostanyan, A. (2017). Fuzzy string matching with finite automat. In 2017 Computer Science and Information Technologies (CSIT) (pp. 9-11). Yerevan, Armenia. https://doi.org/10.1109/CSITechnol.2017.8312128

Hernández, M. G. (2015, 22 julio). Deserción universitaria en México. Grupo Milenio. https://www.milenio.com/opinion/maximiliano-gracia-hernandez/la-economia-del-tunel/desercion-universitaria-en-mexico

Secretaría de Educación Pública. (2023). Estadística educativa e indicadores: Puebla, ciclo escolar 2022-2023. Dirección General de Planeación, Programación y Estadística Educativa. https://planeacion.sep.gob.mx/Doc/estadistica_e_indicadores/EstIndEntFed2023/21_PUE.pdf

Grajales, G. (2024, 10 septiembre). El “abandono” y la “eficiencia” en la BUAP | Guadalupe Grajales. E-consulta Puebla | Referencia Obligada. https://www.e-consulta.com/opinion/2024-09-10/el-abandono-y-la-eficiencia-en-la-buap

Dharmapurikar, S., & Lockwood, J. W. (2006). Fast and scalable pattern matching for network intrusion detection systems. IEEE Journal on Selected Areas in Communications, 24(10), 1781–1792. https://doi.org/10.1109/JSAC.2006.877131

Zare, M., Jampour, M., Arezoomand, A. S., & Sabouri, M. (2019). Handwritten recognition based on hand gesture recognition using deterministic finite automata and fuzzy logic. In 2019 4th International Conference on Pattern Recognition and Image Analysis (IPRIA) (pp. 93-99). Tehran, Iran. https://doi.org/10.1109/PRIA.2019.8786038

Rational Unified Process (RUP) Plug-ins for Rational Method Composer 7.5.1. (s. f.). https://www.ibm.com/support/pages/rational-unified-process-rup-plug-ins-rational-method-composer-751

Dwinandana, M., Riskiana, R. R., & Kusumo, D. S. (2022). Extended finite state machine-model based testing on mobile application. In 2022 1st International Conference on Software Engineering and Information Technology (ICoSEIT) (pp. 41-45). Bandung, Indonesia. https://doi.org/10.1109/ICoSEIT55604.2022.10030001

Cisneros-Bravo, B. E., Rodríguez-Aguilar, R. M., Niño-Membrillo, Y. E., & Cuevas-Rasgado, A. D. (2023). Falta de orientación vocacional como factor en la deserción universitaria. Caso de estudio: zona Oriente del Estado de México. RIDE Revista Iberoamericana Para La Investigación Y El Desarrollo Educativo, 14(27). https://doi.org/10.23913/ride.v14i27.1715

Object Management Group. (2013). Business Process Model and Notation (BPMN) Version 2.0.2 (ISO/IEC 19510:2013). Needham, MA: OMG. Disponible en https://www.omg.org/spec/BPMN

IEEE Computer Society. (1998). IEEE recommended practice for software requirements specifications (IEEE Std 830-1998). Institute of Electrical and Electronics Engineers. https://doi.org/10.1109/IEEESTD.1998.88286

Kruchten, P. (2003). *The Rational Unified Process: An Introduction* (3rd ed.). Addison-Wesley Professional.

Jacobson, I., Booch, G., & Rumbaugh, J. (1999). *The Unified Software Development Process*. Addison-Wesley Professional.

Change Vision, Inc. (2024). Astah UML (Version 8.5.0) [Software de modelado]. https://astah.net/

Shi, Z. (2006). Scheduling tasks with precedence constraints on heterogeneous distributed computing systems [Tesis doctoral, University of Tennessee]. Trace: Tennessee Research and Creative Exchange. https://trace.tennessee.edu/cgi/viewcontent.cgi?article=3477&context=utk_graddiss

Benemérita Universidad Autónoma de Puebla, Facultad de Ciencias de la Computación. (2021). Mapa curricular de la Licenciatura en Ciencias de la Computación (LCC). Recuperado de https://www.cs.buap.mx/

Benemérita Universidad Autónoma de Puebla, Facultad de Ciencias de la Computación. (2017). Mapa curricular de la Ingeniería en Ciencias de la Computación (ICC). Recuperado de https://www.cs.buap.mx/

Benemérita Universidad Autónoma de Puebla, Facultad de Ciencias de la Computación. (s.f.). Mapa curricular de la Ingeniería en Tecnologías de la Información (ITI). Recuperado de https://www.cs.buap.mx/