



TEC, Tecnológico de Costa Rica

Proyecto #2

Profesor:

Luis Pablo Soto Chaves

Integrantes:

Gustavo Vargas Ramírez

José Andrés Salazar Rodríguez

Sidney Salazar Jiménez

Curso:

POO, programación Orientada a Objetos

Semestre:

Primer semestres, 2024

ÍNDICE

INTRODUCCIÓN
ESTRATEGIA METODOLÓGICA
CRONOGRAMA DE TRABAJO
DIAGRAMA GENERAL DE CASOS DE USO
DIAGRAMA DE ARQUITECTURA CONCEPTUAL
DIAGRAMA DE CLASES DE BAJO NIVEL
JUSTIFICACIÓN DE RELACIONES ENTRE OBJETOS
DIAGRAMA DE PAQUETES
ANÁLISIS DE RESULTADOS
DOCUMENTACIÓN JAVADOC
LECCIONES APRENDIDAS
BITÁCORAS INDIVIDUALES
BIBLIOGRAFÍA Y FUENTES
MANUAL DE USUARIO
ANEXOS

INTRODUCCIÓN

En el contexto actual de la educación, la gestión eficiente y precisa de los planes de estudio es esencial para garantizar la calidad y coherencia de la formación académica. Un sistema de gestión de planes de estudio es una herramienta integral que facilita la administración de diversos elementos educativos, como escuelas, cursos, requisitos y planes de estudio, a través de una plataforma centralizada y accesible.

Este documento presenta el desarrollo de un sistema de gestión de planes de estudio diseñado para una institución educativa. Este sistema no solo permite a los usuarios registrar y organizar información crucial sobre escuelas y cursos, sino que también integra funcionalidades avanzadas para establecer y gestionar requisitos y planes de estudio detallados. Además, incluye la capacidad de generar reportes en formato PDF de los planes creados, lo que facilita la revisión y distribución de la información.

A lo largo del desarrollo de este proyecto, el equipo ha adquirido valiosas lecciones relacionadas con la organización del código, el uso de herramientas de documentación, la creación de interfaces gráficas intuitivas y la gestión eficiente de entradas y salidas de datos. Estas experiencias no solo han contribuido a la creación de un sistema robusto y funcional, sino que también han mejorado las competencias técnicas y organizativas del equipo de desarrollo.

ESTRATEGIA METODOLÓGICA

No se definieron roles en específico. Cada clase que se construyó entre los miembros del equipo se hizo basándose en el diagrama de clases ya establecido y las clases ya construidas a principios del proyecto. Las tareas se repartieron según la disponibilidad de los miembros, cada miembro releva al otro en sus tareas pendientes. Se mantuvo la comunicación entre los miembros del grupo mediante un chat grupal de Discord, donde se anuncian avances del proyectos y se comprobó la funcionalidad del código y el manejo de errores.

CRONOGRAMA DE TRABAJO

https://docs.google.com/spreadsheets/d/1wBWPmXNvcbfXwlcmUK_vtpOF36OWs4wu/edit?usp=sharing&oid=115059698561706764585&rtpof=true&sd=true

JUSTIFICACIÓN DE LAS RELACIONES ESTABLECIDAS ENTRE LOS OBJETOS DEL DIAGRAMA

Como se puede observar en la figura 1. Existen diversas relaciones entre las clases creadas por el equipo.

Relaciones de Agregación:

Gestor-Escuela: Las escuelas pueden existir independientemente del gestor, por lo que se establece una relación de

agregación. El gestor utiliza las escuelas, pero éstas no dependen de él.

Escuela-Curso: Una escuela tiene múltiples cursos, pero los cursos pueden existir independientemente de estar asociados a una escuela específica. Por lo tanto, se establece una relación de agregación.

Plan-Curso: Un plan de estudios incluye múltiples cursos, pero los cursos pueden existir sin ser parte de un plan particular. Esto refleja la flexibilidad de los cursos para ser reutilizados en diferentes planes.

Relación de Composición:

Escuela-Plan: Un plan de estudios es una parte integral de una escuela y no puede existir sin ella. Si se elimina una escuela, también se deben eliminar sus planes. Por lo tanto, se establece una relación de composición.

Relaciones de Asociación:

PDF-Escuela: Un PDF se asocia con una única escuela para representar sus datos, pero una escuela puede estar asociada con varios PDFs. Esto indica una relación donde el PDF necesita conocer la escuela para generar el contenido, pero no es necesario que la escuela sepa los datos del PDF.

PDF-Plan: Misma justificación que la relación PDF-Escuela, donde el PDF se asocia con un plan de estudios específico para generar su contenido.

Gestor-PDF: El gestor depende del PDF, ya que es el gestor quien invoca la generación del PDF. Por lo tanto, se establece una relación de dependencia entre el gestor y el PDF.

DIAGRAMA DE PAQUETES

A continuación, se explica la naturaleza de cada uno de los paquetes:

controlador: Este paquete contiene las clases que actúan como controladoras en la arquitectura MVC. Las clases controladoras son responsables de manejar la lógica de control de la aplicación, procesando las entradas del usuario (desde la vista) y coordinando la interacción con el modelo

de datos. Las clases controladoras también pueden ser responsables de actualizar la vista con los datos relevantes obtenidos del modelo.

modelo: Este paquete contiene las clases que representan el modelo de datos de la aplicación. Las clases del modelo definen la estructura de los datos y la lógica de negocio relacionada con el manejo y manipulación de esos datos. Estas clases encapsulan la lógica y los datos de la aplicación, independientemente de la interfaz de usuario o la lógica de control.

vista: Este paquete contiene las clases que representan la interfaz de usuario de la aplicación. Las clases de la vista son responsables de mostrar los datos del modelo al usuario y recoger las entradas del usuario para enviarlas al controlador. Estas clases normalmente no contienen lógica de negocio y sólo se encargan de la presentación visual de la aplicación.

ANÁLISIS DE RESULTADOS

Actividades	Porcentaje de realización	Justificación
Clase Plan	100%	
Clase Escuela	100%	
Clase Curso	100%	
Clase gestor	100%	
Clase PDF	100%	
Interfaz Gráfica	100%	
Control de entradas	100%	
Control de salidas	100%	

ENLACE AL JAVADOC GENERADO:

[Link a Javadoc Proyecto2](#)

LECCIONES APRENDIDAS DEL EQUIPO:

Gustavo:

1. Organización de códigos mediante paquetes: Aprendí a mantener la organización de los códigos mediante paquetes, lo que facilitó la estructura y la navegación del proyecto.
2. Uso de Javadocs: Aprendí a utilizar Javadocs para documentar los métodos y clases, lo que mejoró la legibilidad y la comprensión del código.
3. Creación de interfaces gráficas: Aprendí a crear interfaces gráficas utilizando Java, lo que nos permitió interactuar con la aplicación de manera visual.
4. Gestión de entradas y salidas: Aprendí a gestionar las entradas y salidas de la aplicación, garantizando la seguridad y la integridad de los datos.
5. Control de errores: Aprendí a controlar errores y excepciones en el código, lo que mejoró la estabilidad y la robustez de la aplicación.

Jose:

1. Uso de diagramas de clases: Aprendí a utilizar diagramas de clases para representar la estructura y la relación entre las clases, lo que facilitó la comprensión y el diseño del proyecto.

2. Implementación de métodos y atributos: Aprendí a implementar métodos y atributos en las clases, lo que nos permitió representar y manipular los datos de manera efectiva.
3. Creación de archivos PDF: Aprendí a crear archivos PDF utilizando Java, lo que nos permitió generar documentos de manera automatizada.
4. Trabajo en equipo: Aprendí a trabajar en equipo, colaborando y comunicándonos efectivamente para lograr los objetivos del proyecto.
5. Gestión del tiempo y priorización de tareas: Aprendí a gestionar el tiempo y priorizar tareas, lo que nos permitió mantener el progreso del proyecto y ajustar los plazos según sea necesario.

Sidney:

1. Análisis de errores y solución de problemas: Aprendí a analizar errores y solucionar problemas, lo que mejoró nuestra capacidad para abordar desafíos técnicos.
2. Uso de herramientas de desarrollo: Aprendí a utilizar herramientas de desarrollo como NetBeans y Git, lo que nos permitió trabajar de manera más eficiente y colaborativa.
3. Creación de tests unitarios: Aprendí a crear tests unitarios para verificar la funcionalidad de los métodos y clases, lo que mejoró la calidad y la confiabilidad del código.
4. Documentación interna: Aprendí a crear documentación interna para la aplicación, lo que facilitó la comprensión y el mantenimiento del proyecto.
5. Aprendizaje de nuevas habilidades: Aprendí nuevas habilidades y tecnologías, como la creación de interfaces gráficas y la implementación

de archivos PDF, lo que nos permitió crecer y mejorar como desarrolladores.

BITÁCORA O DIARIO DE TRABAJO

Gustavo Vargas:

<https://proyecto2gustavovargas.blogspot.com/>

Sidney Salazar:

<https://sisalazarjim21.blogspot.com/>

Jose Salazar:

<https://bitacoraprojectopoo1.blogspot.com/>

BIBLIOGRAFÍA, FUENTES DIGITALES Y APIS UTILIZADAS:

Crear Documento PDF

https://www.youtube.com/watch?v=DezkSACVG_w

<https://pdfbox.apache.org/>

Introducción a Patrones de Diseño

<https://youtu.be/G4h5EnrfzQM>

Patrón MVC

<https://youtu.be/WkrXK-bXmMo>

Introducción al modelado de Software

https://youtu.be/H_HXqW5Cqls

MANUAL DE USUARIO

Al darle click izquierdo al archivo main.java, veremos la opción “run file”, este ejecutara el archivo main.java, el cual mostrará la interfaz del gestor de planes de estudios. En esta interfaz existen 6 opciones disponibles:



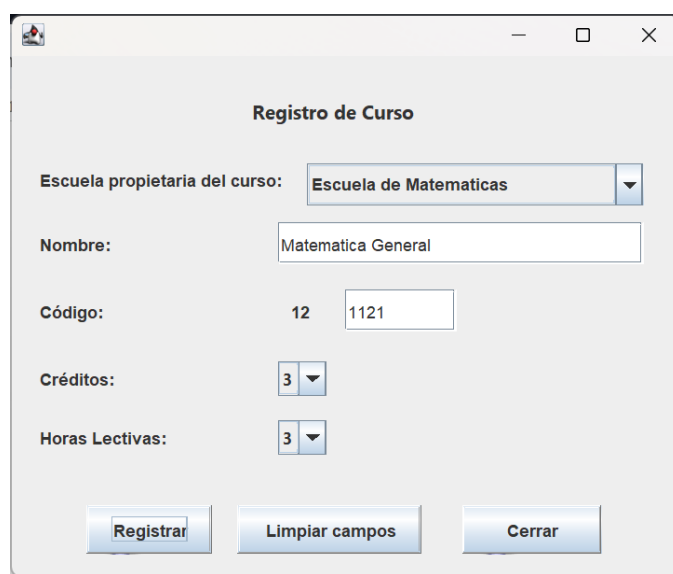
1. Registrar escuela: En esta opción existen dos entradas, en la primera debemos escribir el nombre de la escuela que será propietaria de un curso y en la segunda el código que deseamos asignarle a la escuela. No se debe dejar ninguna entrada vacía y el código que le asignemos a la escuela debe ser de dos dígitos ni debe estar asignado a otra escuela. Cuando cumplamos con todos los requisitos, la escuela será guardada en el sistema.

curso:
curso,
elegir
una

2. Registrar
Al registrar un
debemos
por medio de
combobox la
escuela
propietaria del

curso que registramos, ingresar el nombre del curso e ingresar el

código del curso. No se debe dejar la entrada vacía, el número del curso debe ser de 4 dígitos y no puede estar asignado a otro curso. También se debe elegir la cantidad de créditos y horas lectivas de curso. El mínimo de créditos es entre 0 a 5 y el mínimo de 3 horas lectivas es entre 1 a 5. Al completar todos los campos el curso quedará registrado.



Registro de Curso

Escuela propietaria del curso: Escuela de Matematicas

Nombre: Matematica General

Código: 12 1121

Créditos: 3

Horas Lectivas: 3

Registrar Limpiar campos Cerrar

3. Registrar requisitos: al escoger esta opción, debemos elegir la escuela asociada al curso que queremos asignarle requisitos y correquisitos. Al hacer esto tendremos disponible en una combobox que contendrá los cursos propietarios de la escuela que hayamos elegido. Al elegir un curso tendremos la opción de asignarle requisitos y/o correquisitos y confirmar estos datos por medio de dos botones separados.
4. Crear Plan: Al elegir esta opción, debemos elegir la escuela a la que pertenece el curso que incluiremos en el plan de estudios. Después

Asignar requisitos y correquisitos a un curso

Escuela propietaria del curso: Ing en Computacion

Código del curso: IC1122

<p>Requisitos del curso</p> <p>Código del curso: IC1000</p> <p>Registrar requisito</p>	<p>Correquisitos del curso</p> <p>Código del curso: IC1000</p> <p>Registrar correquisito</p>
---	---

Cerrar

debemos asignarle al plan de estudios un código de 4 dígitos. Debemos definir la vigencia del plan de estudios con el formato DD/MM/YYYY y ninguna de las entradas deben estar vacías. De no escribirse en este formato la entrada será rechazada. Al escoger a la escuela propietaria del plan, se mostrará en una combobox los cursos relacionados a la escuela. Es a elección del usuario en qué bloque quiere matricular un curso. No se pueden matricular dos cursos en el mismo bloque. Cuando hayamos terminado de agregar los cursos al plan de estudios, agregamos el plan por medio de un botón.

5. Generar PDF: Al seleccionar esta opción, debemos elegir la escuela propietaria del plan junto al código del plan de estudios.

Generar Plan en PDF

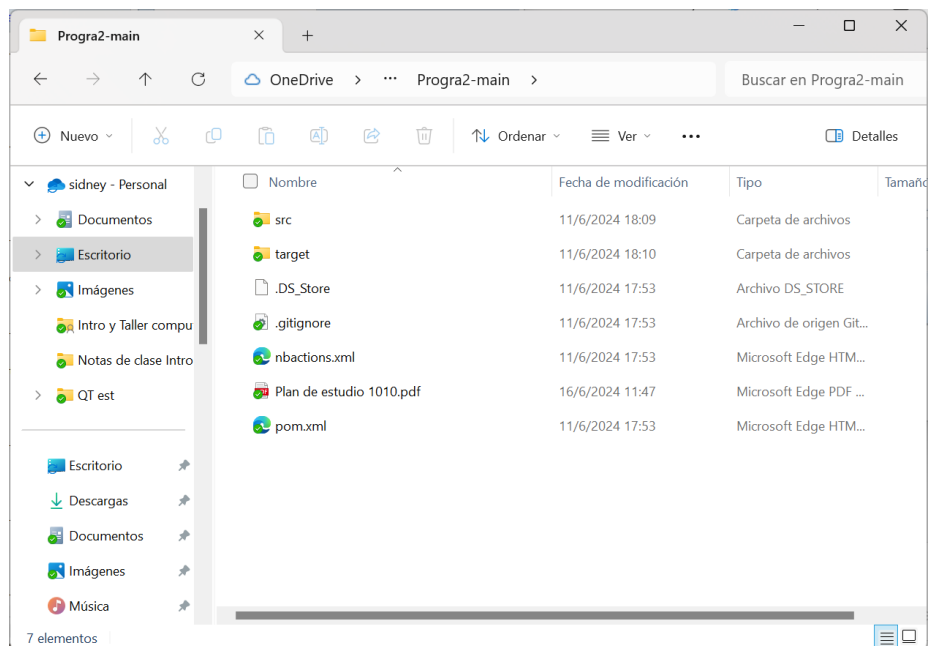
Escuela propietaria del plan: Ing en Computacion

Código del plan de estudios: 1010

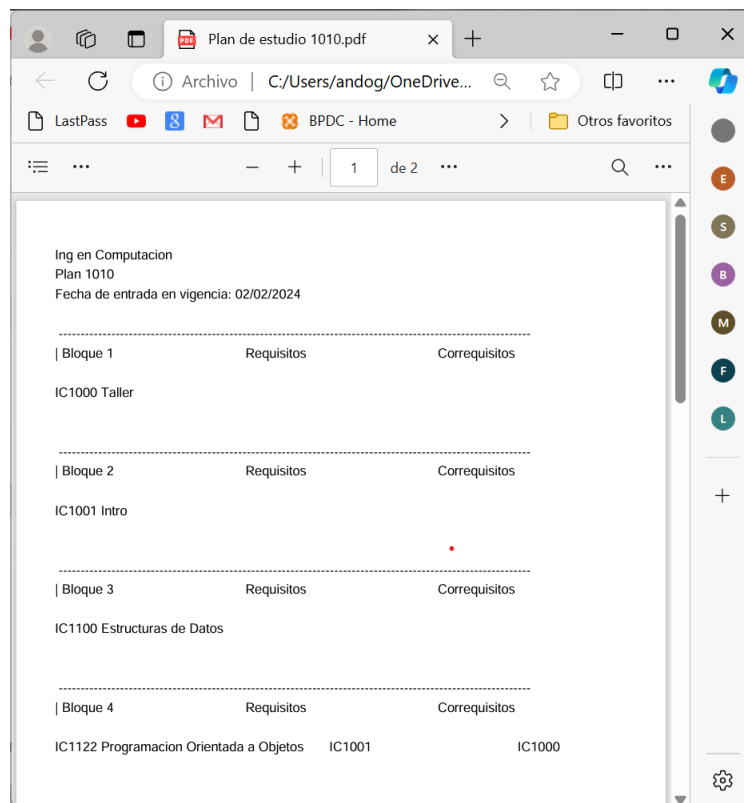
Generar Limpiar

Cerrar

Al presionar el botón para generar el pdf del plan de estudios elegido. Podremos ver el archivo creado en la carpeta del sistema.



Si entramos al documento veremos la información de los cursos estructurada de la siguiente manera:



Requisitos para correr el programa:

versión de JDK: JDK 21 o 22

Librerías:

javax.swing

com.itextpdf

java.util.ArrayList

java.util.Objects

java.util.List

java.util.logging.Level y java.util.logging.Logger

java.io.FileNotFoundException y java.io.FileOutputStream

java.time

java.util.Objects

java.util.List

javax.swing.text.DateFormatter
 javax.swing.JComboBox
 javax.swing.JTextField
 javax.swing.JLabel
 javax.swing.JOptionPane

Anexos

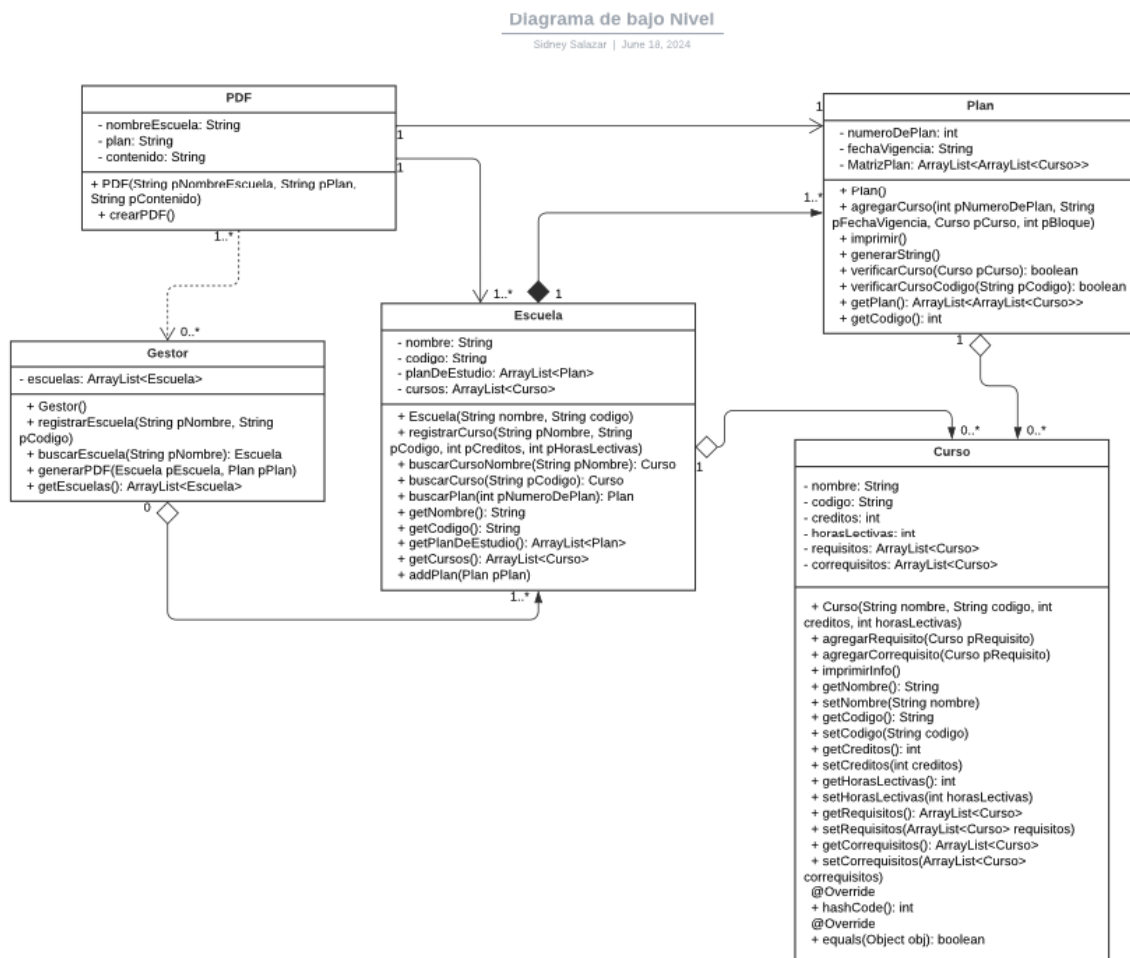


Figura 1. Diagrama de bajo Nivel.

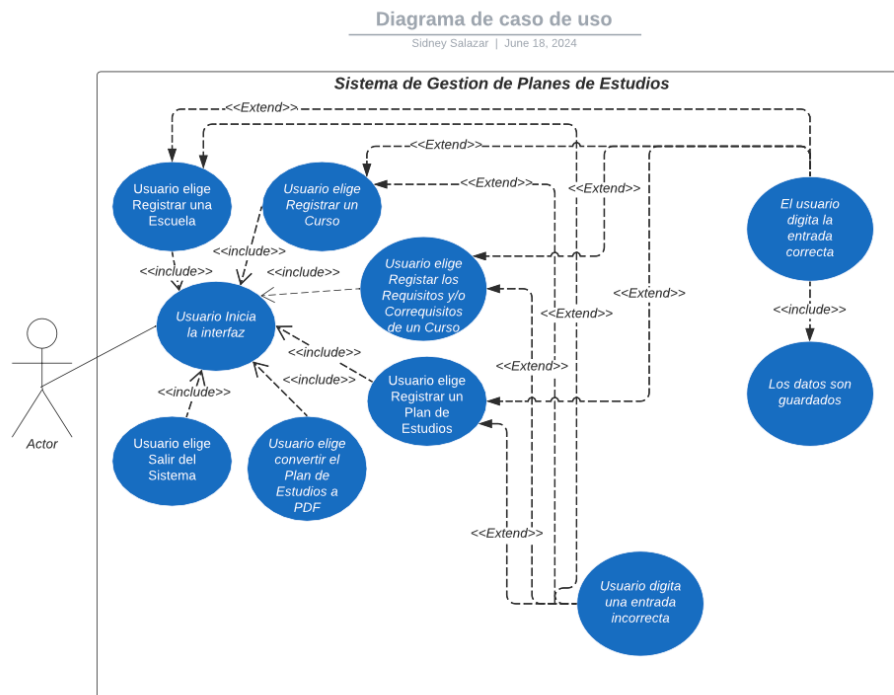


Figura 2. Diagrama de caso de uso.

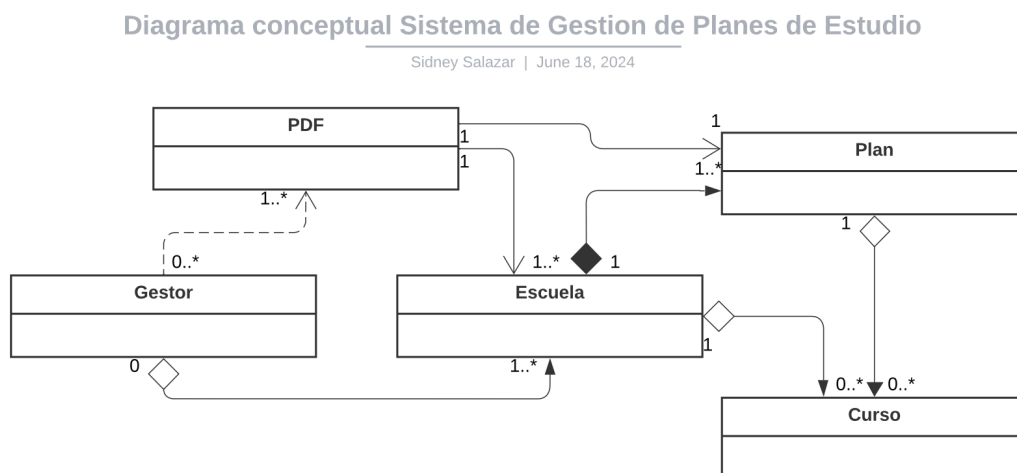


Figura 3. Diagrama Conceptual.

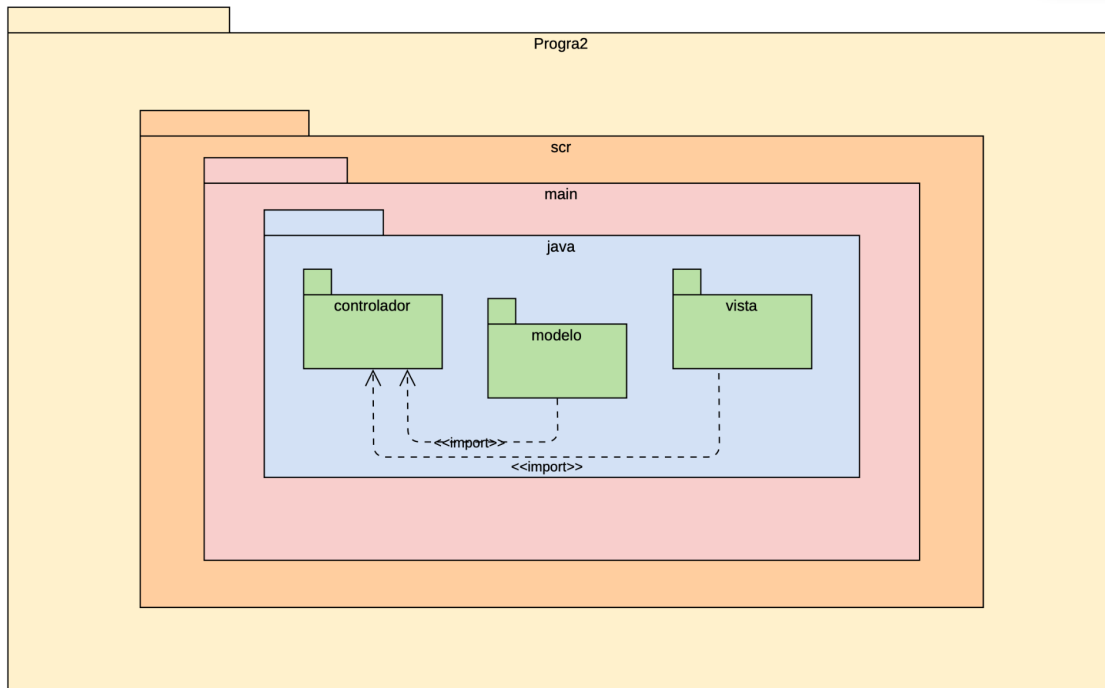


Figura 4. Diagrama de paquetes.

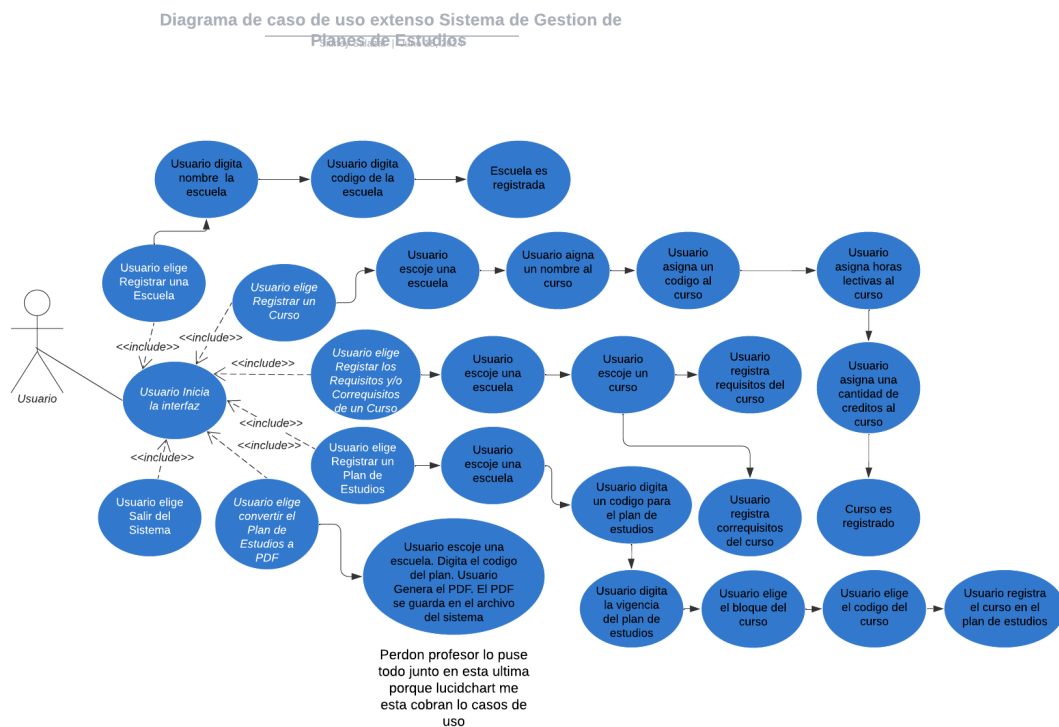


Figura 5. Diagrama de Casos de uso extendidos para TODOS los requerimientos en el documento.