Objetivo general del proyecto

El objetivo general de este proyecto es practicar varias etapas del desarrollo de una aplicación de software, desde el análisis hasta la construcción una aplicación funcional. A través del proyecto, los estudiantes pondrán en práctica todas las habilidades desarrolladas en el curso.

Objetivos específicos del proyecto

Durante el desarrollo de este proyecto se buscará el desarrollo de las siguientes habilidades:

1. Identificar y abstraer entidades en un dominio particular, así como sus características y relaciones.
2. Construir diagramas de clase UML para expresar modelos de dominio y modelos de diseño.
3. Diseñar una aplicación basada en el paradigma orientado a objetos y expresar el diseño, justificando las decisiones importantes que hayan tomado.
4. Implementar un diseño utilizando el lenguaje de programación Java, poniendo énfasis en la implementación de las funcionalidades requeridas.
5. Construir un conjunto de programas sencillos que demuestren que las funcionalidades están correctamente implementadas.

Instrucciones generales

A continuación, encontrará la definición inicial del proyecto, que será la base del trabajo de los 3 proyectos que se construirán durante todo el semestre.

El trabajo del proyecto #2 debe partir de la base del proyecto #1, y el trabajo del proyecto #2 será la base del proyecto #3. Si es necesario, podrán hacerse cambios en cada una de las etapas para mejorar o corregir decisiones que se hayan tomado en las etapas anteriores.

En el proyecto #1 no es necesario que haya una interfaz que le permita al usuario interactuar con la aplicación. Sin embargo, deben construirse programas que demuestren que la lógica de la aplicación y las historias de usuario quedaron debidamente implementadas. Estos programas (o consolas) deben solicitarle al usuario la menor cantidad posible de información para correr (por ejemplo, el nombre de un archivo para cargar los datos, o el nombre de una funcionalidad que se quiera demostrar)[1]. Estos programas también tienen que mostrar, a través de la consola, el estado de la aplicación de una forma que sea fácil de entender y que permita apreciar que la funcionalidad es correcta.

En el proyecto #2, uno de los requerimientos será implementar un mecanismo de interacción con el usuario basada en consola. En este proyecto sí será necesario implementar una por una las historias de usuario y permitir toda la interacción con los usuarios para que entreguen la información que se requiere. El motivo de esta

[1] Estos programas también podrían verse como precursores de pruebas automatizadas. Si usted quiere, puede implementar pruebas usando Junit, pero eso se estudiará más adelante en el curso.

separación entre los requerimientos del proyecto #1 y el proyecto #2 es que en el primero el foco debería estar en el diseño y la implementación de la lógica de la aplicación y no en la lógica de la interacción con los usuarios, validación de las entradas y presentación de los resultados.

El proyecto debe desarrollarse en **grupos que podrán cambiar para el proyecto #2, pero no para el proyecto #3.**

Contexto del proyecto

En este proyecto, desarrollará un **Learning Path Recommendation System** que permite a profesores crear Learning Paths (caminos de aprendizaje) que los estudiantes pueden seguir para explorar sus intereses y desarrollar habilidades a través de actividades estructuradas. La plataforma cuenta con registro de usuarios y autenticación, permitiendo a los estudiantes y profesores acceder a experiencias de aprendizaje personalizadas. Los profesores pueden crear y gestionar Learning Paths personalizadas, mientras que los estudiantes reciben recomendaciones basadas en sus intereses. El sistema rastrea las actividades y el progreso de los estudiantes, proporcionando información sobre sus logros y avances. En el futuro, se integrará con plataformas educativas (LMS – Learning Management Systems, como Bloque Neón), ofreciendo notificaciones y recordatorios para actividades próximas. Un sistema de feedback y rating ayuda a mejorar los Learning Paths y proporciona a los profesores información para seguir mejorando su contenido.

El núcleo del sistema son los Learning Paths. Un Learning Path es creado y curado por un profesor y tiene información básica como un título, una descripción general de su contenido y objetivos, nivel de dificultad, duración en minutos (calculada) y rating. También posee metadatos como la fecha de creación, fecha de modificación y versión.

Un Learning Path está compuesto por una secuencia de actividades que los estudiantes deben realizar, las cuales tienen diferentes características dependiendo del tipo específico. No obstante, cada actividad tiene una descripción, un objetivo, un nivel de dificultad, una duración esperada y actividades previas sugeridas: si un estudiante intenta realizar una actividad sin completar sus pre-requisitos, el sistema debe permitirlo, pero con una advertencia explicando por qué no es una buena idea. Las actividades también tienen una fecha límite basada en la actividad anterior. Por ejemplo, un quiz sobre una lectura podría tener una fecha límite que fuera 1 hora después de terminar la lectura, para asegurar que el estudiante complete el quiz justo después de completar la lectura. Note que estas fechas son solo sugerencias para los estudiantes. Finalmente, las actividades en un Learning Path pueden tener actividades de seguimiento recomendadas, las cuales pueden depender del resultado de la actividad. Por ejemplo, si un quiz fue exitoso, realizar la lectura siguiente, pero si fue insatisfactorio, volver a la lectura. Nota que cada actividad debe tener un resultado, aunque para la mayoría de ellas el resultado será solo “Completada”.

Cualquier profesor en el sistema puede crear actividades y, al editar un Learning Path, puede incluir actividades existentes. Las actividades pueden estar marcadas como obligatorias u opcionales. Tanto los profesores como los estudiantes pueden dejar reseñas en las actividades, así como un rating. Cuando un profesor está creando un Learning Path o editando uno existente, tiene acceso a esas reseñas para seleccionar las actividades más apropiadas. Solo el creador de una actividad puede editarla, pero otros profesores pueden clonar una actividad y editar su copia.

Cuando un estudiante se inscribe en un Learning Path, puede ver la estructura del path y la descripción de cada actividad. Luego, puede decidir qué actividad comenzar, no necesariamente la primera. Solo se puede iniciar una actividad a la vez.

El sistema rastreará el progreso de los estudiantes que se inscriben en un Learning Path, el cual se calcula simplemente como un porcentaje de las actividades obligatorias que fueron completadas exitosamente. El sistema también rastrea detalles como cuándo se inició un Learning Path, cuándo se completó, tiempo dedicado por actividad, tasas de éxito y fracaso, y cuándo fueron completadas. En el futuro, realizaremos análisis con estos

datos para responder preguntas como “¿Qué tan exitosos son los estudiantes en quices cuando los realizan tarde en la noche versus durante el día?”.

El sistema soportará diferentes tipos de actividades. Por ahora, sus tipos serán bastante limitados, pero en el futuro se añadirán nuevos tipos. El primer tipo de actividad es revisar un recurso educativo, que puede ser un video, un sitio web, un libro o un recurso adjunto como un pdf. El resultado de estas actividades siempre es exitoso al completarse (es decir, el estudiante afirma que completó la actividad y el sistema lo cree).

Otro tipo de actividad es una tarea. En este caso, el estudiante debe realizar alguna actividad y luego registrar en el sistema que la tarea fue completada y entregada a través de otro medio (por ejemplo, el LMS, por correo, etc.). La actividad estará en el estado “enviada” hasta que el profesor responsable del Learning Path la marque como exitosa o no exitosa. En el futuro, cuando el sistema se conecte al LMS, esto será automático, pero por ahora los profesores desempeñarán el rol de integradores.

El tercer tipo de actividad es un quiz, que tendrá una serie de preguntas de opción múltiple. Cada pregunta debe tener 4 opciones, solo una debe ser correcta, y cada opción debe tener una explicación que se le da al estudiante después de completar el quiz. La calificación mínima para aprobar se establece cuando se crea el quiz.

También es posible tener exámenes, que tienen preguntas abiertas y, de manera similar a las tareas, permanecen en el estado “enviada” hasta que el profesor califica las respuestas de los estudiantes.

El último tipo de actividad es una encuesta donde se le hacen preguntas abiertas al estudiante. Las encuestas se completan después de que el estudiante las envía.

Aspectos técnicos y restricciones

Toda la información debe ser persistente. La información debe almacenarse en archivos (pueden ser planos, en el formato que ustedes definan, o binarios), dentro de una carpeta y se puede suponer que sólo la aplicación va a escribir y leer de esa carpeta (ningún usuario malicioso va a modificar los archivos que ahí se encuentren sin utilizar la aplicación). La carpeta no puede ser la misma carpeta donde se encuentre el código fuente de la aplicación.

La persistencia no necesariamente debe hacerse en un solo archivo: diseñe con cuidado cuántos archivos habrá y cómo van a estar estructurados.

Todos los usuarios del sistema deben tener un login y un password.

La aplicación debe estar hecha en Java.

El hecho de que en esta entrega no se tenga que construir una interfaz para los usuarios (sólo programas que demuestren que la lógica quedó correctamente implementada), no significa que no deban estar implementadas todas las funcionalidades necesarias para que en el proyecto #2 se puedan completar las historias de usuario.

No es necesario que se soporten funcionalidades que no hayan sido mencionadas, pero se pueden incluir funcionalidades adicionales si facilitan el trabajo.

Ejemplo de Learning Path

El siguiente es un ejemplo de un posible Learning Path, con varias actividades y con una actividad para la cual se presenta el contenido completo.

Nivel de Python 3 - Ruta de aprendizaje

1. Introducción a los bucles

- **Tipo**: Clase
- **Objetivo**: Comprender el concepto y la sintaxis de los bucles en Python.

• **Nivel**: Principiante

• **Tiempo estimado**: 30 minutos

2. Uso de bucles for

• **Tipo**: Tarea

• **Objetivo**: Aprender a usar los bucles for para iterar sobre listas y rangos.

• **Descripción**: En esta actividad, explorarás cómo usar los bucles for en Python para iterar sobre diferentes secuencias como listas y rangos. Al final de esta actividad, serás capaz de escribir bucles for para realizar tareas repetitivas de manera eficiente.

• **Contenido:**

1. **Introducción a los bucles for**
   - **Lee la siguiente explicación**:
     - Un bucle **for** en Python se usa para iterar sobre una secuencia (como una lista, tupla, diccionario, conjunto o cadena).
     - La sintaxis de un bucle **for** es:

```
for variable in sequence:
    # bloque de código a ejecutar
```

2. **Ejemplo: Iterando sobre una lista**
   - **Código de ejemplo**:

```
fruits = ["apple", "banana", "cherry"]
for fruit in fruits:
    print(fruit)
```

   - **Explicación**: Este código imprimirá cada fruta en la lista una por una.

3. **Ejercicio 1: Iterando sobre una lista**
   - **Tarea**: Crea una lista de tus animales favoritos y escribe un bucle **for** para imprimir cada animal.
   - **Pista**: Usa el ejemplo anterior como referencia.
   - **Salida esperada**:

```
dogcatelephant
```

4. **Ejemplo: Usando range()** en bucles **for**
   - **Código de ejemplo**:

```
for i in range(5):
    print(i)
```

   - **Explicación**: Este código imprimirá números del 0 al 4.

5. **Ejercicio 2: Usando range()**
   - **Tarea**: Escribe un bucle **for** para imprimir los números del 1 al 10.
   - **Pista**: Ajusta la función **range()** para que comience en 1 y termine en 10.
   - **Salida esperada**:

```
1
2
3
4
5
6
7
8
9
10
```

6. **Ejemplo: Iterando sobre una cadena**
   - **Código de ejemplo**:

```
for letter in "Python":
    print(letter)
```

   - **Explicación**: Este código imprimirá cada letra de la cadena “Python”.

7. **Ejercicio 3: Iterando sobre una cadena**
   - **Tarea**: Escribe un bucle **for** para imprimir cada carácter de tu nombre.
   - **Pista**: Reemplaza “Python” con tu nombre en el ejemplo anterior.
   - **Salida esperada**:

```
J
o
h
n
```

8. **Ejercicio 4: Suma de números en una lista**
   - **Tarea**: Crea una lista de números y escribe un bucle **for** para calcular la suma de esos números.
   - **Pista**: Inicializa una variable **total** en 0 antes del bucle y suma cada número a **total** dentro del bucle.
   - **Salida esperada**:

```
Suma de los números: 15
```

   - **Código de ejemplo**:

```
python
numbers = [1, 2, 3, 4, 5]
total = for number in numbers:
    total += number
print("Suma de los números:", total)
```

9. **Reflexión y resumen**
   - **Tarea**: Reflexiona sobre lo que has aprendido acerca de los bucles **for** .
   - **Pregunta**: ¿Cómo pueden ayudar los bucles **for** a automatizar tareas repetitivas en la programación?

10. **Cuestionario sobre los bucles for** (a completar en una actividad posterior)
    - **Tarea**: Pon a prueba tu comprensión de los bucles **for** con un breve cuestionario.

- **Nivel**: Principiante
- **Tiempo estimado**: 45 minutos

3. Uso de bucles while

- **Tipo**: Clase
- **Objetivo**: Entender cómo usar los bucles while para repetir acciones hasta que se cumpla una condición.
- **Nivel**: Principiante
- **Tiempo estimado**: 30 minutos

4. Cuestionario sobre la sintaxis y los casos de uso de los bucles

- **Tipo**: Cuestionario
- **Objetivo**: Evaluar la competencia de los estudiantes en la escritura de bucles.
- **Nivel**: Intermedio
- **Tiempo estimado**: 60 minutos

5. Sentencias de control y bucles

- **Tipo**: Tarea
- **Objetivo**: Practicar sentencias de control y bucles.
- **Nivel**: Intermedio
- **Tiempo estimado**: 60 minutos

6. Iteración sobre cadenas y diccionarios

- **Tipo**: Clase
- **Objetivo**: Usar bucles para iterar sobre cadenas y diccionarios.
- **Nivel**: Intermedio
- **Tiempo estimado**: 40 minutos

7. Ejercicios prácticos con bucles

- **Tipo**: Tarea
- **Objetivo**: Resolver problemas prácticos usando bucles (por ejemplo, encontrar la suma de números, invertir una cadena).
- **Nivel**: Intermedio
- **Tiempo estimado**: 90 minutos

8. Bucles anidados

- **Tipo**: Clase
- **Objetivo**: Entender cómo usar bucles anidados para resolver problemas complejos.

- **Nivel**: Avanzado
- **Tiempo estimado**: 30 minutos

9. Optimización de bucles

- **Tipo**: Tarea
- **Objetivo**: Aprender técnicas para optimizar el rendimiento de los bucles.
- **Nivel**: Avanzado
- **Tiempo estimado**: 60 minutos

10. Proyecto final: Construcción de un juego simple

- **Tipo**: Tarea
- **Objetivo**: Aplicar todos los conceptos aprendidos para construir un juego de texto simple que utilice bucles.
- **Nivel**: Avanzado
- **Tiempo estimado**: 120 minutos
- **Evaluación**: Presentación y entrega del proyecto

11. Examen sobre bucles

- **Tipo**: Examen
- **Objetivo**: Evaluar la competencia y comprensión de los bucles.
- **Nivel**: Avanzado
- **Tiempo estimado**: 120 minutos

Entrega 1: Análisis del proyecto

La primera etapa del proyecto consiste en realizar el análisis del sistema que debe construir.

Actividades

1. Construya un modelo de dominio (diagrama de clases) a partir de la información del caso:
   a. Identifique las entidades que aparecen dentro del caso.
   b. Identifique las características (atributos) de esas entidades.
   c. Establezca las relaciones entre las entidades, incluyendo asociaciones y relaciones de herencia.
2. Construya un documento de análisis para su proyecto. El documento debe incluir:
   a. El modelo de dominio que construyó en el paso anterior.
   b. Una descripción con las restricciones del proyecto.
   c. Una descripción de lo que demostrará cada uno de los programas de prueba.

Entrega

1. El proyecto debe entregarse en una carpeta dentro del repositorio GIT del grupo con el nombre "**Proyecto 1"**. Dentro de esta carpeta debe existir una carpeta con el nombre "**Entrega 1**" donde deben quedar todos los

elementos correspondientes a esta entrega, incluyendo tanto los archivos fuente de los diagramas como imágenes que se puedan leer con facilidad.

2. Entregue un enlace al repositorio a través de Bloque Neón en la actividad designada como "**Proyecto 1 - Entrega 1**".

Entrega 2: Diseño e implementación

Teniendo en cuenta el análisis realizado en la primera entrega del proyecto, para la segunda entrega debe realizar el diseño detallado del sistema y construir la implementación del sistema.

Actividades

1. Realice el diseño y construya un documento de diseño donde presente el diseño, haciendo especial énfasis en justificar las decisiones clave que haya tomado. El documento debe incluir, como mínimo, los siguientes elementos:
    a. Un diagrama de clases de diseño que incluya todas las clases, incluyendo sus las relaciones, atributos y métodos.
    b. Un diagrama de clases de alto nivel, que incluya todas las clases y sus relaciones, pero no todos los métodos ni atributos. Este diagrama facilitará entender las relaciones entre clases.
    c. Diagramas de secuencia para las funcionalidades que usted considere críticas:

    Estos elementos NO son los únicos que debe incluir su documento: piense en qué otros diagramas, descripciones y justificaciones pueden ser necesarias para poder entender totalmente el diseño. Si su documento incluye únicamente los tres elementos mencionados antes, con seguridad será considerado insuficiente.

2. Implemente el sistema que diseñó. Tenga en cuenta que los detalles de la implementación deben ser coherentes tanto con el modelo de clases, como con los diagramas de secuencia.
3. Asegúrese de que la documentación del diseño sea consistente con la implementación.

Entrega

1. Dentro de la carpeta del proyecto debe crear una carpeta con el nombre "**Entrega 2**" donde deben quedar todos los elementos correspondientes a esta entrega, incluyendo los archivos fuente de los diagramas, el documento de diseño, y el proyecto Eclipse con instrucciones para su ejecución. Incluya también archivos de prueba para poder correr las aplicaciones y tener datos con los que se pueda probar con facilidad.
2. Entregue un enlace al repositorio a través de Bloque Neón en la actividad designada como "**Proyecto 1 - Entrega 2**".