Universidad Veracruzana

Facultad de Contaduría y Administración

DISEÑO PRELIMINAR DE LA

APLICACIÓN ORIENTADO A OBJETOS

Ingeniería de Software

Principios de Construcción de Software

Equipo:

Luis Ángel Barrientos Pérez

Jose Ángel Rincón Martínez

Jaime Antonio Hernández Cabrera

Gabriel Reyes Cruz

Carlos Antonio Gallegos Palencia

25 de junio de 2020 **Coatzacoalcos, Ver., México**

Diseño preliminar del programa

En el primer diseño preliminar del programa se ha optado por el lenguaje de modelado UML. En el siguiente diagrama se muestran las clases propuestas para dar solución y visualizar de una mejor manera el problema propuesto de un horario de clases.

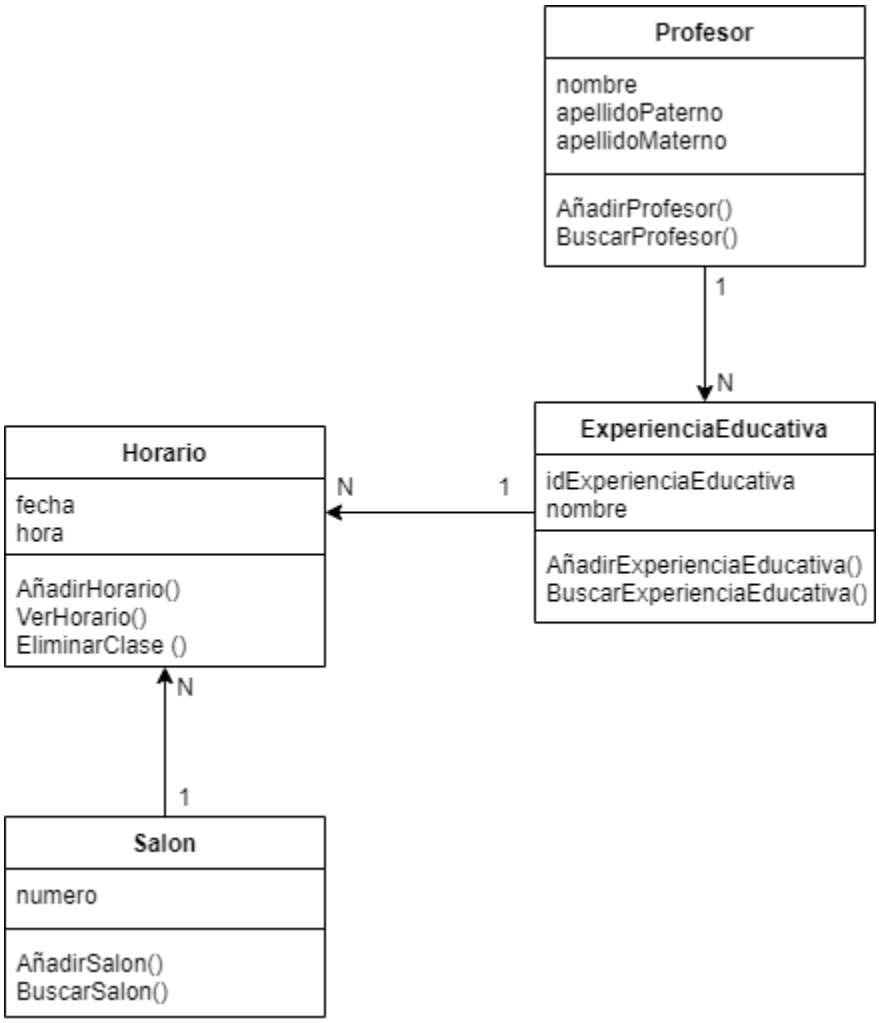


Diagrama UML del programa de horario de clases (Primera versión)

Después de comenzar a desarrollar el proyecto se han modificado algunas clases, a continuación, se muestra el diagrama de clases actual.

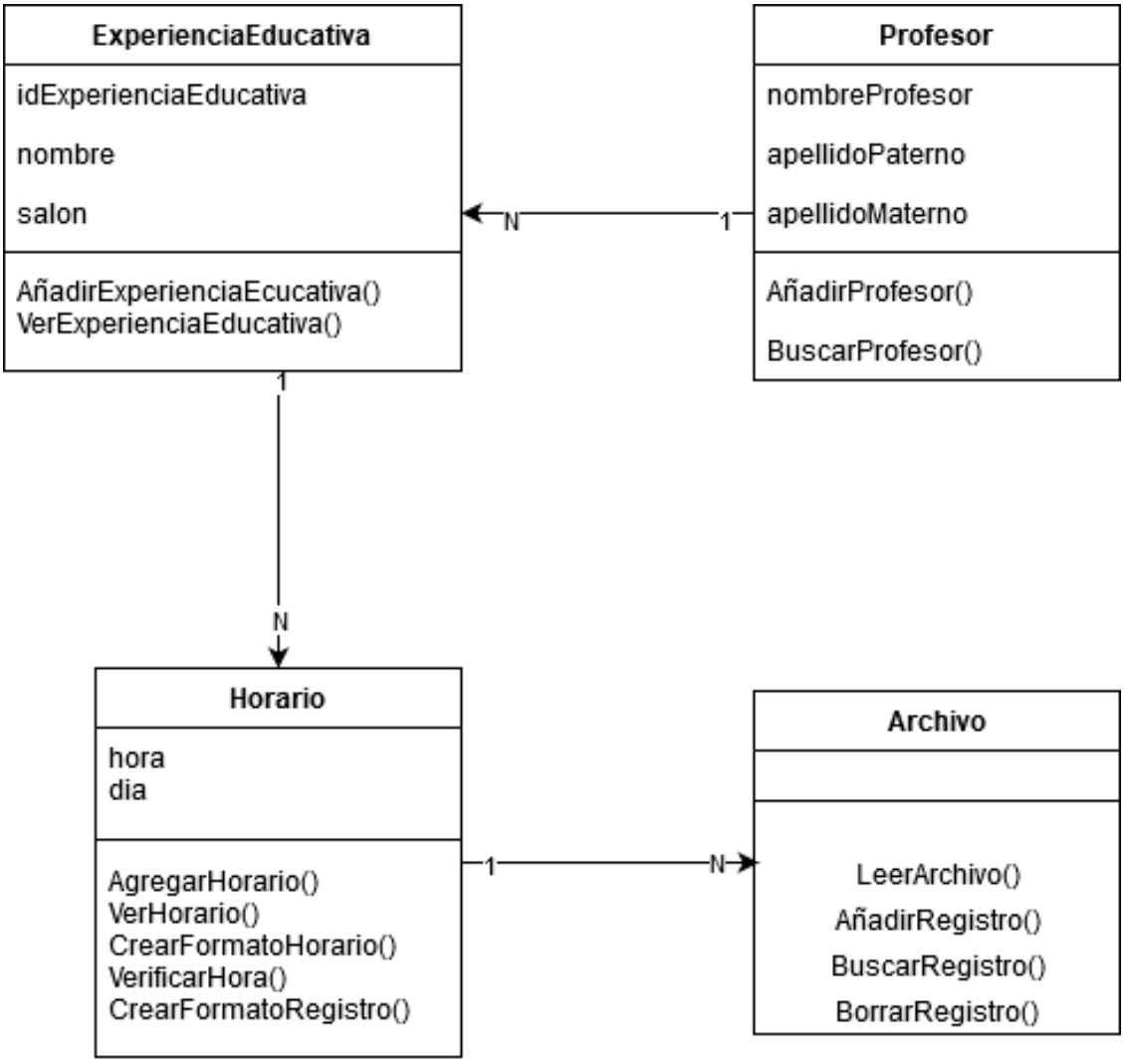


Diagrama UML del programa de horario de clases (Segunda versión)

En esta segunda versión del diagrama UML, se ha eliminado la clase salón, salón ha pasado a ser un atributo de la clase ExperienciaEducativa, además se ha agregado la clase Archivo para guardar el registro de los datos del horario.

Para esta fase se ha tomado en cuenta la heurística que consiste en **encontrar objetos del mundo real.**

Diseño preliminar de la aplicación orientado a objetos.

Para realizar el diseño de nuestra aplicación se realizó un diagrama UML donde especificamos las diferentes clases que identificamos junto a sus atributos y métodos, también la relaciones que existen entre las diferentes clases identificadas.

Identificar los objetos y sus atributos (métodos y datos).

Para empezar en el diagrama realizado se han propuesto cuatro clases que se han identificado en el análisis previo del problema, probablemente durante la fase de codificación se dé la necesidad de modificar algún elemento del diagrama. Las clases identificadas son: Profesor, ExperienciaEducativa, Horario y Archivo.

- **Clase Profesor:** De la clase "Profesor" se han identificado tres atributos: nombre, apellidoPaterno y apellidoMaterno, que son necesarios para poder identificar a un profesor.

Los métodos identificados son dos: AñadirProfesor y BuscarProfesor.

AñadirProfesor: método utilizado para agregar un profesor a nuestro programa, el método recibe como parámetros el nombre, apellido paterno y materno del profesor.

BuscarProfesor: método utilizado para buscar algún profesor registrado en nuestro programa, el método recibe como parámetro el nombre completo del maestro.

- **Clase ExperienciaEducativa:** De la clase "ExperienciaEducativa" se han identificado tres atributos: idExperienciaEducativa, nombre, y se ha agregado un nuevo atributo, el salón. Para poder gestionar las Experiencias Educativas con su respectivo salón donde será impartida.

Los métodos identificados son dos: AñadirExperienciaEducativa y BuscarExperienciaEducativa.

AñadirExperienciaEducativa: método utilizado para agregar una experiencia educativa a nuestro programa, el método recibe como parámetro el nombre de la experiencia educativa.

BuscarExperieciaEducativa: método utilizado para buscar una experiencia educativa registrado en nuestro programa, el método recibe como parámetro el nombre de la experiencia educativa o el id de la materia.

- **Clase Horario:** De la clase "Horario" se habían identificado dos atributos: fecha y hora, ahora el atributo fecha se cambiado por día, estos con el objetivo de asignar días y horas a las Experiencias Educativas.

Los métodos identificados son dos: AgregarHorario, VerHorario y se han agregado nuevos métodos, CrearFormatoHorario, VerificarHora y CrearFormatoRegistro

AgregarHorario: Este método permitirá agregar un horario al archivo de texto tomando los métodos de la clase Archivo.

VerHorario: Este método permitirá visualizar la información del horario de clases que se encuentra almacenado en la clase Archivo.

CrearFormatoHorario: Este método permitirá dar formato a la información para que sea visualizado en un formato de horario.

VerificarHora: Este método permitirá validar la disponibilidad de los horarios para que al asignar alguna experiencia educativa no interfiera con otra.

CrearFormatoRegistro: Este método permitirá acomodar los datos de un registro al ser ingresados al archivo de texto.

- **Clase Archivo:** Esta clase se ha agregado para guardar y gestionar los registros del horario de clases en un archivo de texto.

Se ocupará la clase archivo para manejar los métodos de la librería **java.util.io**, para almacenar la información en un archivo de texto plano.

Los métodos identificados son cuatro: LeerArchivo, AñadirRegistro, BuscarRegistro y BorrarRegistro.

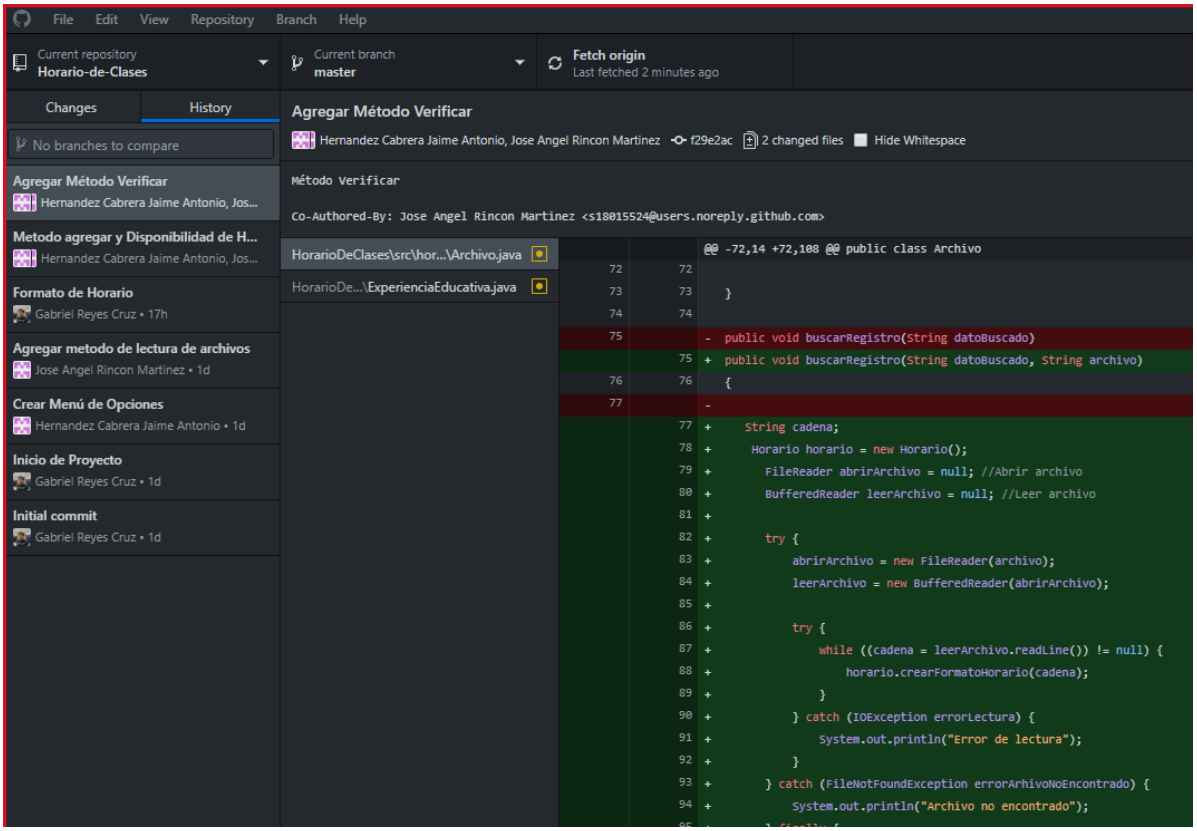
LeerArchivo: Este método servirá para leer el archivo de texto y mostrarlo en línea de comando.

AñadirRegistro: Este método servirá para añadir un registro de horario de clase al archivo de texto.

BuscarRegistro: Este método permitiría realizar la búsqueda de un registro en el archivo de texto.

BorrarRegistro: Este método permitirá borrar un registro en específico del archivo de texto.

Control de versiones. Antes de comenzar con el desarrollo del proyecto, se ha creado un repositorio en GitHub para llevar un control de versiones del programa, para que cada uno de los participantes del equipo pueda compartir de manera eficiente sus contribuciones al proyecto y los demás puedan descargar esos cambios realizados.



Control de versiones (GitHub)

Determine qué se puede hacer con cada objeto.

Es importante identificar qué acciones se podrán realizar con cada una de las clases, esto permite al desarrollador tener un mejor panorama acerca de los métodos que se estarán utilizando en cada una de ellas y la funcionalidad que traerá con esta. Una mejor representación podemos observar en el diagrama UML del diagrama de horario de clases

Determine qué puede hacer cada objeto a otros objetos.

Se debe de determinar de qué forma van a interactuar las diferentes clases de nuestro código, si se utiliza la herencia, cuáles son los métodos y atributos de la clase padre heredará a la clase hijo.

Definición y aplicación de un estándar o convención de codificación.

Para este proyecto se contempla trabajar con el lenguaje de programación JAVA, por lo que nos vamos a regir por los estándares de codificación del lenguaje Java presentados en *Java Language Specification*, de Sun Microsystems, Inc.

Comentarios en código.

Comentar el código es reconocido como una buena práctica, por lo que al inicio de nuestro código estaremos dando una breve descripción de lo que estará realizando el programa. Los comentarios deben contener sólo información que es relevante para la lectura y entendimiento del programa, así como no debe contar con caracteres especiales. Los programas pueden tener cuatro estilos de comentarios de implementación: de bloque, de una línea, de remolque, y de fin de línea. De estos tipos de comentarios que se pueden utilizan al momento de escribir código se buscará ser específico garantizando el entendimiento acerca de la función de los métodos o clases, pero a su vez buscando no abusar de ellas ya que la frecuencia de comentarios a veces refleja una pobre calidad del código.

Uso de programación defensiva.

En cuanto a la programación defensiva se verificarán los datos de los parámetros de entrada de las rutinas del programa, así como también se tendrán que validar los datos de entrada del usuario para que en realidad se cumpla los que se está pidiendo, en este caso si al usuario se le pide el nombre de un profesor se verificará que el nombre sea una cadena de texto, posteriormente cuando se solicite un número de salón se verificará que el valor que el usuario ingreso sea un número.

Identificar áreas que pueden cambiar. Uno de los aspectos importante que podemos observar y deseamos tomar en cuenta es identificar las áreas donde nuestro sistema puede cambiar, nuestro objetivo al dividir en diferentes clases es poder aislar áreas inestables que pueden estar en constante cambio para que esto no afecte a todo el código, algunos de los pasos que buscamos tener en cuenta son:

- **Identifique los elementos que parecen cambiar:** Uno de los elementos que se encuentran en constante tiempo podemos ubicarlo en la clase ExperienciaEducativa, el nombre de la experiencia puede cambiar dependiendo de la actualización de los programas educativos, así como los identificadores para poder lograr la relación entre el nombre de la experiencia educativa y el identificador. Otras de las clases que se pueden encontrar en constante cambio son el Salón y Horario, la información de estas clases puede cambiar constantemente dependiendo de la institución para la cual el sistema estará funcionando, los horarios de clases son diferentes los de las escuelas públicas a los de instituciones privadas que el rol de salones horarios y materias se encuentran en constante cambio.

- **Artículos separados que pueden cambiar:** Las clases que se estarán utilizando para asegurar el funcionamiento correcto del sistema son: profesor con los atributos (nombre, apellido paterno, apellido materno), ExperienciaEducativa con los siguientes atributos (IdExperienciaEducativa y nombre), Horario con los atributos (fecha y hora) y Salón con el atributo (número), de esta manera se busca asegurar el correcto funcionamiento y al momento de cambiar algún elemento no afecte a todo el sistema.
- **Aísle los elementos que parecen cambiar:** Abstraer un objeto del mundo real y dividirlo en distintas clases y métodos cómo se estará llevando a cabo nos garantizara el correcto funcionamiento del sistema, no siempre todos los datos de un registro se deben cambiar cuando se necesite modificar la información por esta razón tomamos la decisión de dividir el horario de clases en módulos.