Universidad Tecnológica de la Habana “José Antonio Echevarría”

Ingeniería Informática

Base de Datos

Proyecto # 1: Control de la docencia en un centro estudiantil.

Autores:

Daniel Fernández Lorenzo dfernandezl@ceis.cujae.edu.cu

Ernesto Leandro Fariña Benítez efarinab@ceis.cujae.edu.cu

Damián Aramis Ordoñez Rieumont dordonezr@ceis.cuaje.edu.cu

Grupo: 22

Noviembre

2021

Resumen

El presente informe describe la propuesta de proyecto presentada como solución de la problemática #1 sobre el Control de la Docencia en un Centro Estudiantil. El enunciado del proyecto plantea la necesidad de crear un sistema que permita manejar y administrar las evaluaciones docentes de los estudiantes así como la matrícula de los mismos en la facultad. Además también se exige la presencia de determinados reportes parametrizables que incluyen listar los estudiantes por grupos, sus promedios, escalafón, localizar datos de cursos previos entre otros que se detallaran más adelante. Para dar solución a esta problemática se concibió el desarrollo de una aplicación de escritorio desarrollada en lenguaje Java y con interfaz gráfica de usuario para facilitar el uso de la aplicación a usuarios que se sientan más cómodos con las mismas. Dicha aplicación se conecta a una base de datos relacional gestionada por PostgreSQL. En la base de datos se encuentran las tablas que modelan entidades de la vida real como son los estudiantes, los grupos y las asignaturas. Dichas tablas se relacionan entre ellas para simular la interrelación de dichas entidades en el mundo real. Por ejemplo, un estudiante pertenece a un grupo en un curso determinado. El desarrollo de este sistema es una solución simple que da solución al problema de digitalización y administración de los datos de la situación planteada.

Introducción

En cualquier centro de estudios es necesario un sistema que administre correctamente sus datos. Enfocado en esta idea, de una manera general y simple, es realizado un proyecto informático capaz controlar y automatizar eficientemente los principales datos básicos en cuanto a los resultados docentes alcanzados en dichos centros estudiantiles.

Para el desarrollo de dicho sistema, el proyecto tiene como arquitectura establecida la de cliente servidor y, además, se ha implementado bajo el paradigma orientado a objetos por medio un lenguaje reconocido mundialmente (Java). Por medio de este lenguaje se implementarán las interfaces de usuario con gran variedad y comodidad de estilo utilizando el paquete javafx. Para la realización de estas interfaces se trabajó cumpliendo el patrón de Modelo Vista Controlador. En consecuencia a esto, se ha organizado en paquetes donde se organiza el modelado lógico del programa, el modelado visual y otros paquetes de utilidades que fueron surgiendo a medida del proceso de codificación.

En cuanto a la elaboración de la base de datos, se ha utilizado como sistema de gestión de datos el software PostgreSQL. En dicho software se implementarán el modelo relacional de datos que da solución al problema dado. Para este modelo, se detallara su desarrollo en uno de los primeros epígrafe. Para este modelo se ha utilizado como herramienta gestora de las relaciones de entidades ERECASE.

En este informe se abordará principales aspectos del desarrollo del software. Para ello se hará alusión de su Modelación Lógica y Física de los datos, un diagrama de clases donde se expondrá ciertos patrones de diseños implementados, además se cuenta con secciones donde se explicará brevemente el proceso de acceso a datos, así como su seguridad y validaciones que verifiquen la coherencia lógica de los datos ingresados.

Problema A Resolver

El problema a resolver está planteado en el siguiente enunciado:

En los centros estudiantiles se desea automatizar el control de los resultados docentes alcanzados por los alumnos, así como la actualización de la matrícula y otras informaciones necesarias para controlar la docencia. Para ellos se desarrollará un sistema que estará disponible en cada Facultad del Centro y que permita obtener las salidas siguientes:

1. Listado de los alumnos por grupo
2. Listado de las asignaturas que se imparten en cada año
3. Listado de las evaluaciones por grupo en cada asignatura
4. Listado de los promedios finales obtenidos por los alumnos de cada grupo
5. Escalafón
6. Certificación de notas de cada estudiante
7. Listado de los alumnos desaprobados por grupo para un rango de fecha
8. Listado por grupo o por año de los alumnos que causan baja en un curso
9. Listado de los repitentes por año

También se detallan algunas consideraciones que se han de tomar en cuenta como son las posibles de causas de la baja, las categorías de cada nota, que debe ocurrir antes de iniciar un curso escolar, entre otras.

Requisitos Funcionales y No Funcionales

Requisitos Funcionales:

- El sistema permite administrar los datos relacionados a la docencia del centro.
- El sistema persiste los datos a través de una base de datos relacional.
- El sistema devuelve una serie de reportes específicos.
- Los reportes del sistema son parametrizables.
- El sistema implementa niveles de seguridad para el acceso y la manipulación de los datos.
- Las contraseñas de los usuarios se almacenan encriptadas.

Requisitos No Funcionales:

- El DER debe de generarse con la herramienta ERECASE.
- Los nombres de las entidades, atributos, etc. deben ser en inglés.
- Se hace uso de los comentarios en la codificación.
- Versión de Java 17
- Biblioteca JavaFX 17
- Biblioteca AnimateFX-1.2.1
- Driver postgresql-9.0-801.jdbc3
- Servidor PostgreSQL versión 9.5
- Utilizar la arquitectura Modelo-Vista-Controlador

Descripción de la Solución Propuesta

Modelación Lógica DER

El proyecto tuvo su inicio en el desarrollo de un Diagrama de Entidades y Relaciones (DER), o lo que es lo mismo, la modelación lógica de los datos. Se realizó un análisis del enunciado y de los reportes que se necesitaban. Existían entidades que eran evidentes como la entidad Estudiante o la entidad Curso. Al final se concluyó en determinar las siguientes entidades:

- Brigade
- Subject
- Course
- Student
- Year
- DropOut
- Status
- Grade

Posteriormente se agregaron las entidades User y Rol para el control del acceso y manipulación de la base de datos a través de la aplicación.

En el análisis de las relaciones entre las entidades se determinaron las siguientes relaciones:

- Brigade_In_Year:
  Es una relación de 1 (Year) a muchos (Brigade) que relaciona a un grupo con el año al que pertenece. Por ejemplo, el Grupo número 1 que pertenece a Primer año. Esta relación primero se valoró modelarla no como relación sino que Brigade fuese una entidad débil de la entidad Year, pero se decidió utilizar la relación era más cómoda de implementar.

- Student_Status:
  Es una relación de 1 (Status) a muchos (Student) que relaciona a un estudiante con un estado (por ejemplo el estudiante es repitente) donde un estudiante puede tener un estado y un estado lo es de muchos estudiantes.

- Student_DropOut:
  Es una relación de muchos a muchos entre Student, DropOut y Course, que relaciona a un estudiante que haya causado baja con la causa de la baja en un curso específico.

- Student_In_Brigade:
  Es una relación de muchos a muchos entre Student, Brigade y Course que describe cómo un estudiante cursa un curso perteneciendo a un grupo determinado.

- Subject_In_Course:
  Es una relación de muchos a muchos entre Subject, Year y Course que relacion a una asignatura con el año escolar en que se imparte en un curso determinado y a su vez tiene asociada una duración en horas.

  Esta relación está a su vez relacionada con otras dos entidades que se explicarán a continuación, por lo que se decidió convertirla en una agregación. Ver Anexo 1.

- Student_Grade:
  Es una relación de muchos a muchos entre Student, Grade y la relación convertida a agregación compuesta por Subject, Year y Course (Subject_In_course). De manera que un estudiante tiene una nota en una asignatura que recibió en un año escolar específico en un curso dado.

Analizar el anexo 2 para una mayor referencia sobre el DER.

Modelo Físico de Datos

El modelo físico de los datos almacenados en la base de datos están estructurados de la siguiente manera:

Brigade

Esta tabla almacena la información referente a los grupos: los identificadores(id) de cada uno de los ellos dentro de un curso, su número dentro de un año (number) y una llave foránea del identificador del año(year_id).

Course

Registra cada uno de los datos de los cursos: el identificador de cada uno(id), cuando empiezan(start) y cuando terminan(finish).

Dropout

Tiene los identificadores de cada uno de los tipos de bajas (id) y las causas (cause) de estas.

Grade

Directamente almacena como llave primaria la propia nota (value) y como información adicional almacena la escala de estas notas (scale), que serían excelente, bien, regular, mal.

Status

Tabla para almacenar los tipos de estados que puede tener un estudiante, es un parámetro auxiliar para facilitar ciertas consultas, tiene los identificadores (id) y una breve descripción de este (description)

Student

Almacena toda la información referente a un estudiante, su identificador dentro de la base de datos (id), su carnet de identidad que es único (id_num), su nombre, (first_name), sus apellidos (last_name), su sexo (gender), su dirección (municipality), y una llave foránea del id del estado que presenta(status_id)

Subject

Guarda la información de las asignaturas, su identificador (id) y su nombre (name)

Year

Registra los años que tiene la facultad, su identificador (id) es el propio número del año

Student_dropout

Tabla relacional que guarda los estudiantes que son baja, su tipo de baja y el curso donde fue realizada. Sus columnas son llaves foráneas de los identificadores en las tablas **dropout**, **course**, **student**.

Student-grade

Almacena las notas de un estudiante en una asignatura en un curso relacionando las tablas

student, **grade**, **subject**, **course** y **year** mediante sus identificadores como llave foránea.

Student_in_brigade

Registra el grupo de un estudiante en un curso dado relacionando las tablas **student**, **brigade** y **course** usando para ello los identificadores en estas tablas.

Subject_in_course

tabla relacional creada con el objetivo de llevar en cada curso las asignaturas que se dan en cada año y la cantidad de horas que dura en ese curso (hours_long). Guarda como llaves foráneas los identificadores en las tablas **subject**, **course** y **year.**

Rol y **User**

Estas tablas son utilizadas para el acceso a la aplicación.

Diagramas de Clases

Para representar la información almacenada en las tablas de la base de datos se implementó el patrón de diseño Data Transfer Object. Este patrón consiste en tener un objeto que almacene cada valor de una tabla o entidad y para cada tabla un objeto DTO correspondiente. Siguiendo este principio tenemos siguientes clases de objetos:

SubjectDTO	StudentDTO	YearDTO	CourseDTO	BrigadeDTO
StatusDTO	DropOutDTO	GradeDTO	RolDTO	UserDTO

Y para las tablas relacionales:

StudentDropOutDTO	StudentGradeDTO	StudentBrigadeDTO	SubjectInCourseDTO

Para poder convertir las entidades y relaciones en objetos DTO, es necesario la creación de clases encargadas de este procedimiento. Estas clases son las llamadas servicios y existen una por cada dto. Todos los servicios cuentan con métodos como: insertar, actualizar, eliminar, cargar, y encontrar. Estos métodos los recopilamos en interfaces de las cuales, los servicios puedan implementar, y asi programar orientado a interfaces.

El **Diagrama 1** (ver anexo 3) explica de una forma gráfica la relación entre un servicio, un objeto dto y las interfaces que almacenan los procedimientos CRUD (Create, Read, Update, Delete).

El **Diagrama 2** (ver anexo 4) muestra la relación entre un servicio y un objeto, que representan a una tabla de relaciones en la base de datos. Este tipo de servicio conlleva una mayor

complejidad organizativa, pues a diferencia de los otros, para este, el método de encontrar, y actualizar puede o no diferir en la cantidad de parámetros. Por esta razón se subdivide la interfaz IServices y la IServicesBasic.

Acceso a Datos

La conexión entre la aplicación y la base de datos se produce haciendo uso de la arquitectura Java Database Connectivity (JDBC). Esta arquitectura presenta ventajas como independizar el método de acceso a los datos de la plataforma en que corre el sistema. Se utiliza un Driver que sirve de puente entre el Sistema Gestor de Bases de Datos (SGBD) y la API de JDBC. En el caso de este proyecto el driver es uno que permite conectar a una base de datos PostgreSQL.

Una vez cargado el driver (traductor entre la aplicación y la base de datos), se obtiene una conexión a la base de datos la cual se realiza a través de la clase Connection, suministrando la dirección de la base de datos y las credenciales para acceder a la misma. A través de esta conexión se realizan las llamadas a procedimientos almacenados en la base de datos o se realizan queries directos a las tablas. La información es devuelta a la aplicación y se utiliza a través de clases que provee la API tales como ResultSet.

Para el acceso a la base de datos se implementó una clase de servicios por cada tabla, donde están los métodos encargados de la interacción con su tabla especifica en la base de datos. Es decir, la clase StudentServices contiene los métodos que la aplicación utiliza para interactuar en la base de datos con la tabla “student”, estos métodos son por ejemplo llamadas a procedimientos almacenados que insertan o eliminan a un nuevo estudiante, siendo los métodos de esta clase de servicios el enlace al CRUD de dicha tabla en la base de datos. En la aplicación se emplean los Data Transfer Objects (DTOs) que consisten en clases que simulan las tablas de la base de datos al poseer atributos equivalentes a las columnas de las tablas; donde una instancia de la clase StudentDTO es el equivalente a una fila de la tabla “student”. En la aplicación se utilizó el patrón Singleton a través de la implementación de la clase ServicesLocator, a la cual acuden las clases controladoras de las vistas para buscar acceso a una clase de servicios específicas. De esta manera se asegura que exista únicamente una instancia de cada clase de servicios a la vez.

Seguridad

El primer nivel de seguridad que presenta la aplicación es al inicio de la ejecución del mismo. En la primera vista podemos notar un control de acceso que cuenta con campos de Identificación y autenticación de usuarios. Estos datos son verificados con datos ya existentes almacenados en la base de datos. En cuestión de seguridad de contraseña cabe destacar que

el algoritmo de cifrado utilizado es el MD5 para así brindar más protección al no almacenar en la base de datos directamente la contraseña.

La segunda vista es la de la aplicación en general, está dependiendo del componente donde interactúe el usuario se desplegarán vistas consecuentes. Cada vista solo mostrará lo necesario para el usuario. Es decir, si el usuario no es administrador, no tendrá los mismos niveles de acceso a componentes y por ende menos vistas y menos acciones.

En cuanto a la seguridad en la base de datos, esta cuenta con una seguridad brindada por el propio sistema de gestión de datos, en este caso lo es el pgAdmin. Esta seguridad también se basa en autentificación e identificación antes de poder manipularlo tan siquiera observar el contenido o estructura de base de datos.

Otro medio de seguridad a tener en cuenta es el obtenido al establecer el patrón MVC (Modelo Vista Controlador). Este modelo implica que la lógica de negocio y validaciones se encuentran modularizadas en clases encargadas de gestionar cualquier interacción del usuario con la interfaz, y según la lógica, por medio de este controlador se solicitará a las clases correspondientes al modelo (Servicios) el acceso a la base de datos. Además cualquier cambio en este modelo se reflejará en la interfaz del usuario. En resumidas cuentas, este patrón es beneficioso ya que el usuario no tiene acceso directo a la base de datos por medio de su interfaz. Esto garantiza más seguridad e integralidad lógica de los datos.

Validaciones

Durante la recogida de información ofrecida por el usuario deben cerciorarse que estos estén correctos para evitar daños y errores futuros.

Validaciones generales:

-Las entradas tienen un tamaño finito para almacenar en la base de datos (15-20 caracteres)

-Las entradas no pueden ser nulas

-Durante ciertas acciones se bloquean los botones de enviar hasta que no estén completados y seleccionados todos los campos (dar baja, añadir un usuario)

-El carnet de identidad se verifica su tamaño y la fecha de nacimiento sea correcta. También se revisa que no contenga letras.

-Los datos de las personas no debe contener números a menos que así lo requiera la entrada.

Interfaces:

Las interfaces graficas de usuario fueron desarrolladas con JavaFX en su versión 17. Principalmente existen tres tipos de ventanas.

Primeramente la ventana de Login, a través de la cual el usuario de la aplicación se autentica introduciendo su nombre de usuario y su contraseña. Esta ventana tiene un tamaño fijo. Ver anexo 5.

Próximamente está la ventana principal. En esta ventana se desarrollan todas las interacciones entre el usuario y la aplicación. En la esquina superior izquierda el usuario puede cerrar su sesión actual. En el lado izquierdo posee un menú de navegación para acceder a distintas secciones en dependencia de la información que se desee encontrar. En la zona central de la ventana se muestran las tablas con los datos que se cargan de la base de datos. Ver anexo 6.

Por ultimo tenemos las ventanas emergentes que se utilizan para las operaciones que requieren datos introducidos por el usuario. Estas son sencillas, contienen únicamente los campos requeridos para que el usuario proporcione la información y cuando se cierran regresan a la ventana principal. Ver anexo 7.

La aplicación está desarrollada siguiendo los principios de la arquitectura Modelo-Vista-Controlador (MVC) en la cual hay una separación de responsabilidades entre la capa visual, la de los datos, y una capa controladora que se comunica con estas dos.

Conclusiones

Este trabajo significo la realización de un proyecto que integra desarrollo de vistas y de servidor con base de datos solidificando los conocimientos del desarrollo de cada una de ellas por separado. Cabe destacar las ventajas que presenta el desarrollo de las interfaces en Java FX, pues pese a ser algo relativamente nuevo de aprender, facilita mucho ciertos trabajos, independientemente de la calidad visual. El modelo relacional presenta importantes ventajas para organizar un acceso rápido a la información almacenada en las bases de datos, evitando así largos tiempos de carga, algo fundamental para acceder a gran cantidad de información como la que puede presentar una aplicación de este estilo. El guardado de las funciones para acceder a los datos, modificar, añadir y eliminar datos exime de esta responsabilidad a la aplicación, añadiendo seguridad y portabilidad a la plataforma, pues pueden ser usados independientemente de la esta. El uso de la arquitectura cliente-servidor permite la posibilidad de realizar diversas consultas provenientes de diferentes clientes y así responder de manera eficiente cada una de ellas sin provocar una sobrecarga de parte del servido. Por esta razón, este sistema es adecuado para mantener a flote un software que demanda muchas solicitudes de partes de muchos usuarios que conforman un departamento educacional.

Recomendaciones

Se pudiera recomendar el uso de herramientas más avanzadas para la creación de las funciones CRUD de la base de datos, pues a pesar de ser un proyecto de menor escala y con fines educativos, la realidad es que generalmente surgen una serie de errores humanos que atrasan el desarrollo del mismo, de manera significativa.

Otra realidad con la que se puede chocar en la realización de estos proyectos es dar suma importancia a estructurar correctamente el proyecto, planificar los métodos, clases, vistas y de manera general todo antes de llevar el proyecto a la realidad. Todo lo anterior es con el objetivo de evitar retornos innecesarios y evitar importantes pérdidas de tiempo.

Bibliografía y Recursos

1. C.J.Date. “Introducción a las Sistemas de Bases de Datos”

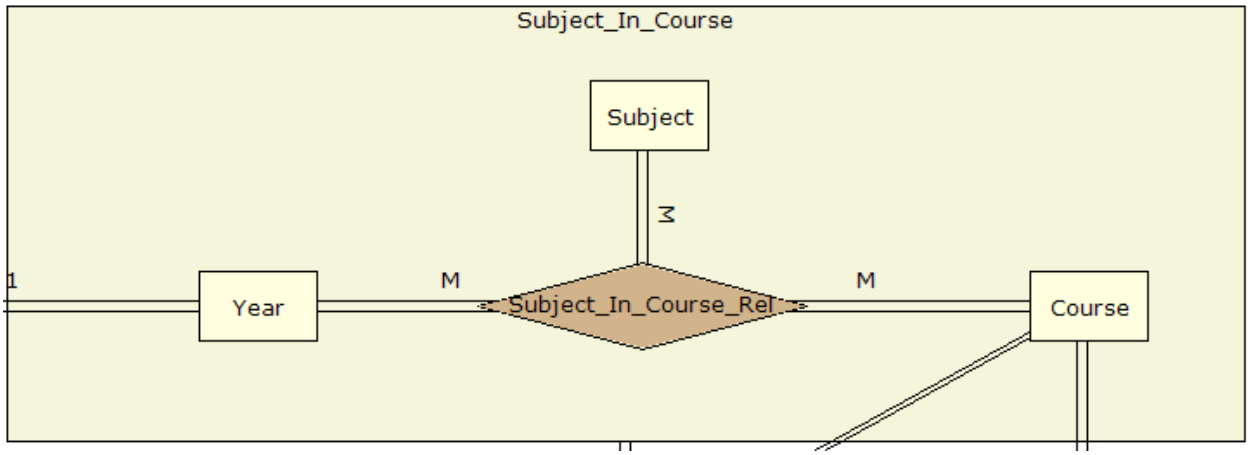
2. Documentación del lenguaje Java.

3. Materiales disponibles en el ICloud acerca del lenguaje Plpgsql

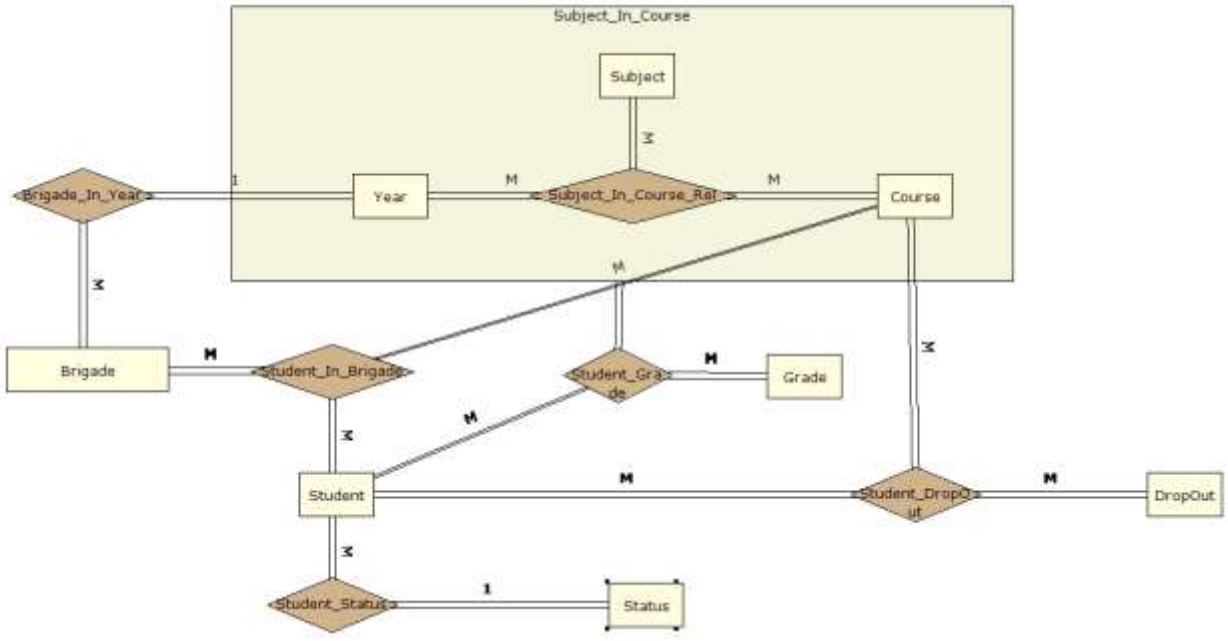
4. ¨MVC Design Pattern¨ GeeksForGeeks https://www.geeksforgeeks.org/mvc-design-pattern/

5. StackOverflow.com

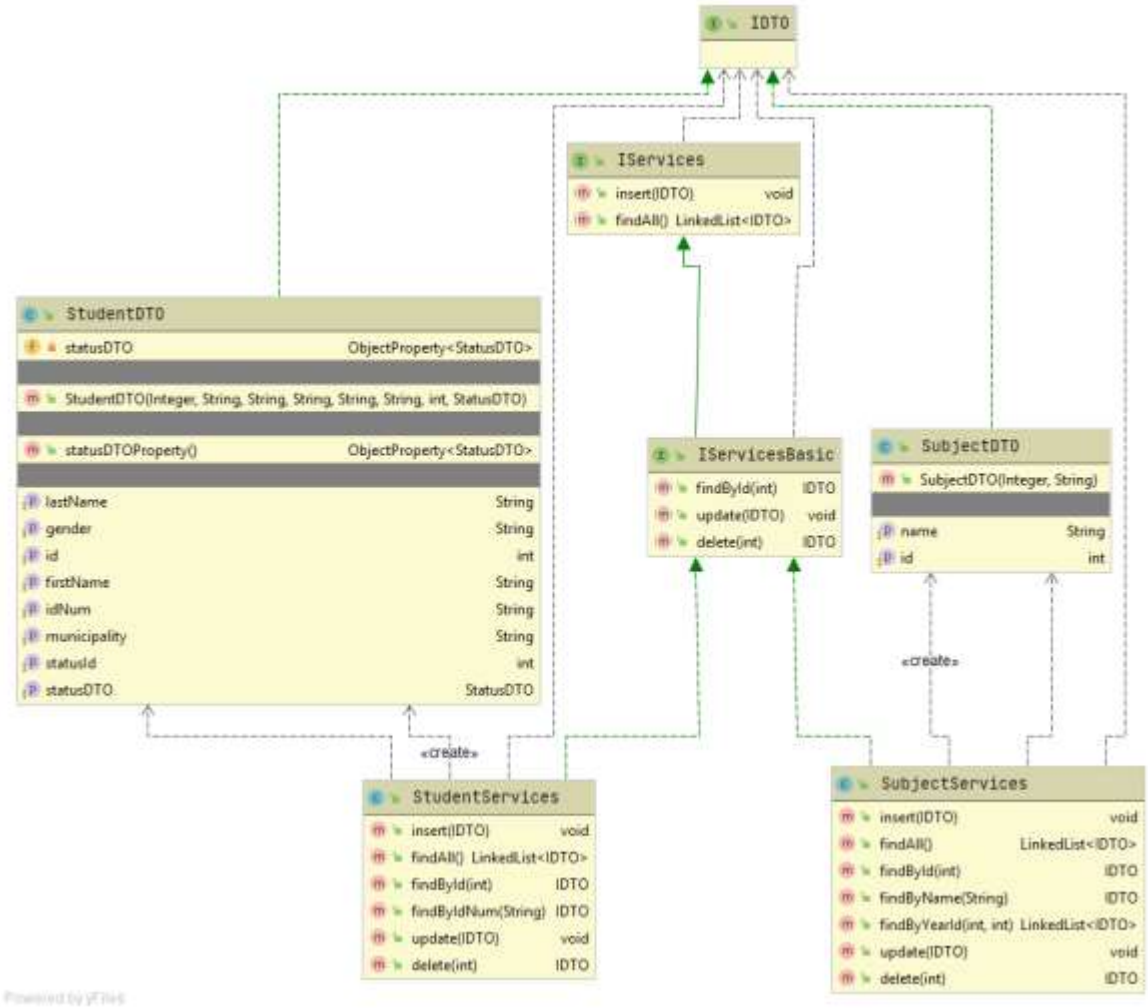
Anexo 1



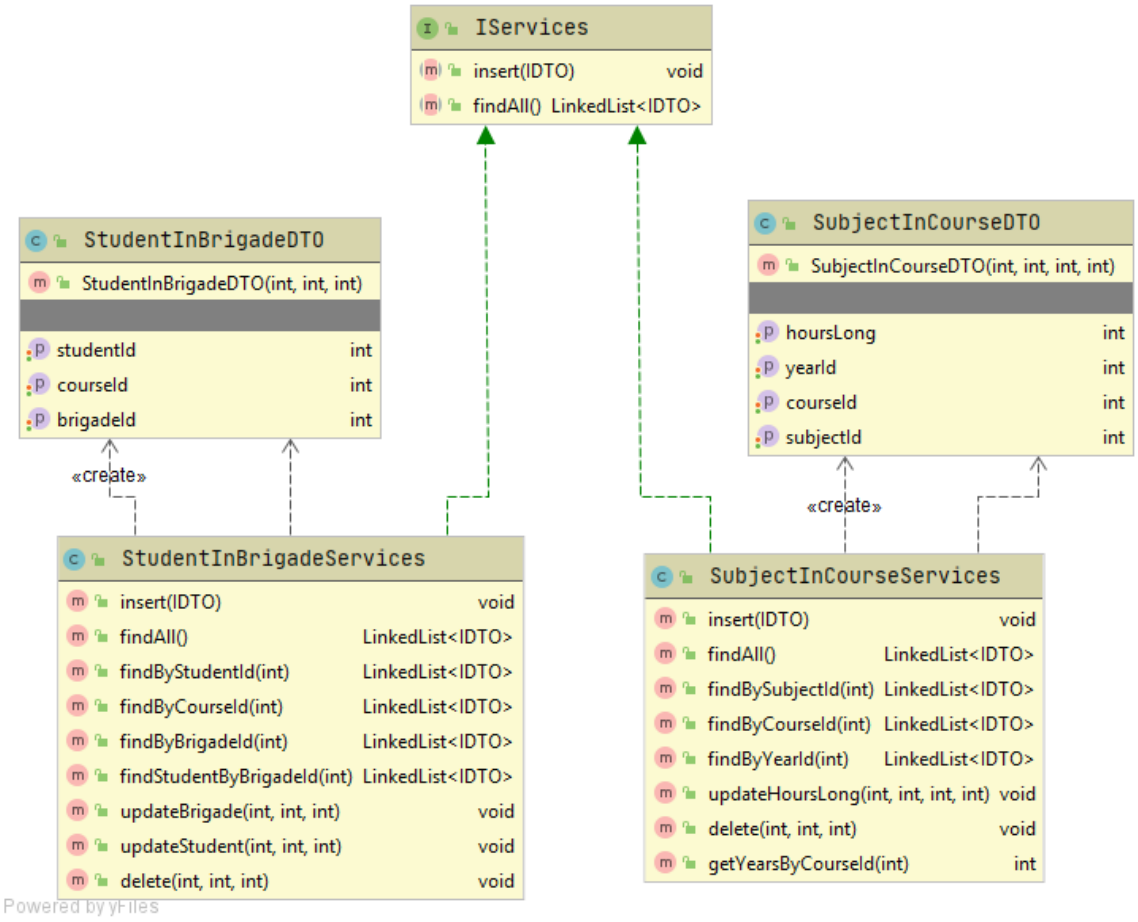
Anexo 2



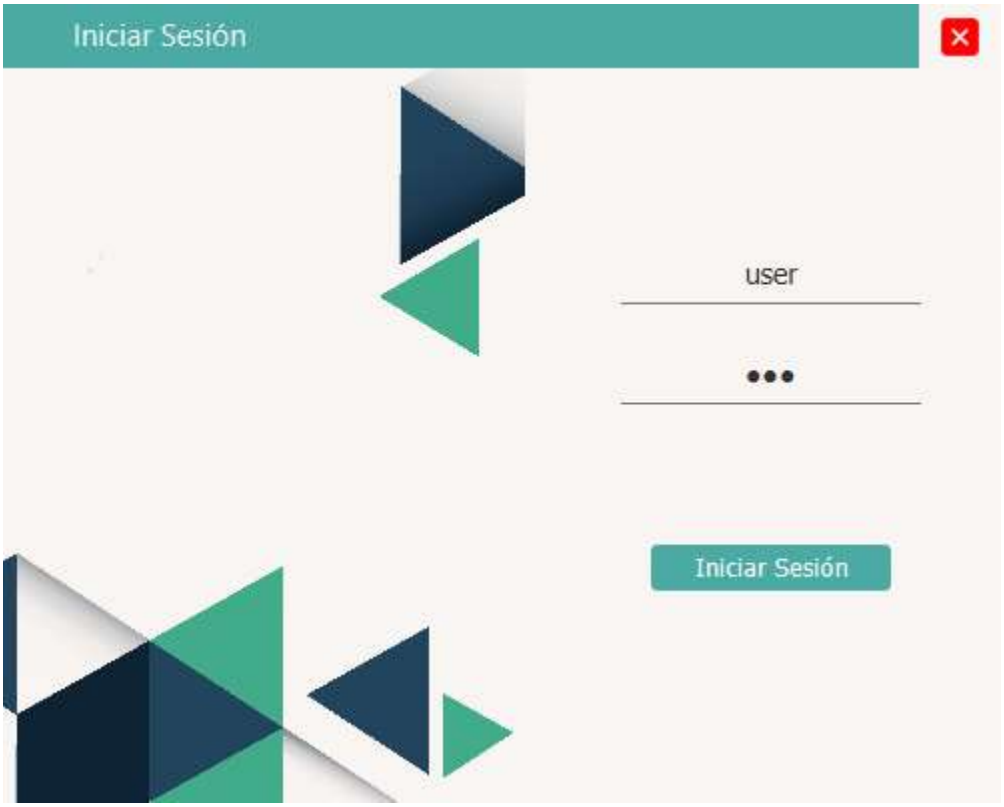
Anexo 3 (Diagrama 1)



Anexo 4 (Diagrama 2)



Anexo 5



Anexo 6

Anexo 7