SPGER

Experiencia Educativa:

Principios de Construcción de Software

Nombre del docente:

Juan Carlos Pérez Arriaga

Elaborado por:

-Luis Alonso Andrade López

-Jesus Manuel Mújica Conde

-Xavier Arian Olivares Sánchez

Fecha de elaboración:

14 de Junio, 2023

API

Capa de Acceso a datos

DataBaseManager

Esta clase es la única que tiene autorización para establecer una conexión con la base de datos.

Atributos:

- connection: Connection: metodo que es la conexión a la base de datos.
- dataBaseUserPropertiesFile: Archivo properties con nombre de la base de datos, el usuario y la contraseña del usuario de la base de datos para conectarse.

Métodos:

- DataBaseManager(): Constructor por defecto que asigna la ruta del archivo properties al metodo dataBaseAttributesPropertiesFile.
- closeConnection(): void: Metodo para cerrar la conexión a la base de datos.
- getConnection(): Connection: Metodo para obtener la conección de la base de datos a travez del atributo connection.
- connect(): void: Metodo que asigna el archivo properties al metodo connection.

Capa Lógica

DAOs

AcademicBodyDAO

Dao que permite crear consultas para poder hacer CRUD con la base de datos en la tabla Cuerpo Academico.

Métodos:

- addAcademicBody(academicBody: AcademicBody): int: Este método genera una consulta para agregar un Cuerpo Academico en la base de datos con ayuda del DataBaseManager.
- getAcademicBodiesList(): ArrayList<AcademicBody>: Este metodo trae los datos de todos los Cuerpos Academicos registrados de la base de datos con ayuda del DataBaseManager.
- getAcademicBodyById(academicBodyId: int): AcademicBody: Este metodo trae el Cuerpo Academico de la base de datos que tenga asignada la id que es introducida como parámetro en esta función, todo esto con ayuda del DataBaseManager.

AcademicBodyHeadDAO

Dao que permite crear consultas para poder hacer CRUD con la base de datos en las tablas Usuarios, Profesores y ResponsablesCA.

Métodos:

- addAcademicBodyHead(academicBodyHead: AcademicBodyHead): int: Este método genera una consulta para agregar un Miembro del Cuerpo Académico en la base de datos con ayuda del DataBaseManager.
- modifyAcademicBodyHeadData(academicBodyHead: AcademicBodyHead): int: Este metodo genera una consulta para modificar un Miembro del Cuerpo Academico de la base de datos con ayuda del DataBaseManager.
- getAcademicBodyHeads(): ArrayList<AcademicBodyHead>: Este metodo trae los datos de todos los Miembros del Cuerpo Academico registrados de la base de datos con ayuda del DataBaseManager.
- getSpecifiedAcademicBodyHeads(academicBodyHeadName: String): ArrayList<AcademicBodyHead>: Este metodo trae los Miembros del Cuerpo Academico de la base de datos que empiezen con la palabra que es introducida como parámetro en esta función, todo esto con ayuda del DataBaseManager.

- getAcademicBodyHead(staffNumber: int): AcademicBodyHead: Este metodo devuelve los datos del Miembro del Cuerpo Academico que tenga asignada la id que es introducida como parámetro en esta función, todo esto con ayuda del DataBaseManager.
- deleteAcademicBodyHeadFromUsersTable(): void: Este metodo elimina de la tabla Usuarios de la base de datos los datos del Miembro del Cuerpo Academico agregado en el metodo addAcademicBodyHead o modifyAcademicBodyHeadData, todo esto con ayuda del DataBaseManager.

ActivityDAO

Dao que permite crear consultas para poder hacer CRUD con la base de datos en la tabla Actividad.

Métodos:

- addActivity(activity: Activity): int: Este método genera una consulta para agregar una Actividad en la base de datos con ayuda del DataBaseManager.
- getActivityList(researchId: int): ArrayList<Activity>: Este metodo trae los datos de todas las Actividades registrados de la base de datos con ayuda del DataBaseManager.
- modifyActivity(activity: Activity): int: Este método genera una consulta para actualizar la información de la Actividad de la base de datos que coincida con el id ingresado de parámetro en la función, todo esto con ayuda del DataBaseManager.
- setComment(comment: String, activityId: int): int: Este metodo sirve para agregar un comentario a una Actividad de la base de datos que coincida con la id ingresada en los parámetros, todo esto con ayuda del DataBaseManager.
- setFeedback(feedback: String, activityId: int): int: Este metodo sirve para agregar una retroalimentación a una Actividad de la base de datos que coincida con la id ingresada en los parámetros, todo esto con ayuda del DataBaseManager.

AdvanceDAO

Dao que permite crear consultas para poder hacer CRUD con la base de datos en la tabla Avances.

Métodos:

- addAdvance(advance: Advance): int: Este método genera una consulta para agregar un Avance en la base de datos con ayuda del DataBaseManager.
- getAdvanceList(): ArrayList<Advance>: Este metodo trae los datos de todos los Avances registrados de la base de datos con ayuda del DataBaseManager.
- getActivityAdvance(activityId: int): ArrayList<Advance>: Este metodo trae los datos de todos los Avances asignados a la Actividad que coincida con la id ingresada en el parámetro, todo esto con ayuda del DataBaseManager.

- setFeedBack(advance: Advance): int: Este método genera una consulta para agregar una retroalimentación a un Avance en la base de datos con ayuda del DataBaseManager.
- getAdvanceById(activityId: int): Advance: Este metodo trae los datos del Avance de la base de datos que coincida con la id del parámetro de la función, todo esto con ayuda del DataBaseManager.
- updateAdvanceInfo(int: advanceToBeUpdatedID): int: Este método genera una consulta para actualizar la información del Avance de la base de datos que coincida con el id ingresado de parámetro en la función, todo esto con ayuda del DataBaseManager.

CourseDAO

Dao que permite crear consultas para poder hacer CRUD con la base de datos en la tabla Cursos.

Métodos:

- addCourse(course: Course): int: Este método genera una consulta para agregar un Curso en la base de datos con ayuda del DataBaseManager.
- modifyCourseData(course: Course): int: Este metodo genera una consulta para modificar un Curso de la base de datos con ayuda del DataBaseManager.
- getCourses(): ArrayList<Course>: Este metodo trae los datos de todos los Cursos registrados de la base de datos con ayuda del DataBaseManager.
- getSpecifiedCourses(courseNrc: int): ArrayList<Course>: Este metodo trae los Cursos de la base de datos que empiezen con el NRC que es introducido como parámetro en esta función, todo esto con ayuda del DataBaseManager.
- getCourse(courseNrc: int): Course: Este metodo devuelve los datos del Curso que tenga asignada la id que es introducida como parámetro en esta función, todo esto con ayuda del DataBaseManager.

DegreeBossDAO

Dao que permite crear consultas para poder hacer CRUD con la base de datos en las tablas Usuarios, Profesores y JefesCarrera.

Métodos:

- addDegreeBoss(degreeBoss: DegreeBoss): int: Este método genera una consulta para agregar un Jefe de Carrera en la base de datos con ayuda del DataBaseManager.
- modifyDegreeBossData(degreeBoss: DegreeBoss): int: Este metodo genera una consulta para modificar un Jefe de Carrera de la base de datos con ayuda del DataBaseManager.

- getDegreeBosses(): ArrayList<DegreeBoss>: Este metodo trae los datos de todos los Jefes de Carrera registrados de la base de datos con ayuda del DataBaseManager.
- getSpecifiedDegreeBosses(degreeBossName: String): ArrayList<DegreeBoss>: Este metodo trae los Jefes de Carrera de la base de datos que empiezen con la palabra que es introducida como parámetro en esta función, todo esto con ayuda del DataBaseManager.
- getDegreeBoss(staffNumber: int): DegreeBoss: Este metodo devuelve los datos del Jefe de Carrera que tenda la asignada la id que es introducida como parámetro en esta función, todo esto con ayuda del DataBaseManager.
- deleteDegreeBossFromUsersTable(): void: Este metodo elimina de la tabla Usuarios de la base de datos los datos del Jefe de Carrera agregado en el metodo addDegreeBoss o modifyDegreeBossData, todo esto con ayuda del DataBaseManager.

DirectorDAO

Dao que permite crear consultas para poder hacer CRUD con la base de datos en las tablas Usuarios, Profesores y Director.

Métodos:

- addDirector(director: Director): int: Este método genera una consulta para agregar un Director en la base de datos con ayuda del DataBaseManager.
- modifyDirectorData(director: Director): int: Este metodo genera una consulta para modificar un Director de la base de datos con ayuda del DataBaseManager.
- getDirectors(): ArrayList<Director>: Este metodo trae los datos de todos los Directores registrados de la base de datos con ayuda del DataBaseManager.
- getSpecifiedDirectors(directorName: String): ArrayList<Director>: Este metodo trae los Directores de la base de datos que empiezen con la palabra que es introducida como parámetro en esta función, todo esto con ayuda del DataBaseManager.
- getDirector(staffNumber: int): Director: Este metodo devuelve los datos del Director que tenga asignada la id que es introducida como parámetro en esta función, todo esto con ayuda del DataBaseManager.
- deleteDirectorFromUsersTable(): void: Este metodo elimina de la tabla Usuarios de la base de datos los datos del Director agregado en el metodo addDirector o modifyDirectorData, todo esto con ayuda del DataBaseManager.

FileDAO

Dao que permite crear consultas para poder hacer CRUD con la base de datos en la tabla Archivos.

Métodos:

- addFile(filepath: FIlePath): int: Este método genera una consulta para agregar un Archivo en la base de datos con ayuda del DataBaseManager.
- getFileById(fileId: int): File: Este metodo devuelve los datos del Archivo que tenga asignada la id que es introducida como parámetro en esta función, todo esto con ayuda del DataBaseManager.

KGALDAO

Dao que permite crear consultas para poder hacer CRUD con la base de datos en la tabla LGAC.

Métodos:

- addKGAL(kgal: KGAL): int: Este método genera una consulta para agregar un LGAC en la base de datos con ayuda del DataBaseManager.
- updateKGALDescription(kgalID: int, description: String): int: Este metodo genera una consulta para modificar la descripción de un LGAC de la base de datos con ayuda del DataBaseManager y con la id ingresada en el parámetro.
- getKGALList(): ArrayList<KGAL>: Este metodo trae los datos de todos los LGACS registrados de la base de datos con ayuda del DataBaseManager.
- getKGALById(kgalId: int): KGAL: Este metodo trae el LGAC de la base de datos que tenga asignada la id que es introducida como parámetro en esta función, todo esto con ayuda del DataBaseManager.
- getKGALByDescription(description: String): KGAL: Este metodo devuelve los datos del LGAC que tenga asignada la descripcion que es introducida como parámetro en esta función, todo esto con ayuda del DataBaseManager.
- getKGALListByDescription(description: String): ArrayList<KGAL>: Este metodo devuelve los datos del los LGACS que coincidan con la descripción que es introducida como parámetro en esta función, todo esto con ayuda del DataBaseManager.

LoginDAO

Dao que permite crear consultas para poder hacer verificaciones de datos con la base de datos en las tablas Usuarios, Estudiantes, Profesores , DIrectores, JefesCarrera y ResponsablesCA.

Métodos:

- loginAdmin(emailAddress: String, password: String): DegreeBoss: Este método genera una consulta para devolver el Jefe De Carrera que coincide con la dirección de correo electrónico y la contraseña introducida en los parámetros, todo esto con ayuda del DataBaseManager.

- loginAcademicBodyHead(emailAddress: String, password: String): AcademicBodyHead: Este método genera una consulta para devolver el Miembro del Cuerpo Académico que coincide con la dirección de correo electrónico y la contraseña introducida en los parámetros, todo esto con ayuda del DataBaseManager.
- loginDirector(emailAddress: String, password: String): Director: Este método genera una consulta para devolver el Director que coincide con la dirección de correo electrónico y la contraseña introducida en los parámetros, todo esto con ayuda del DataBaseManager.
- loginProfessor(emailAddress: String, password: String): Professor: Este método genera una consulta para devolver el Profesor que coincide con la dirección de correo electrónico y la contraseña introducida en los parámetros, todo esto con ayuda del DataBaseManager.
- loginStudent(matricle: String, password: String): Student: Este método genera una consulta para devolver el Estudiante que coincide con la matrícula y la contraseña introducida en los parámetros, todo esto con ayuda del DataBaseManager.

ProfessorDAO

Dao que permite crear consultas para poder hacer CRUD con la base de datos en las tablas Usuarios y Profesores.

Métodos:

- addProfessor(professor: Professor): int: Este método genera una consulta para agregar un Profesor en la base de datos con ayuda del DataBaseManager.
- modifyProfesorData(professor: Professor): int: Este metodo genera una consulta para modificar un Profesor de la base de datos con ayuda del DataBaseManager.
- getProfessors(): ArrayList<Professor>: Este metodo trae los datos de todos los Profesores registrados de la base de datos con ayuda del DataBaseManager.
- getSpecifiedProfessors(professorName: String): ArrayList<Professor>: Este metodo trae los Profesores de la base de datos que empiezen con la palabra que es introducida como parámetro en esta función, todo esto con ayuda del DataBaseManager.
- getProfessor(staffNumber: int): Professor: Este metodo devuelve los datos del Profesor que tenga asignada la id que es introducida como parámetro en esta función, todo esto con ayuda del DataBaseManager.
- deleteProfessorFromUsersTable(): void: Este metodo elimina de la tabla Usuarios de la base de datos los datos del Profesor agregado en el metodo addProfessor o modifyProfessorData, todo esto con ayuda del DataBaseManager.

ResearchDAO

Dao que permite crear consultas para poder hacer CRUD con la base de datos en la tabla Anteproyectos.

Métodos:

- addResearch(research: Research): int: Este método genera una consulta para agregar un Anteproyecto en la base de datos con ayuda del DataBaseManager.
- getResearchProjectList(): ArrayList<ResearchProject>: Este metodo trae los datos de todos los Anteproyectos registrados de la base de datos con ayuda del DataBaseManager.
- getDirectorsResearch(int staffNumber): ArrayList<ResearchProject>: Este metodo trae los datos de todos los Anteproyectos registrados que tengan asignado un Director que coincida con el numero de personal ingresado en el parámetro de la función, todo esto con ayuda del DataBaseManager.
- getStudentsResearch(String matricle): ResearchProject: Este metodo trae los datos de todos los Anteproyectos registrados que tengan asignado un Estudianteque coincida con la matrícula ingresada en el parámetro de la función, todo esto con ayuda del DataBaseManager.
- getCourseResearch(int NRC): ArrayList<ResearchProject>: Este metodo trae los datos de todos los Anteproyectos registrados que tengan asignado un Curso que coincida con el nrc ingresado en el parámetro de la función, todo esto con ayuda del DataBaseManager.
- getSpecifiedResearchProjectList(String researchName): ArrayList<ResearchProject>: Este metodo trae los datos de todos los Anteproyectos registrados de la base de datos que empiecen con el nombre introducido en el parámetro de la función, todo esto con ayuda del DataBaseManager.
- getSpecifiedValidatedResearchProjectList(String researchName): ArrayList<ResearchProject>: Este metodo trae los datos de todos los Anteproyectos validados registrados de la base de datos que empiecen con el nombre introducido en el parámetro de la función, todo esto con ayuda del DataBaseManager.
- getSpecifiedNotValidatedResearchProjectList(String researchName): ArrayList<ResearchProject>: Este metodo trae los datos de todos los Anteproyectos no validados registrados de la base de datos que empiecen con el nombre introducido en el parámetro de la función, todo esto con ayuda del DataBaseManager.
- getSpecifiedValidatedAndNotValidatedResearchProjectList(String researchName): ArrayList<ResearchProject>: Este metodo trae los datos de todos los Anteproyectos validados y no validados registrados de la base de datos que empiecen con el nombre introducido en el parámetro de la función, todo esto con ayuda del DataBaseManager.
- modifyResearch(ResearchProject research): int: Este metodo genera una consulta para modificar un Anteproyecto de la base de datos con ayuda del DataBaseManager.
- validateResearch(ResearchProject researchProject): void: Este metodo genera una consulta para validar un Anteproyecto de la base de datos con ayuda del DataBaseManager.

ResearchReportDAO

Dao que permite crear consultas para poder hacer lecturas de datos con la base de datos en la tabla Anteproyecto.

Métodos:

- getResearches(title: String, query: String): ArrayList<ResearchProject>: Este metodo trae los datos de todos los Anteproyectos registrados de la base de datos con ayuda del DataBaseManager.
- getValidatedResearches(title: String): ArrayList<ResearchProject>: Este metodo trae los datos de todos los Anteproyectos validados registrados de la base de datos con ayuda del DataBaseManager.
- getNotValidatedResearches(title: String): ArrayList<ResearchProject>: Este metodo trae los datos de todos los Anteproyectos no validados registrados de la base de datos con ayuda del DataBaseManager.
- getValidatedAndNotValidatedResearches(title: String): ArrayList<ResearchProject>: Este metodo trae los datos de todos los Periodos Anteproyectos validados o no validados registrados de la base de datos con ayuda del DataBaseManager.
- getSelectedResearches(selectedResearchesTitles: ArrayList<String>): ArrayList<ResearchProject>: Este metodo trae los datos de tlos Periodos Escolares.

ScholarPeriodDAO

Dao que permite crear consultas para poder hacer CRUD con la base de datos en la tabla PeriodosEscolares.

Métodos:

- getScholarPeriods(): ArrayList<ScholarPeriod>: Este metodo trae los datos de todos los Periodos Escolares registrados de la base de datos con ayuda del DataBaseManager.
- getScholarPeriod(scholarPeriodId: int): ScholarPeriod: Este metodo devuelve los datos del Periodo Escolar que tenga asignada la id que es introducida como parámetro en esta función, todo esto con ayuda del DataBaseManager.

StudentDAO

Dao que permite crear consultas para poder hacer CRUD con la base de datos en las tablas Usuarios y Estudiantes.

Métodos:

- addStudent(student: Student): int: Este método genera una consulta para agregar un Estudiante en la base de datos con ayuda del DataBaseManager.
- modifyStudentData(student: Student): int: Este metodo genera una consulta para modificar un Estudiante de la base de datos con ayuda del DataBaseManager.
- getStudents(): ArrayList<Student>: Este metodo trae los datos de todos los Estudiantes registrados de la base de datos con ayuda del DataBaseManager.

- getSpecifiedStudents(studentName: String): ArrayList<Student>: Este metodo trae los Estudiantes de la base de datos que empiezen con la palabra que es introducida como parámetro en esta función, todo esto con ayuda del DataBaseManager.
- getStudent(matricle: String): Student: Este metodo devuelve los datos del Estudiante que tenga asignada la matrícula que es introducida como parámetro en esta función, todo esto con ayuda del DataBaseManager.
- deleteStudentFromUsersTable(): void: Este metodo elimina de la tabla Usuarios de la base de datos los datos del Estudiante agregado en el metodo addStudent o modifyStudentData, todo esto con ayuda del DataBaseManager.

StudentsCoursesDAO

Dao que permite crear consultas para poder hacer CRUD con la base de datos en la tabla EstudiantesCurso.

Métodos:

- addStudentCourse(student: Student): int: Este método genera una consulta para asignar un Estudiante a un Curso en la base de datos con ayuda del DataBaseManager.
- getStudentsCourse(matricle: String): ArrayList<Course>: Este metodo trae los datos de todas las asignaciones de Estudiantes a Cursos registrados de la base de datos con ayuda del DataBaseManager.
- getStudentsMatriclesByCourseNRC(courseNrc: String): ArrayList<String>: Este metodo trae los Estudiantes de la base de datos que estén registrados a un curso en específico, todo esto con ayuda del DataBaseManager.
- removeStudentCourse(matricle: String, courseNrc: String): void: Este metodo elimina el registro de un Estudiante a un Curso en específico de la base de datos, todo esto con ayuda del DataBaseManager.

Capa Gráfica

Controladores

Metodos en común

LoadHeader()

Este método se utiliza para cargar el header en la parte superior de los frames que lo requieran. Requiere conocer el usuario que accede al sistema

SetUser()

Este método se utiliza para definir el usuario que accede a cierta clase. Tiene el objetivo de regular los permisos que tiene dicho usuario para realizar ciertas acciones o acceder a ciertos apartados.

SetCourse()

Este método tiene como principal objetivo regular si se está accediendo al cronograma mediante la vista de un curso, o desde la parte de revisión. En su mayoría solo sirve para los headers.

LoginController

Controlador que permite verificar que tipo de menú principal abrir dependiendo el correo, matrícula o contraseña asignada.

Métodos:

- login(): void: Este metodo verifica que tipo de usuario es el que encontró con la contraseña y el correo o matrícula ingresados en la ventana de login.
- openMainMenu(user: User): void: Este metodo checa que tipo de menú principal abrir dependiendo el usuario que recibe como parámetro.

MainMenuController

Controlador que abre un tipo de menú principal dependiendo el tipo de Usuario con el que se acceda.

Métodos:

- exitButtonController(): void: Este metodo simplemente cierra el menú principal.
- loadCourses(): void: Este metodo crea objetos para acceder a los cronogramas.
- loadSpecialPanes(): void: Este metodo crea objetos para ingresar a los administradores de cursos, anteproyectos, lgacs o usuarios.

HeaderPaneController

Controlador que tiene como función hacer la carga el panel que tiene el nombre SPGER, el logo y el nombre del usuario con el que se ingresó al sistema.

Métodos:

- goHome(): void: Este metodo simplemente cierra la ventana actual y abre el menú principal.
- setCourse(): void: Este metodo modifica el texto del centro del panel con el texto deseado

GuiUsersController

Controlador que sirve para mostrar el administrador de Usuarios del sistema.

Métodos:

- registerUserButtonController(): void: Este metodo simplemente abre la ventana GuiRegisterUser.
- searchByNameButtonController(): void: Este metodo busca los usuarios cuyo nombre empiecen con el exto introducido en el campo de texto.
- loadUserButtons(): void: Este metodo crea los botones con el nombre, tipo de usuario y nombre o numero de personal de los usuarios del sistema.
- openPaneWithUserInformation(): void: Este metodo muestra un panel con el nombre, apellidos, correos y numero de telefono del usuario seleccionado.
- openModifyUserPane(): void: Este metodo muestra un panel para modificar el nombre, apellidos, correos y numero de telefono del usuario seleccionado.

GuiRegisterUserController

Controlador que abre una ventana para registrar usuarios al sistema.

Métodos:

- registerUserButtonController(): void: Este metodo manda los datos de los campos de texto en los daos de usuarios.

ResearchManagerController

Controlador que abre el administrador de Anteproyectos del sistema.

Métodos:

- addResearch(): void: Este metodo abre la ventana AddResearch.
- toGoReports(): void: Este metodo abre la ventana ToGoReports.
- searchButtonController(): void: Este metodo busca los anteproyectos que empiecen con el texto escrito en el campo de texto.

AddResearchController

Controlador que abre una ventana para poder registrar anteproyectos al sistema.

Métodos:

- addResearch(): void: Este metodo manda los datos de los campos de texto en los daos de Anteproyectos.

GuiResearchReportController

Controlador que abre una ventana para seleccionar los anteproyectos para poder hacer un reporte con ellos.

Métodos:

- backButtonController(): void: Este metodo cierra la ventana GuiResearchReport.
- generateReportButtonController(): void: Este metodo genera un reporte con los Anteproyectos seleccionados.
- searchResearchesButtonController(): void: Este metodo busca los anteproyectos que empiecen con el texto escrito en el campo de texto.

KGALListController

Controlador que abre el administrador de LGACS del sistema.

Métodos:

- exitKGALScene(): void: Este metodo cierra la ventana KGALList.
- switchToCreateNewKGALScene(): void: Este metodo abre la ventana CreateNewKGAL.
- searchResearchesButtonController(): void: Este metodo busca los LGACS que empiecen con el texto escrito en el campo de texto.

CreateNewKGALController

Controlador que abre una ventana para poder registrar un LGAC al sistema.

Métodos:

- createNewKGAL(): void: Este metodo crea un LGAC con los datos de los campos de texto y usa un dao para registrarlos datos en la base de datos.
- switchToKGALListScene(): void: Este metodo abre la ventana KGALList.

GuiCoursesController

Controlador que abre el administrador de cursos del sistema.

Métodos:

- registerCourseButtonController(): void: Este metodo simplemente abre la ventana GuiRegisterCourse.
- searchByNrcButtonController(): void: Este metodo busca los cursos cuyo nrc empiecen con el texto introducido en el campo de texto.
- loadCourseButtons(): void: Este metodo crea los botones con el nombre, nrc y periodo escolar de los cursos del sistema.
- openPaneWithCourseInformation(): void: Este metodo muestra un panel con el bloque, seccion, nombre, periodo escolar y nrc del curso seleccionado.
- openModifyCoursePane(): void: Este metodo muestra un panel para modificar llos datos del curso seleccionado.

GuiRegisterCourseController

Controlador que abre una ventana para poder registrar cursos al sistema.

Métodos:

- registerUserButtonController(): void: Este metodo manda los datos de los campos de texto en los daos de Cursos.

GuiUsersCourseController

Controlador que abre una ventana con todos los estudiantes asignados a un curso en específico

Métodos:

- addButtonsController(): void: Este metodo crea los botones con el nombre y matrícula de los usuarios asignados al curso seleccionado.
- backButtonsController(): void: Este metodo abre la ventana GuiUsersCourseController.

GuiStudentAdderController

Controlador que abre una ventana para seleccionar los estudiantes disponibles para asignación a cursos.

Métodos:

- addStudentsButtonsController(): void: Este metodo crea los botones con el nombre, de los estudiantes que hay disponibles en el sistema.
- showByMatricleButtonController(): void: Este metodo muestra un panel con el bloque, seccion, nombre, periodo escolar y nrc del curso seleccionado.

ChronogramController

Controlador que abre la ventana de cronogramas dependiendo el tipo de usuario con el que se esté accediendo.

Métodos:

- switchChronogram(): void: Este metodo sirve para seleccionar los cronogramas disponibles, solo es accesible para los usuarios que deriven de Profesor
- setDirectorView(): void: Este metodo muestra el cronograma específico para los Directores.
- setProfessorView(): void: Este metodo muestra el cronograma específico para los Profesores.
- setStudentView(): void: Este metodo muestra el cronograma específico para los Estudiantes.

CreateActivityController

Controlador que abre una ventana para poder crear acttividades en un cronograma en específico.

Métodos:

- createActivity(): void: Este metodo crea una actividad con los campos introducidos y los guarda en la base de datos con ayuda del dao de actividades.
- returnToChronogram(): void: Este metodo abre la pestaña del cronograma.

Modelos de datos

Diagrama ER

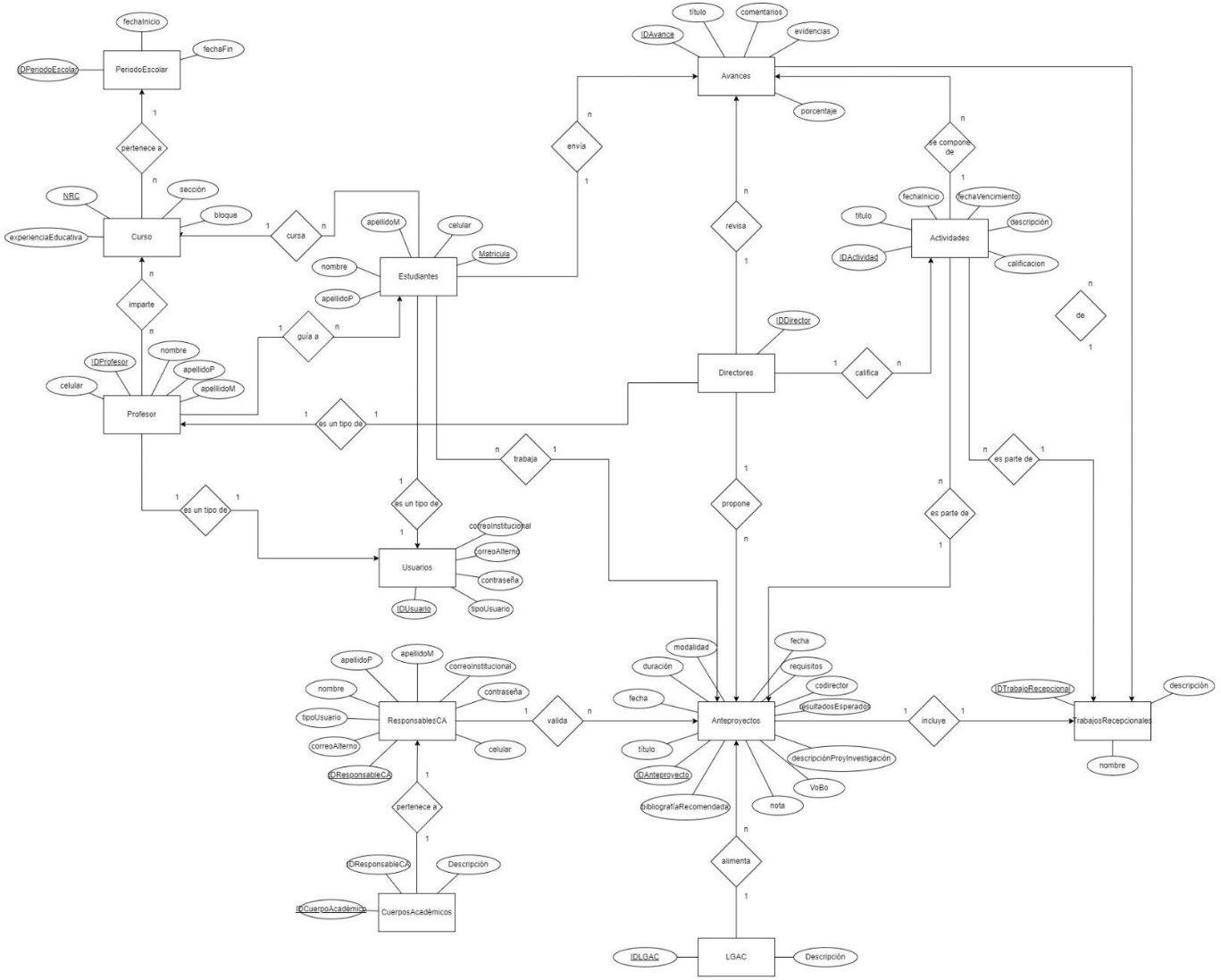

Diagrama Relacional

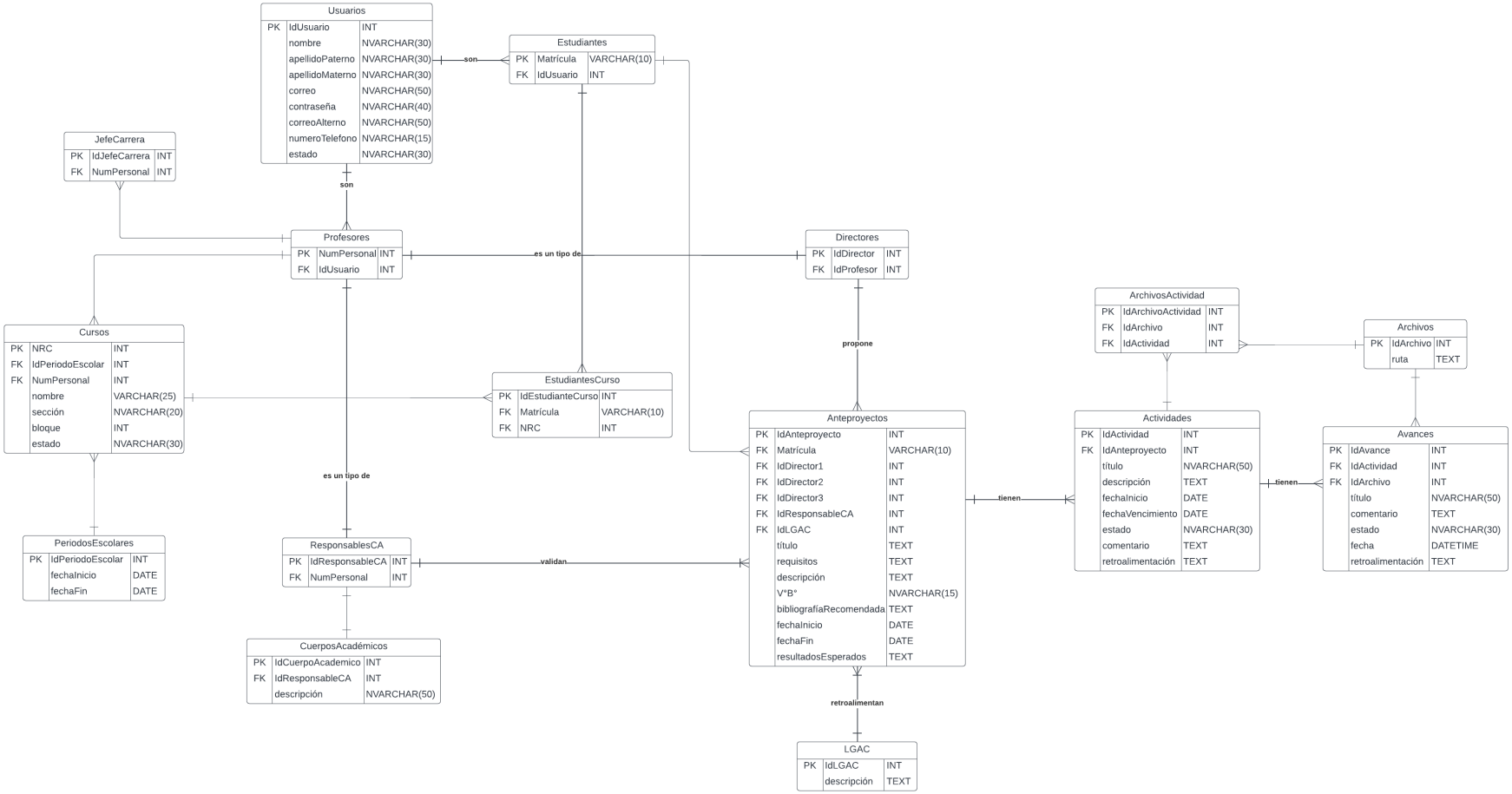

Tabla Pruebas

#Caso de Uso	#Caso de Prueba	Descripción	Entradas	Precondición	Poscondición	Resultado obtenido	Resultado esperado	Estado
CU-01	CP-01	Buscar Usuario Exitosamente	Nombre = “Al”	Debe de haber usuarios registrados en el sistema	Los usuarios deben de ser mostrados en la ventana GUI_USUARIOS	Alonso Andrés	Alonso Andrés	Passed
CU-01	CP-02	Buscar Usuario Erroneamente	Nombre = “F@”	Debe de haber usuarios registrados en el sistema	Los usuarios deben de ser mostrados en la ventana GUI_USUARIOS	Nada	Nada	Passed
CU-01	CP-03	Registrar Usuario con Datos Correctos	Nombre = “José” Apellido Paterno = “Jorge” Apellido Materno = “Herrera” Correo = r@uv.mx Correo Alterno = H@uv.mx Numero De Telefono = 2288478599 Matrícula = zS21938499 Tipo de Usuario = Estudiante	Ninguna	Se debe de registrar el usuario en la base de datos	Registro hecho exitosamente	Registro hecho exitosamente	Passed
CU-01	CP-04	Registrar Usuario con Datos Inválidos	Nombre = “José” Apellido Paterno = “Jorge” Apellido Materno = “Herrera” Correo = r@uv.mx Correo Alterno = H@uv.mx Teléfono = 2288478599 Numero de Personal = zS21043583 Tipo de Usuario = Profesor	Ninguna	No se debe de registrar ningún usuario en la base de datos	Algunos campos contienen datos inválidos	Algunos campos contienen datos inválidos	Passed

CU-01	CP-05	Registrar Usuario con falta de Datos	Nombre = “José” Apellido Paterno = “Jorge” Apellido Materno = null Correo = r@uv.mx Correo Alterno = H@uv.mx Teléfono = 2288478599 Numero de Personal = null Tipo de Usuario = Profesor	Ninguna	No se debe de registrar ningún usuario en la base de datos	Faltan campos por llenar	Faltan campos por llenar	Passed
CU-01	CP-06	Modificar Usuario con datos correctos	Nombre = “José” Apellido Paterno = “Jorge” Apellido Materno = “Herrera” Correo = fdj@uv.mx Correo Alterno = H@uv.mx Numero De Telefono = 2288478889 Matrícula = zS21938499 Tipo de Usuario = Estudiante	El usuario a modificar debe de estar registrado en el sistema	Se debe de modificar el usuario seleccionado.	Modificación hecha exitosamente	Modificación hecha exitosamente	Passed
CU-01	CP-07	Modificar Usuario con datos incorrectos	Nombre = “José” Apellido Paterno = “Jorge” Apellido Materno = “Herrera” Correo = r@uv.mx Correo Alterno = H@uv.mx Numero de teléfono = 2288478599 Numero de Personal = 8499r43 Tipo de Usuario = Profesor	El usuario a modificar debe de estar registrado en el sistema	No se debe modificar el usuario seleccionado	Algunos campos contienen datos inválidos	Algunos campos contienen datos inválidos	Passed
CU-01	CP-08	Modificar Usuario con falta de datos	Nombre = “José” Apellido Paterno = “Jorge” Apellido Materno = “Herrera” Correo = r@uv.mx Correo alterno = H@uv.mx Numero de Personal = null	El usuario a modificar debe de estar registrado en el sistema	No se debe modificar el usuario seleccionado	Faltan campos por llenar	Faltan campos por llenar	Passed

			TipoUsuario = Profesor					
CU-02	CP-01	Agregar estudiante a curso exitosamente	N/A	El estudiante a agregar no debe de estar registrado en algún curso	El curso al que fué agregado el estudiante debe de aparecer en el menú principal del estudiante en cuestión	N/A	N/A	Passed
CU-02	CP-02	Agregar estudiante a curso excepción	N/A	El estudiante a agregar no debe de estar registrado en algún curso	La excepción es lanzada correctamente	DataInsertion Exception	DataInsertion Exception	Passed
CU-02	CP-03	Dar de baja usuario de curso exitosamente	N/A	El estudiante a eliminar debe de estar registrado en el curso	El estudiante ya no tiene acceso al curso en donde estaba registrado	N/A	N/A	Passed
CU-02	CP-04	Dar de baja usuario de curso excepción	N/A	El estudiante a eliminar debe de estar registrado en el curso	La excepción es lanzada correctamente	DataInsertion Exception	DataInsertion Exception	Passed
CU-03	CP-01	Buscar Curso Exitosamente	NRC = “43”	Debe de haber cursos registrados en el sistema	Los cursos deben de ser mostrados en la ventana GUI_USUARIOS	Alonso Andrés	Alonso Andrés	Passed
CU-03	CP-02	Buscar Curso Erroneamente	NRC = “AL”	Debe de haber cursos registrados en el sistema	Los cursos deben de ser mostrados en la ventana GUI_USUARIOS	Nada	Nada	Passed
CU-03	CP-03	Registrar Curso con Datos Correctos	Periodo Escolar = 2023-06-28 2023-02-07 Profesor = Fernando Juan Gonzalez NRC = “45637” Bloque: “8”	Ninguna	Se debe de registrar el curso en la base de datos	Registro hecho exitosamente	Registro hecho exitosamente	Passed

			Sección: “2”					
CU-03	CP-04	Registrar Curso con Datos Inválidos	Periodo Escolar = 2023-06-28 2023-02-07 Profesor = Fernando Juan Gonzalez NRC = “abcdef” Bloque: “8” Sección: “2”	Ninguna	No se debe de registrar ningún curso en la base de datos	Algunos campos contienen datos inválidos	Algunos campos contienen datos inválidos	Passed
CU-03	CP-05	Registrar Curso con falta de Datos	Periodo Escolar = 2023-06-28 2023-02-07 Profesor = Fernando Juan Gonzalez NRC = null Bloque: “8” Sección: “2”	Ninguna	No se debe de registrar ningún curso en la base de datos	Faltan campos por llenar	Faltan campos por llenar	Passed
CU-03	CP-06	Modificar Curso con datos correctos	Periodo Escolar = 2023-06-28 2023-02-07 Profesor = Humberto Guzman Gutierrez NRC = “47953” Bloque: “7” Sección: “2”	El curso a modificar debe de estar registrado en el sistema	Se debe de modificar el curso seleccionado.	Modificación hecha exitosamente	Modificación hecha exitosamente	Passed
CU-03	CP-07	Modificar Curso con datos incorrectos	Periodo Escolar = 2023-06-28 2023-02-07 Profesor = Humberto Guzman Gutierrez NRC = “YAM02” Bloque: “7” Sección: “2”	El curso a modificar debe de estar registrado en el sistema	No se debe modificar el curso seleccionado	Algunos campos contienen datos inválidos	Algunos campos contienen datos inválidos	Passed
CU-03	CP-08	Modificar Curso con falta de datos	Periodo Escolar = 2023-06-28 2023-02-07 Profesor = Humberto Guzman Gutierrez NRC = null Bloque: “7” Sección: “2”	El curso a modificar debe de estar registrado en el sistema	No se debe modificar el curso seleccionado	Faltan campos por llenar	Faltan campos por llenar	Passed

CU-04	CP-01	Guardar Avance en BD Exitosamente.						Passed*
CU-04	CP-02	Guardar Avance en BD Error.						Passed*
CU-04	CP-03	Guardar Avance en BD Excepción.	N/A					Passed
CU-04	CP-04	Guardar Archivo en BD Exitosamente.						Passed
CU-04	CP-05	Guardar Archivo en BD Error.						Passed*
CU-04	CP-06	Guardar Archivo en BD Excepción.	N/A					Passed
CU-05	CP-01	Modificar Avance en BD Exitosamente.						Passed
CU-05	CP-02	Modificar Avance en BD Error.						Passed
CU-05	CP-03	Modificar Avance en BD Excepción.	N/A					Passed
CU-05	CP-04	Recuperar Archivo de la BD Exitosamente.						Passed
CU-05	CP-05	Recuperar Archivo de la BD Error.						Passed*
CU-05	CP-06	Recuperar Archivo de la BD Excepción.	N/A					Passed
CU-06	CP-01	Retroalimentar actividad Exitosamente	Retroalimentación = “Bien hecho, tiene 8”	La base debe estar activa	La retroalimentación fue guardada exitosamente	ColumnasAfectadas > 0	ColumnasAfectadas > 0	Passed
CU-06	CP-02	Retroalimentar actividad Excepción	Retroalimentación = “Retroalimentación excepción”	La base debe estar apagada	La excepción se lanzó correctamente	DataInsertion Exception	DataInsertion Exception	Passed
CU-07	---							

CU-08	CP-01	Crear actividad Exitosamente	Título = "Add Activity Test" Descripción = "This activity is made to test ActivityDAO.addActivity" FechaInicio = 26/03/2023 FechaFin = 02/12/2023	La base debe estar activa	La actividad fue registrada exitosamente.	IdGenerado > 0	IdGenerado > 0	Passed
CU-08	CP-02	Crear actividad Excepción	N/A	La base debe estar apagada	La excepción se lanzó correctamente	DataInsertion Exception	DataInsertion Exception	Passed
CU-09	CP-01	Obtener lista de actividades Exitosamente	Id anteproyecto precargado	Hay al menos 1 actividad asignada al anteproyecto	N/A	Lista no vacía	Lista no vacía	Passed
CU-09	CP-02	Obtener lista de actividades Excepción	N/A	La base debe estar apagada	La excepción se lanzó correctamente	DataRetrieval Exception	DataRetrieval Exception	Passed
CU-10	CP-01	Modificar actividades Exitosamente	Título = “Modified activity Test” Descripción = “This is a modification of the preloaded activity” FechaInicio = “12/05/2023” FechaFin = “19/05/2023”	Debe haber conexión con la base de datos	Los cambios a la actividad fueron guardados exitosamente	ColumnasAfectadas > 0	ColumnasAfectadas > 0	Passed
CU-10	CP-02	Modificar actividades Excepción	N/A	La base debe estar apagada	La excepción se lanzó correctamente	DataInsertion Exception	DataInsertion Exception	Passed
CU-11	CP-01	Retroalimentar actividades Exitosamente	Retroalimentación = “Bien hecho, pasó en extra”	El Id de la actividad debe ser válido	N/A	ColumnasAfectadas > 0	ColumnasAfectadas > 0	Passed
CU-11	CP-02	Retroalimentar actividades Excepción	N/A	La base debe estar apagada	La excepción se lanzó correctamente	Data Insertion Exception	Data Insertion Exception	Passed
CU-12	CP-01	Guardar Archivo de Actividad Exitosamente.						Passed
CU-12	Cp-02	Guardar Archivo de Actividad Error.	N/A					Passed

CU-12	CP-03	Guardar Archivo de Actividad Excepción.						Passed
CU-12	CP-04	Recuperar Archivos de Actividad Éxitosamente.						Passed
CU-12	CP-05	Recuperar Archivos de Actividad Error.						Passed
CU-12	CP-06	Recuperar Archivos de Actividad Excepción.	N/A		La excepción se lanzó correctamente	DataRetrieval Exception	DataRetrieval Exception	Passed
CU-14	CP-01	Crear anteproyecto exitosamente	Título = “Anteproyecto de prueba para el método addResearch()” Descripción = “Este es un anteproyecto de prueba para el ResearchDAOTest” Resultado esperado = “Se espera que el anteproyecto devuelva el Id del proyecto generado” Requisitos = “Para lograrlo se necesitan hacer buenas pruebas” BibliografíaRecomendada = “APA”			Id generado > 0	Id generado > 0	Passed
CU-14	CP-02	Crear anteproyecto Excepción	N/A	La base debe estar apagada	N/A	DataInsertion Exception	DataInsertion Exception	Passed
CU-14	CP-03	Consultar anteproyecto exitosamente	N/A					Passed

CU-14	CP-04	Consultar anteproyecto Excepción	N/A	La base debe estar apagada	N/A	DataRetrieval Exception	DataRetrieval Exception	Passed
CU-14	CP-05	Modifcar anteproyecto exitosamente	Título = “” Descripción = “” Resultado esperado = “” Requisitos = “” BibliografíaRecomendada = “”			AffectedRows > 0	AffectedRows > 0	Passed
CU-14	CP-06	Modificar anteproyecto excepción	N/A	La base debe estar apagada	N/A	DataInsertion Exception	DataInsertion Exception	Passed
CU-15	CP-01	Validar anteproyectos Exitosamente	N/A	El anteproyecto debe de estar en estado “Propuesto”	El estado del anteproyecto pasa a “Validado”	Anteproyecto validado exitosamente	Anteproyecto validado exitosamente	Passed
CU-15	CP-02	Validar anteproyectos excepción	N/A	La base debe estar apagada	N/A	DataInsertion Exception	DataInsertion Exception	Passed
CU-16	CP-01	Buscar todos los Anteproyectos Exitosamente	Título = “N”	Deben de haber Anteproyectos registrados en la base de datos	Los anteproycetos deben de ser mostrados en la ventana GUI_REPORTE_ANTEPROYECTO	Mostrar anteproyectos en la ventana GUI_REPORTE_ANTEPROYECTO	Los anteproyectos se muestran en l ventana GUI_REPORTE_ANTEPROYECTO	Passed
CU-16	CP-02	Buscar todos los Anteproyectos con errores	Título = “4”	Deben de haber Anteproyectos registrados en la base de datos	Los anteproycetos deben de ser mostrados en la ventana GUI_REPORTE_ANTEPROYECTO	Nada	Nada	Passed
CU-16	CP-03	Buscar todos los Anteproyectos	Título = “N”	Deben de haber Anteproyectos validados	Los anteproycetos deben de ser mostrados en la	Mostrar anteproyectos en la	Los anteproyectos se muestran	Passed

		validados Exitosamente		registrados en la base de datos	ventana GUI_REPORTE_ANTEPROYECTO	ventanaGUI_REPORTE_ANTEPROYECTO	en l ventana GUI_REPORTE_ANTEPROYECTO	
CU-16	CP-04	Buscar todos los Anteproyectos validados con errores	Título = "|13"	Deben de haber Anteproyectos validados registrados en la base de datos	Los anteproycetos deben de ser mostrados en la ventana GUI_REPORTE_ANTEPROYECTO	Nada	Nada	Passed
CU-16	CP-05	Buscar todos los Anteproyectos no validados Exitosamente	Título = "Rec"	Deben de haber Anteproyectos no validados registrados en la base de datos	Los anteproycetos deben de ser mostrados en la ventana GUI_REPORTE_ANTEPROYECTO	Mostrar anteproyectos en la ventanaGUI_REPORTE_ANTEPROYECTO	Los anteproyectos se muestran en l ventana GUI_REPORTE_ANTEPROYECTO	Passed
CU-16	CP-06	Buscar todos los Anteproyectos no validados con errores	Título = "@3"	Deben de haber Anteproyectos no validados registrados en la base de datos	Los anteproycetos deben de ser mostrados en la ventana GUI_REPORTE_ANTEPROYECTO	Nada	Nada	Passed
CU-16	CP-07	Buscar todos los Anteproyectos validados y no validados Exitosamente	Título = "N"	Deben de haber Anteproyectos validados y no validados registrados en la base de datos	Los anteproycetos deben de ser mostrados en la ventana GUI_REPORTE_ANTEPROYECTO	Mostrar anteproyectos en la ventanaGUI_REPORTE_ANTEPROYECTO	Los anteproyectos se muestran en l ventana GUI_REPORTE_ANTEPROYECTO	Passed
CU-16	CP-08	Buscar todos los Anteproyectos	Título = "6"	Deben de haber Anteproyectos validados y no	Los anteproycetos deben de ser mostrados en la	Nada	Nada	Passed

		validados con errores		validados registrados en la base de datos	ventana GUI_REPORTE_ANTEPROYECTO			
CU-16	CP-09	Generar Reporte Exitosamente	Título = “Anteproyectos“	Deben de haber Anteproyectos registrados en la base de datos	Los anteproycetos deben de ser mostrados en la ventana GUI_REPORTE_ANTEPROYECTO	Mostrar anteproyectos en la ventanaGUI_REPORTE_ANTEPROYECTO	Los anteproyectos se muestran en l ventana GUI_REPORTE_ANTEPROYECTO	Passed
CU-16	CP-10	Generar Reporte con Errores.	N/A	Deben de haber Anteproyectos registrados en la base de datos	Los anteproycetos deben de ser mostrados en la ventana GUI_REPORTE_ANTEPROYECTO	Ventana de alerta con mensaje de fallo	Ventana de alerta con mensaje de fallo	Passed
CU-17	CP-01	Guardar LGAC en BD Exitosamente						Passed
CU-17	CP-02	Guardar LGAC en BD Error.						Passed
CU-17	CP-03	Guardar LGAC en BD Excepción.	N/A			Data Insertion Exception	Data Insertion Exception	Passed
CU-17	CP-04	Recuperar Lista de LGACs de la BD Exitosamente.						Passed
CU-17	CP-05	Recuperar Lista de LGACs de la BD Error.						Failed
CU-17	CP-06	Recuperar Lista de LGACs de la BD Excepción.	N/A	La base debe estar apagada	N/A	Data Retrieval Exception	Data Retrieval Exception	Passed
CU-17	CP-07	Recuperar Lista de LGACs por						Passed

		Descripción Exitosamente.						
CU-17	CP-08	Recuperar Lista de LGACs por Descripción Error.						Passed
CU-17	CP-09	Recuperar Lista de LGACs por Descripción Excepción.	N/A			Data Retrieval Exception	Data Retrieval Exception	Passed
CU-17	CP-10	Modificar Descripción de LGAC Exitosamente						Passed
CU-17	CP-11	Modificar Descripción de LGAC Error						Passed
CU-17	CP-12	Modificar Descripción de LGAC Excepción	N/A			Data Insertion Exception	Data Insertion Exception	Passed *
N/A	CP-01	Login Profesor exitosamente	Usuario = “luicuellar” Contraseña = “password”	El usuario existe en la base	N/A	DatosUsuario != null	DatosUsuario != null	Passed
N/A	CP-02	Login Profesor fail	Usuario = “luicuellar” Contraseña = “contraseña”	El usuario no existe en la base	N/A	DatosUsuario == null	DatosUsuario == null	Passed
N/A	CP-03	Login Profesor excepción	N/A	La base está apagada	N/A	Login Exception	Login Exception	Passed
N/A	CP-04	Login Director exitosamente	Usuario = “ezacosta” Contraseña = “pensamientoDeCampirán2020”	El usuario existe en la base	N/A	DatosUsuario != null	DatosUsuario != null	Passed*
N/A	CP-05	Login Director fail	Usuario = “ezacosta” Contraseña = “habilidadesDePensamiento”	El usuario no existe en la base	N/A	DatosUsuario == null	DatosUsuario == null	Passed
N/A	CP-06	Login Director excepción	N/A	La base está apagada	N/A	Login Exception	Login Exception	Passed

N/A	CP-07	Login EncargadoCA exitosamente	Usuario = “umarinero” Contraseña = “programaciónOOP5462”	El usuario existe en la base	N/A	DatosUsuario != null	DatosUsuario != null	Passed
N/A	CP-08	Login EncargadoCA fail	Usuario = “umarinero” Contraseña = “prograee24”	El usuario no existe en la base	N/A	DatosUsuario == null	DatosUsuario == null	Passed
N/A	CP-09	Login EncargadoCA excepción	N/A	La base está apagada	N/A	Login Exception	Login Exception	Passed
N/A	CP-10	Login JefeCarrera exitosamente	Usuario = “jorocharan” Contraseña = “vivaElDiseño2023”	El usuario existe en la base	N/A	DatosUsuario != null	DatosUsuario != null	Passed
N/A	CP-11	Login JefeCarrera fail	Usuario = “jorocharan” Contraseña = “viv34567”	El usuario no existe en la base	N/A	DatosUsuario == null	DatosUsuario == null	Passed
N/A	CP-12	Login JefeCarrera excepción	N/A	La base está apagada	N/A	Login Exception	Login Exception	Passed
N/A	CP-13	Login Estudiante exitosamente	Usuario = “zS23587532”” Contraseña=“MeLoveCats1234”	El usuario existe en la base	N/A	DatosUsuario != null	DatosUsuario != null	Passed
N/A	CP-14	Login Estudiante fail	Usuario = “zS23587532” Contraseña = “Xavier0573”	El usuario no existe en la base	N/A	DatosUsuario == null	DatosUsuario == null	Passed
N/A	CP-15	Login Estudiante excepción	N/A	La base está apagada	N/A	Login Exception	Login Exception	Passed

Reporte de pruebas

#Caso de Uso	#Caso de Prueba	Problemas / Correcciones
CU01	CP-04	Algunas expresiones regulares eran incorrectas, puesto que algunos datos los daban como válidos. Se cambiáron las expresiones regulares para que no fallara.
CU-03	CP-03	
CU-04	CP-01	El método no funcionaba si no se vinculaba un archivo al avance. Se incluyo un setNull para cuando el avance no tenga un archivo adjunto.
CU-04	CP-02	El método se rompía al recibir un avance que tenía una idActividad == 0. Ahora lanza una excepción personalizada a la capa gráfica para comunicar el problema al usuario.
CU-04	CP-05	Se corrigió el método para que pueda detectar Strings vacíos y lanzar una excepción a la capa gráfica.
CU-05	CP-05	Se cambió el método para lanzar una excepción a la capa gráfica cuando no se encuentre el archivo.

CU-17	CP-12	Se regresa la excepción esperada pero con un mensaje diferente. Solucionado, era un error al llamar al método con ID nula, lo que lanza primero la excepción de “ID Inválida”, por lo que nunca se lanzaba la SQLException.
N/A	CP-02	La prueba pasaba pero borraba el usuario incorrecto porque me equivoqué al recuperar el Id de usuario generado y puse 1 como Id en lugar de ResultSet.getInt(1) de las generated keys.

CONCLUSIONES

Este ha sido sin duda alguna un proyecto bastante complejo y serio para el nivel en el que nos encontramos ahora, puesto que es la primera vez en la que desarrollamos un sistema en serio, con toda su documentación asociada y con cada una de las cosas que hay que tomar en cuenta a la hora de desarrollar software, como validaciones, investigación de tecnologías para implementar funcionalidades, cumplimiento de un standar, etc.

Todo esto ha sido un proceso complicado y con muchos problemas en el camino, debido a que el ritmo que hemos seguido nos ha traido ventajas como el de poder adaptarnos un poco a la presión y desventajas como el hecho de que no podamos realizar ciertas cosas debido a que como muchisimas de las cosas las tenemos que aprender.

Entrando más a detalle en el desarrollo del sistema, hemos tenido que aprender muchisimas tecnologias, como junit, javafx, jasperreports, y mucha documentación de java para poder implementar lo que se nos pedía.

A diferencia del diseño, en la construcción muchas veces nos sentiamos más seguros de lo que estabamos haciendo, puesto que lo que más nos hace sentir seguridad a la hora de realizar las cosas es la comprobación, en este caso la comprobación se realizaba a la hora de correr el sistema, si funcionaba significaba que estaba bien implementado (por la parte de la funcionalidad), y además por que cosas como seguir el estandar de codificación y entender la teoria de rutinas de calidad y mejores prácticas no nos costó tanto digerir, en cambio con diseño cuando realizabamos cosas nos quedabamos con un sabor de boca agridulce, puesto que no sabiamos si lo que estabamos haciendo estaba bien, por que no hay forma de comprobarlo, y muchas veces nos entraban ganas de codificar primero y luego hacer el diseño, puesto que como ya tenemos comprobado las implementaciones, ya al hacer los diagramas los haciamos con más seguridad.

Esta sin duda alguna ha sido una experiencia inolvidable puesto que ha sido un desafio tremendo, el proyecto para el nivel que tenemos nosotros lo consideramos de una dificultad entre algo elevada puesto que nunca habiamos desarrollado software siguiendo una metodología de desarrollo y teniendo que aprender todo un aspecto nuevo de un lenguaje que habíamos aprendido de manera superficial en Experiencias Educativas anteriores.

REFERENCIAS

- Baquero, L. E. & Hernández, M. A. (2016). *Buenas prácticas de programación orientada a objetos en Java*.
- Züllighoven, H., Beeger, R. F., Bäumer, D., & ScienceDirect (Online service). (2005). *Object-oriented construction handbook: Developing application-oriented software with the tools & materials approach*. Elsevier ; Morgan Kaufmann ; Dpunkt.verlag.
- James Gosling, Bill Joy, Guy L. Steele, Gilad Bracha, and Alex Buckley. 2014. The Java Language Specification, Java SE 8 Edition (1st. ed.). Addison-Wesley Professional.

- Arnold, K., Gosling, J., & Holmes, D. (2005). *The Java programming language*. Addison Wesley Professional.
- Liang, Y. Daniel. (2015). *Introduction to java programming :* brief version. Boston :Pearson,
- GAMMA, Erich. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional. (1998).
- MCCONNELL, S. Code Complete: A Practical Handbook of Software Construction, Microsoft Press. 2004
- Robert C. Martin. (2008). *Clean Code: A Handbook of Agile Software Craftsmanship* (1st. ed.). Prentice Hall PTR, USA.
- McConnell, S. (2004). *Code Complete, Second Edition*. Redmond, WA, USA: Microsoft Press.
- Steve McConnell. (1996). *Rapid Development: Taming Wild Software Schedules* (1st. ed.). Microsoft Press, USA.
- Bloch, Joshua. (2001). *Effective Java Programming Language Guide.* Boston: Addison-Wesley.