

“Universidad Autónoma de Nuevo León”

Facultad de Ingeniería Mecánica y Eléctrica

Administración de base de datos

Documentación del proyecto

EQUIPO 4	
Matricula	Nombre
1858984	Jenifer Alejandra Rodríguez Méndez
1815466	Alondra Monserrat Ramírez Proa
1808135	Cynthia Guadalupe Vega Villalba
1543003	Selene Aydee Abrego Bonilla
1802224	Maricarmen Regalado Espinoza
1733222	Fernando Uriel Zapata Rivas
1747264	Roberto Alan Rodríguez Monroy
1461955	José Alberto Vázquez Martínez
1578267	Juan Carlos Romero Ortiz
1811331	Braulio Rafael Torrez Ruiz
1626560	Diego Alan Sánchez Partida
1603039	William Gilberto Vargas Delgado

Salón: 4206 Grupo: 002

Docente: Dra. Norma Edith Marín Martínez

Día: Martes Hora: N1-N3

Semestre Agosto-Diciembre 2021

Contenido

Cliente.....	6
Nombre del proyecto.....	6
Objetivo del proyecto	6
Justificación	6
Requerimientos del cliente.....	7
Investigación.....	9
Aula 1	11
BEST (Bullet Education Scheduling and Timetabling).....	11
DocCF	12
Esemtia de Edebé	12
FET.....	13
Generador de horarios de ARASAAC	13
Psicología del color.....	14
¿Cómo aplicamos la psicología del color a nuestro proyecto?.....	14
Paletas de colores	15
Diseño de prototipos.....	17
Especificación de software	19
Requisitos de hardware.	19
Especificaciones generales	19
Especificaciones técnicas detalladas	20
Diseño de la base de datos	20
Normalización de la base de datos.....	20
Modelo conceptual del sistema.....	24
Diagrama UML.....	25

Diccionario de datos	26
Interfaces gráficas (entrada, salida, combinadas)	28
Interfaz: Inicio de sesión	28
Interfaz: Visualización.....	29
Interfaz: CRUD	30
Herramientas CASE.....	32
Casos de uso	33
Conclusiones del proyecto.....	36
Pronóstico del tiempo	37
Anexos.....	38
Manual técnico	38
Propósito	38
Alcance	38
Programas utilizados.....	38
Construcción del proyecto.....	39
Manual de usuario	48
Inicio de sesión	48
Menú de opciones	48

Índice de Imágenes

Tabla 1. Tabla de Requerimientos.....	8
Ilustración 1. Logo ascHorarios.	9
Ilustración 2. Precios de ascHorarios.	10
Ilustración 3. AscTimetables.	10
Ilustración 4. Logo aula1.....	11
Ilustración 5. Logo BEST.	11
Ilustración 6. Logo DocCF.	12
Ilustración 7. Interfaz Esemtia de Edebé.	12
Ilustración 8. FETA.	13
Ilustración 9. Interfaz ARASAAC.	13
Ilustración 10. Paleta de colores.....	16
Ilustración 11. Prototipo pantalla Inicio de sesión.	17
Ilustración 12. Prototipo pantalla Visualización.....	18
Ilustración 13. Interfaz ARASAAC.	19
Tabla 2. Tabla de Usuarios.....	20
Tabla 3. Tabla de Clases.	21
Tabla 4. Tabla Materia.	21
Tabla 5. Ejemplo de la tabla clase llena.....	21
Tabla 6. Ejemplo de tabla materia llena.....	22
Tabla 7. Ejemplo de tabla usuario llena.	22
Tabla 8. Ejemplo de tabla agrupación llena.	22
Ilustración 14. Relación entre los campos de la tabla materia.	22
Ilustración 15. Relación entre los campos de la tabla clase.....	23

Ilustración 16. Relación entre los campos de la tabla usuario.	23
Ilustración 17. Relación de los campos en la tabla agrupación.	24
Ilustración 18. Modelo conceptual.	24
Ilustración 19. Diagrama UML.	25
Tabla 9. Diccionario de datos tabla materia.	26
Tabla 10. Diccionario de datos tabla clase.	26
Tabla 11. Diccionario de datos tabla agrupación.	27
Tabla 12. Diccionario de datos tabla usuario.	27
Ilustración 20. Pantalla de inicio de sesión.	28
Ilustración 21. Pantalla interfaz de Visualización de datos.	29
Ilustración 22. CRUD modo Create.	30
Ilustración 23. CRUD modo Update.	31
Tabla 13. Herramientas CASE.	32
Tabla 14. Tabla del primer caso de uso.	33
Tabla 15. Tabla del segundo caso de uso.	33
Tabla 16. Tabla del tercer caso de uso.	34
Tabla 17. Tabla del cuarto caso e uso.	35
Tabla 18. Tabla del quinto caso de uso.	35
Tabla 19. Sexto caso de uso.	35
Ilustración 24. Diagrama de Gantt previsto.	37
Ilustración 25. Diagrama de Gantt en tiempo real.	37
Ilustración 26. Abreviación de nombres asignados al cronograma.	38
Ilustración 27. Pantalla de inicio de sesión.	39
Ilustración 28. Código de función: ingreso().	40
Ilustración 29. Pantalla interfaz de Visualización de datos.	40

Ilustración 30. Código de la función: click_treeview.....	41
Ilustración 31. Código de la función seleccion_del_combobox.....	42
Ilustración 32. Código de la función mostrarTodo.....	42
Ilustración 33. Código de función ir_agregar.	43
Ilustración 34. CRUD modo Create.	43
Ilustración 35. Código de función agregar_elementos.	44
Ilustración 36. Código de función eliminar_registro	45
Ilustración 37. Código función actualizar.	45
Ilustración 38. Código de función crear_grupos (parte1).	46
Ilustración 39. Código de función crear_grupos (parte 2).	46
Ilustración 40. Código de función crear_grupos (parte3).	47
Ilustración 41. Pantalla inicio de sesión.	48
Ilustración 42. Pantalla Visualización.....	49
Ilustración 43. Opciones de filtro.....	49
Ilustración 44. Tabla de grupos dentro de la aplicación.....	50
Ilustración 45. Tabla de las materias que constituyen la agrupación.....	51
Ilustración 46. Funciones de la pantalla Visualizar.	51
Ilustración 47. Funciones de la pantalla CRUD (parte 1).	52
Ilustración 48. Funciones de la pantalla CRUD (parte 2).	53
Ilustración 49. Función de la pantalla Actualizar.....	53

Cliente

Dr. Jesús Adolfo Meléndez Guevara.

Nombre del proyecto

Paquetes de materias FIME.

Objetivo del proyecto

Este proyecto se elabora con la finalidad de satisfacer la necesidad de la Coordinación de Administración y Sistemas de la Facultad de Ingeniería Mecánica y Eléctrica de tener control de los grupos, organizándose en paquetes de materias por semestre. Dando así al coordinador una opción para poder elegir horarios que sean adecuados para los semestres próximos.

Justificación

Debido a la problemática de la distribución de horarios en relación con los paquetes disponibles, se ha tomado la decisión de crear un sistema que pueda resolver el inconveniente que actualmente se presenta. El realizar los paquetes de materias nos permite visualizar los espacios disponibles en los que se pueden agregar las materias de otra coordinación, logrando así tener una mejor distribución en los horarios.

Esto permite al coordinador, el cual es el usuario de la aplicación, tomar la decisión de en caso de que algunos de los paquetes de materias se encuentren con grupos sin asignación se le facilite decidir si se abren más grupos para completar esos paquetes de materias, o en otro caso, eliminarlos.

Requerimientos del cliente

Módulo	Componente	Requerimiento	Descripción
Interfaz	General	Diseño del proyecto	Diseñar un CRUD con los colores de la Facultad de Ingeniería Mecánica y Eléctrica.
Acceso	Login	Ingresar al sistema	Un único usuario para el coordinador.
Opciones	Menú	Menú principal	Desarrollar un menú principal donde se muestran todas las opciones principales de la aplicación con los requerimientos que se especificaron.
		Mostrar Grupos	Desarrollar un componente para el menú principal que contenga el detalle de la información de los grupos creados con el nombre "Vista de Grupos".
		Mostrar Materias del Grupo	Desarrollar un componente el cual recibe y muestra la información del grupo seleccionado en el componente Vista de Grupos.
		Filtro con lista desplegable	Implementar en el menú principal un filtro con los parámetros "Carrera", "Semestre" y "Turno" los cuales pueden ser seleccionados por medio de una lista desplegable para facilitar y agilizar el proceso de búsqueda.
		Botón para mostrar toda la información	Implementar un botón con el nombre "Mostrar Todo" con la función de limpiar el filtro de búsqueda y mostrar los datos almacenados.
		Botón para ingresar al modo "Edición"	Implementar un botón con el nombre "Editar Grupo" que tiene la función de mostrar una ventana nueva donde se puede editar la información de los grupos.
	Edición de Grupos	Agregar Grupo	Desarrollar un componente que tenga la funcionalidad de Agregar Grupos y contenga los parámetros "Plan", "Clave" del grupo, "Materia", "Empleado", "Salón", "Carrera", "Hora", "Semestre" y "Día".

Módulo	Componente	Requerimiento	Descripción
Opciones	Edición de Grupos	Mostrar materias registradas	Desarrollar un componente que recibe y muestra la información de las materias que se están almacenando en la base de datos
		Editar registro	Implementar un botón que tenga la funcionalidad de editar el registro que se tiene seleccionado en el componente de “Materias Registradas” y muestre una ventana emergente con el nombre “Actualizar registro” y reciba todos los datos del registro seleccionado para editar.
		Eliminar registro	Implementar un botón que tenga la funcionalidad de eliminar el registro que se tiene seleccionado en el componente de “Materias Registradas”
		Actualizar Grupos	Implementar un botón que después de ingresar materias nuevas a la base de datos, al presionarlo actualice los grupos con la nueva información.
		No repetir horas	Implementar en la lógica de la aplicación la funcionalidad que no permita al usuario guardar materias que tengan la misma hora en que se imparten, y muestre una advertencia.
		Botón de regresar	Implementar un botón con el nombre “Menú Principal” que permite al usuario regresar al menú.

Tabla 1. Tabla de Requerimientos.

Investigación

Se realizó una investigación sobre diversas aplicaciones que son utilizadas en la administración horaria, las cuales actualmente se encuentran disponibles en el mercado, esto con la finalidad de tener un punto de comparación con el sistema que se está desarrollando y tomar en cuenta que aspectos se pueden fortalecer y, por lo tanto, crear un sistema que sea capaz de satisfacer las necesidades del cliente.

ascHorarios



Ilustración 1. Logo ascHorarios.

Es un Software que realiza planificaciones introduciendo los requisitos, para evaluar más de 5,000,000 de posibilidades para obtener un horario bien equilibrado, si alguien se equivoca o hizo cambios, no es necesario volver a hacerlo todo, el software lo reprogramará, se adaptará al instante y automáticamente a todas las modificaciones. Cuando esté satisfecho con el nuevo horario de ascTimetables, se puede imprimir para cada clase, para cada profesor, e incluso distribuir copias personalizadas para alumnos específicos. Puedes personalizar la fuente, el diseño y los logotipos así como también exportarlo en formato pdf, Excel o compartirlo online.

Dentro de sus funciones encontramos lo siguiente:

- Generación automática
- Ajustes manuales
- Verificación del horario
- Introducción de datos sencilla
- Acceso Móvil
- Importación de datos en formato electrónico.
- Compatible con clases en distintos edificios
- Totalmente personalizable para funciones individuales a su gusto

- Herramienta útil para programar las sustituciones de profesores, completa con notificaciones e impresión. Varios usuarios pueden planificar al mismo tiempo sus sustituciones o cambios de turnos en cualquier momento
- Crea sitios web para su programa de estudio
- Adaptable a su región
- Libro de texto electrónico

Los costos que tiene el software son los siguientes:

	USD 250 one time payment	USD 350 one time payment	USD 500 one time payment	USD 1995 one time payment
	PEDIR	PEDIR	PEDIR	PEDIR
Generador automático	✓	✓	✓	✓
Licencia múltiple para todos los ordenadores del centro	✓	✓	✓	✓
ascSubstitutions	✓	✓	✓	✓
Horarios móviles Alumnos y profesores tendrán acceso a los horarios actualizados desde sus teléfonos o tablets	✓	✓	✓	✓
Asistencia ilimitada Asistencia y actualizaciones durante dos años	✓ after 2nd year: USD 79	✓ after 2nd year: USD 79	✓ after 2nd year: USD 79	✓ after 2nd year: USD 79
Asistencia Premium Análisis, prueba, revisión y sugerencias de mejora para su horario			✓	✓
Generador automático en función del horario de alumnos Crea horarios individualizados para los alumnos				✓

Ilustración 2. Precios de ascHorarios.

AscTimetables

The screenshot shows the AscTimetables software interface. At the top, there is a search bar labeled ':Find'. Below it are four buttons: '?Questions Comments? Write us', 'School', 'Verification', and 'Generate new'. The main part of the interface is a grid representing a weekly timetable. The grid is divided into two sections: 'Wednesday' on the left and 'Thursday' on the right. Each section has a header row with days of the week (S, M, T, W, T, F, S) and a grid of colored boxes containing subject abbreviations. For example, on Wednesday, the subjects include ART, ENG, ISL, SOS, and ENG. On Thursday, the subjects include ENG, MAT, SOS, ARA, SCE, and ENG.

Ilustración 3. AscTimetables.

Sólo es necesario introducir los requisitos y el software de planificación evaluará todas las posibilidades para obtener un horario equilibrado. Y ante errores o cambios, esta herramienta programará automáticamente todas las modificaciones. Entre sus ajustes manuales, es posible personalizar la fuente, el diseño y los logotipos, y exportarlo en formato pdf, Excel o compartirlo online.

Aula 1



Ilustración 4. Logo aula1.

El software de gestión escolar de Aula1 dispone de un generador de horarios integrado dentro de la plataforma que permite crear varios tipos de horarios. Por ejemplo, los docentes podrán crear distintos horarios según los niveles académicos en los que trabajan. También permite consultar los horarios incompletos en los que queden horas o materias sin asignar y comprobar los errores en las materias por distintos colores.

BEST (Bullet Education Scheduling and Timetabling)



Ilustración 5. Logo BEST.

Este software permite a los centros educativos superar la complejidad y lentitud de realizar la programación de los horarios del curso. Para ello, combina y evalúa varias restricciones y objetivos, de acuerdo con las necesidades de los

colegios, optimizando los horarios de profesores, estudiantes y aulas. Además, asigna automáticamente las salas disponibles y próximamente también será capaz de planificar los exámenes de forma automática. Este algoritmo fue desarrollado hace 15 años en Portugal y hoy en día tiene ya cerca de 200 clientes, algunos de ellos en España.

DocCF



Ilustración 6. Logo DocCF.

Se trata de un software de gestión académica y administrativa. Con él se pueden llevar a cabo cerca de 60 procedimientos distintos y entre ellos está la creación y asignación de horarios para las distintas clases. Ante cualquier problema, cuenta con un sistema de atención al cliente online a través de un chat, y plataformas como Skype, Facebook Messenger o WhatsApp.

Esemtia de Edebé



Ilustración 7. Interfaz Esemtia de Edebé.

Entre la gran variedad de herramientas que propone esta plataforma de gestión destinada a los cursos desde Infantil hasta FP se encuentra una de creación de horarios. Lo hace directamente desde la nube y permite generarlos de forma flexible cada día.

FET



Ilustración 8. FETA.

Es una aplicación de software libre que sirve para generar horarios académicos a través de un algoritmo. Es capaz de definir clases, tiempos, asignaturas, profesores, aulas y crear actividades en un máximo de 20 minutos. Una vez terminado, muestra varias opciones de horarios y genera un documento html con el elegido que se puede colgar en la web del centro para compartir con el resto de los usuarios.

Generador de horarios de ARASAAC

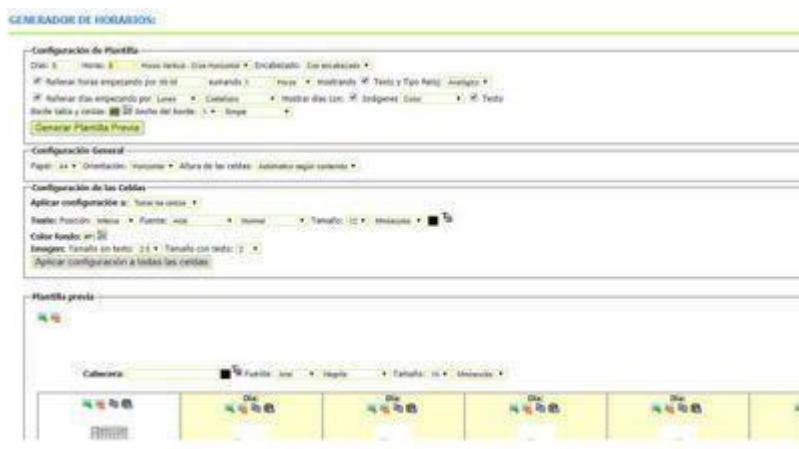


Ilustración 9. Interfaz ARASAAC.

El Portal Aragonés de la Comunicación Aumentativa y Alternativa cuenta con diversas herramientas de gestión online para crear diferentes recursos, entre ellas un generador de horarios fácil de usar. Para ello, basta con elegir el número de días que debe contener, la hora de inicio y la duración de cada clase, así como el color, el tamaño de la tipografía de las celdas o incluir pictogramas.

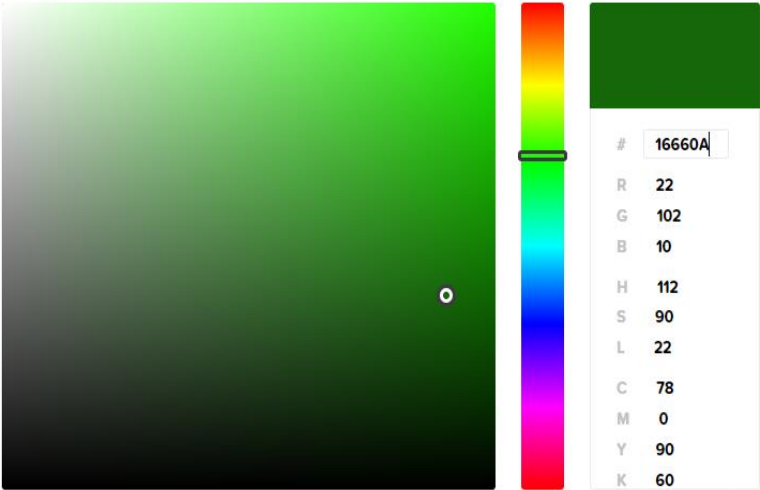
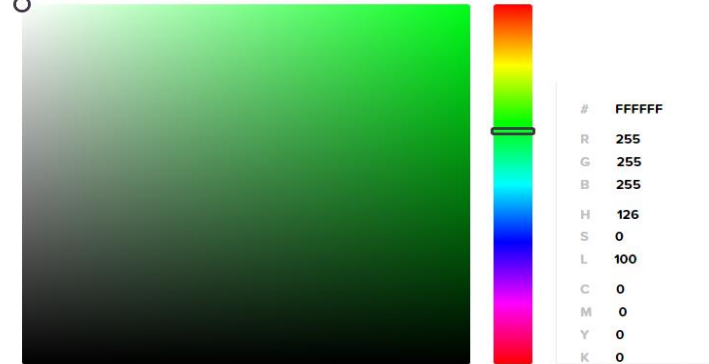
Psicología del color

La psicología del color es un campo de estudio que está dirigido a analizar cómo percibimos y nos comportamos ante distintos colores, así como las emociones que suscitan en nosotros dichos tonos. El color es capaz de estimular o deprimir, puede crear alegría o tristeza. Así mismo, determinados colores despiertan actitudes activas o por el contrario pasivas. Con colores se favorecen sensaciones térmicas de frío o de calor, y también podemos tener impresiones de orden o desorden. Se identifica al color con lo masculino y con lo femenino, con lo natural y con lo artificial, con lo romántico y con lo clásico, con la popularidad, la exclusividad y con la colectividad. El color, por tanto, no sólo es sensación, sino que básicamente es emoción. Sus atributos como significantes son apreciados no solamente por los artistas, sino también por publicistas, diseñadores, decoradores, científicos, educadores, políticos y agentes sociales y laborales, etc.

¿Cómo aplicamos la psicología del color a nuestro proyecto?

La aplicación que se está elaborando tiene como finalidad mejorar la administración y la automatización de materias y grupos de la Facultad de Ingeniería Mecánica y Eléctrica. Debido a que es una institución se usarán los colores representativos de la institución los cuales son verde y blanco, además del gris.

Paletas de colores

Color	Código
Verde	<div><div><div>HEX #16660A RGB 22, 102, 10 HSL 112, 90%, 22%</div><div><p>A color selection tool for the color Verde. It features a large square color field with a gradient from light green to dark green. A vertical color bar on the right shows the full spectrum of colors. A small circle is positioned on the color field. To the right of the color bar is a list of color codes: # 16660A, R 22, G 102, B 10, H 112, S 90, L 22, C 78, M 0, Y 90, K 60.</p></div></div></div>
Blanco	<div><div><div>HEX #FFFFFF RGB 255, 255, 255 HSL 126, 0%, 100%</div><div><p>A color selection tool for the color Blanco. It features a large square color field with a gradient from light green to dark green. A vertical color bar on the right shows the full spectrum of colors. A small circle is positioned on the color field. To the right of the color bar is a list of color codes: # FFFFFFF, R 255, G 255, B 255, H 126, S 0, L 100, C 0, M 0, Y 0, K 0.</p></div></div></div>

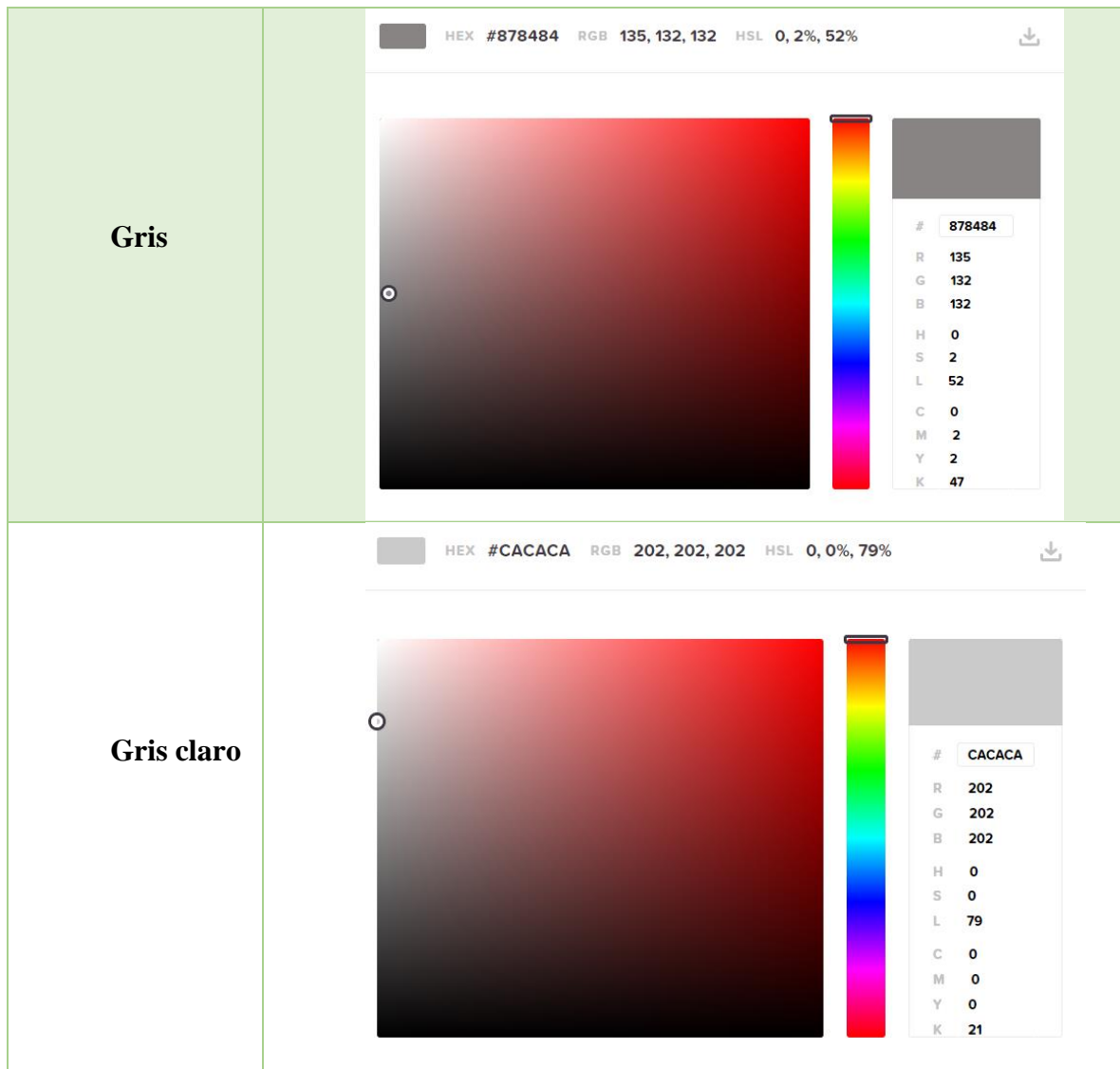


Ilustración 10. Paleta de colores.

Ya que el color verde es la base del color de la interfaz se usarán ciertos efectos visuales en color que podrían cambiar un poco la tonalidad además de incorporar otros verdes que ayuden al color base.

Diseño de prototipos

Se usará una imagen de la institución de fondo debido a que se quiere capturar la belleza del entorno en el que se encuentra la institución. Así mismo, el LOGIN se hace transparente para no perder tanto detalle del fondo. Color blanco y verde usados en la institución, y pequeños usos del negro para palabras.

Se agregó el logotipo de la Universidad Autónoma de Nuevo León a la izquierda del LOGIN y diseño que representa a la Facultad de Ingeniería Mecánica y Eléctrica que es el mismo oso tan representativo de la facultad a la derecha del LOGIN.

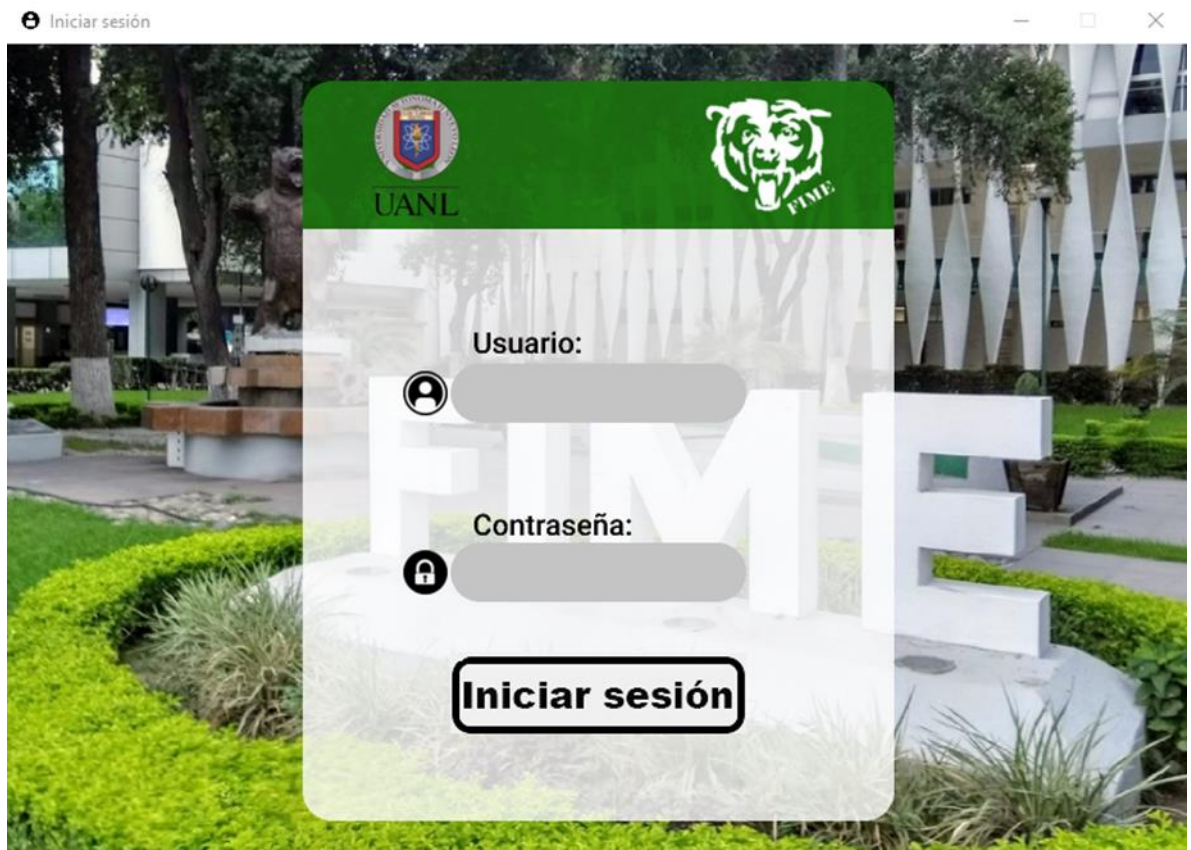


Ilustración 11. Prototipo pantalla Inicio de sesión.

Para la pantalla MENÚ aún existen muchos cambios por hacer, pero se inicia con un Label en la parte superior, con figuras de los colores de la escuela a sus esquinas. Junto con los respectivos diseños y logos de las instituciones para el que se elabora la aplicación.

En la barra MENÚ también se encuentran los botones de navegación. Posteriormente debajo de la barra menú se encuentran ComboBox para la selección de Carrera, Semestre y Turno para buscar de forma más sencilla dentro de los grupos organizados que se encuentran en el Treeview justo debajo. Como se observa, contiene los apartados Agrupación, Plan, Carrera y Turno, esto puede llegar a cambiar en el futuro si se llega a encontrar una forma más optimizada de ordenar los grupos.

A la derecha se encuentra otro Treeview que pertenece a los grupos de materias que conforman la Agrupación a la izquierda. Al seleccionar un elemento del Treeview de la Agrupación se podrá ver las materias que constituyen dicha Agrupación.



Ilustración 12. Prototipo pantalla Visualización.

La pantalla Agregar contiene los mismos elementos de la barra de la pantalla Menú. Debajo de la barra se encuentra el CRUD de los grupos usando el mismo color verde, pero con efectos de transparencia y al fondo el logo de la Facultad de Ingeniería Mecánica y Eléctrica.

La pantalla Agregar incorpora materias a la base de datos con los datos Plan, Materia, Carrera, Semestre, Empleado, Hora y Capacidad. Todos los grupos de

materias en la base de datos se encuentran en el Treeview a la derecha, en el futuro también se agregará ComboBox para la búsqueda.

The screenshot shows the ARASAAC interface. At the top is a green header bar with the UANL logo, a 'Menú' button, and a 'Crear grupos' button. Below the header, on the left, is a green box titled 'Agregar grupo' containing input fields for 'Plan:', 'Clave:', 'Materia:', 'Carrera:', 'Semestre:', 'Empleado:', and 'Hora:', along with an 'Aceptar' button. On the right, a window titled 'Materias del grupo' displays a table of materials for a specific group.

ID Grupo	Plan	Clave	Materia	Carrera	Semestre	Empleado	Hora
8	401	50	Investigación de operaciones	IAS	5	102947	V4
9	401	40	Álgebra lineal	IAS	5	123123	V5
10	401	30	Física	ITS	1	12312	M1
11	401	20	Química	ITS	1	12312	M2
12	401	31	Interfaces	IAS	5	12312	V1
15	401	123	Circuitos	IME	5	123123	N4
16	401	23	Álgebra lineal	IAS	5	123	V4
18	401	12321	Fuerzas	IME	6	123	M3

Ilustración 13. Interfaz ARASAAC.

Especificación de software

El software será creado a partir del lenguaje Python 3.9, el cual cuenta con una gama alta de bibliotecas. Además, el programa está hecho para sistemas operativos Windows en cualquiera de sus versiones.

Requisitos de hardware.

Especificaciones generales

Para su correcto uso el sistema debe ser instalado en una computadora, claramente contar con teclado y ratón. Aunque la mayoría de las computadoras en la actualidad son más que suficientes para el uso de este programa, hay organizaciones que pueden llegar a utilizar computadoras viejas que tenían con anterioridad, el programa igualmente funcionara, pero recordar que si quiere el uso óptimo para la administración debemos usar dispositivos correspondientes. El usuario solo necesita conocimientos básicos del uso de una computadora.

Especificaciones técnicas detalladas

El computador puede usar cualquiera de las versiones de Windows, aunque siempre se aconseja tener la versión más reciente. A partir de ahí, el sistema es muy práctico y estable por lo que no necesita de más requerimientos.

Diseño de la base de datos

El primer paso en el diseño de la base de datos es analizar los datos que se recolectarán y determinar el uso que se piensa hacer de los mismos. Para mayor consistencia, cada grupo de datos debe definirse como un grupo de tablas de datos relacionadas. En un grupo pueden incluirse los datos de una o más encuestas. El análisis de las hojas de datos y de los métodos de recolección de datos identifica varios grupos de datos diferentes.

Normalización de la base de datos

Hay un grupo de tablas de base que se usan a lo largo de toda la base de datos. Estas tablas son comunes a algunos o a casi todos los grupos listados. Por lo común se usa, valores de tabla como nombre de la columna, tipo, nombre descriptivo, rangos válidos o valores, columna del índice y descripción. Constituye una gran parte del documento dependiendo la cantidad de valores.

En un principio analizando el objetivo del proyecto se crearon dos tablas las cuales consideramos que tendría toda la información requerida para el correcto funcionamiento de la aplicación de escritorio. Se crearon tres tablas principales las cuales tienen por nombre Tabla Usuario y Tabla Materia.

Primera Forma Normal (1FN)

Tablas con atributos atómicos

ID_usuario	Nombre	Contraseña

Tabla 2. Tabla de Usuarios.

Tablas sin redundancia. En este caso se buscaron los datos que causaban redundancia y se aplicó un cambio en la tabla para que se tuviera una conexión gracias a las claves foráneas.

ID_clase	Plan de estudio	Agrupación	Hora	Día	Empleado (Maestro)	Id Paquete	Clave (id_materia)

Tabla 3. Tabla de Clases.

Para cumplir con la regla de que no se puede perder información al momento de generar la normalización se creó una nueva tabla en donde por medio de una clave primaria estará conectada con la tabla anterior.

ID_materia	Clave	Nombre_materia

Tabla 4. Tabla Materia.

Segunda forma Normal (2NF). Todos sus atributos que no son de la clave principal tienen dependencia funcional completa respecto de todas las claves existentes en el esquema, pero en lenguaje pueden emplearse de forma más amplia. La base de datos en la programación es bastante versátil por lo que se pueden modificar valores teniendo en cuenta cualquier valor en la tabla, por ello se almacenan la mayoría de los datos en una tabla correspondiente al ámbito. La 2FN se aplica a las relaciones que tienen claves primarias compuestas por dos o más atributos. Si una relación está en 1FN y su clave primaria es simple (tiene un solo atributo), entonces también está en 2FN.

ID_clase	Plan de estudio	Agrupación	Hora	Día	Salón	Empleado (Maestro)	Id Paquete	Clave (id_materia)
01	401	002	M1-M3	Martes	2-412	1542	470	01

Tabla 5. Ejemplo de la tabla clase llena.

ID_materia	Clave	Semestre	Nombre_materia
01	002	1	Matemáticas 1

Tabla 6. Ejemplo de tabla materia llena.

ID_usuario	Nombre	Contraseña
01	Juan José	*****

Tabla 7. Ejemplo de tabla usuario llena.

Agrupación	Carrera	Turno	Semestre
470	IME	Matutino	1

Tabla 8. Ejemplo de tabla agrupación llena.

Por el objetivo de nuestro trabajo que será el generar paquetes de materias todos los atributos que se encuentran dentro de la tabla están ligados directamente con la clave primaria por lo cual las tablas quedarían de la misma forma para que el funcionamiento del programa sea el adecuado para la tarea requerida.

Tercera Forma Normal (3FN). Se considera que una Tabla está en la tercera forma normal cuando se cumplió con la segunda forma y además de eso no existen dependencias transitivas entre atributos que no son clave.

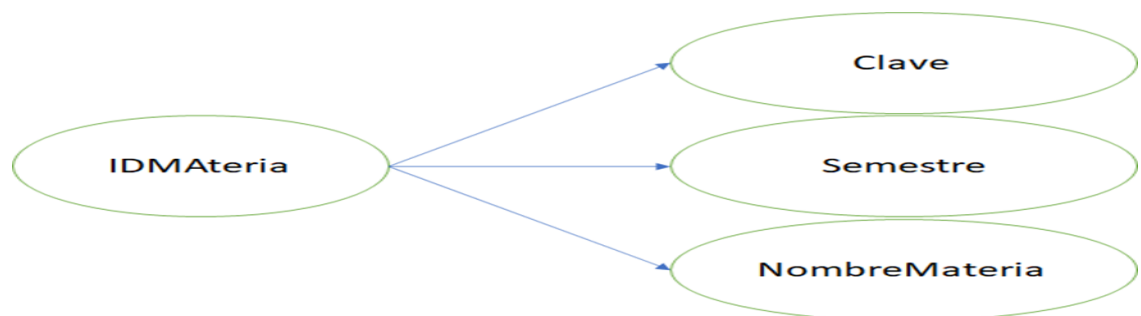


Ilustración 14. Relación entre los campos de la tabla materia.

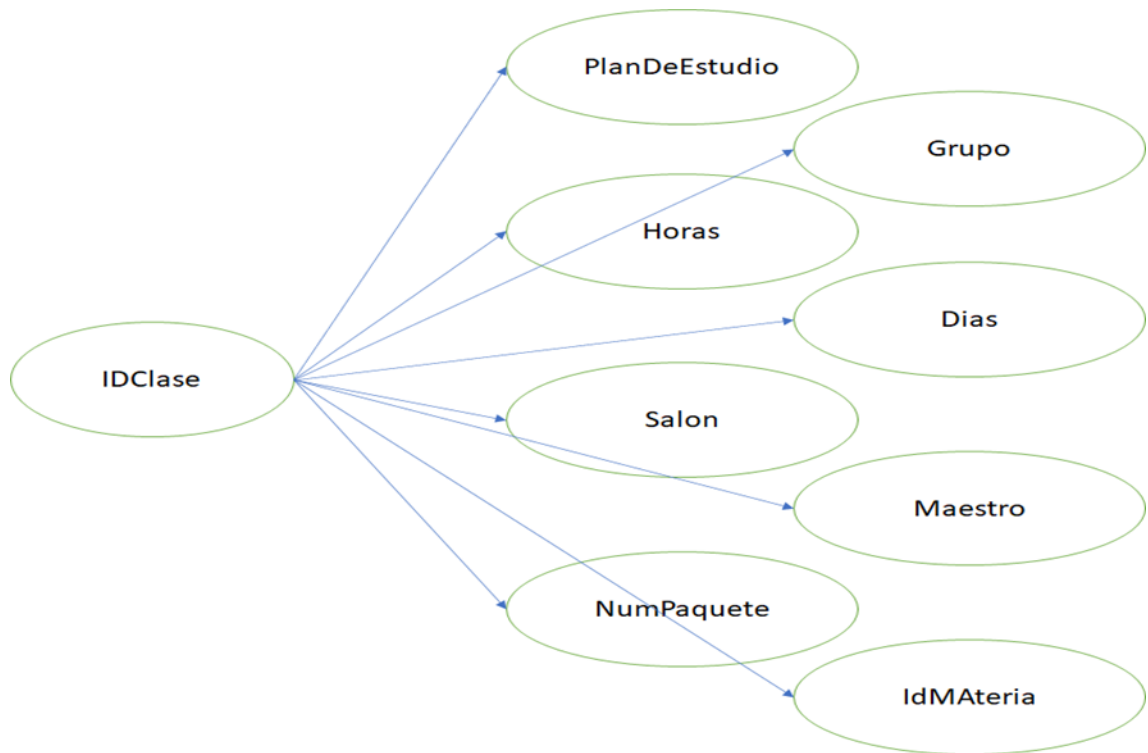


Ilustración 15. Relación entre los campos de la tabla clase.

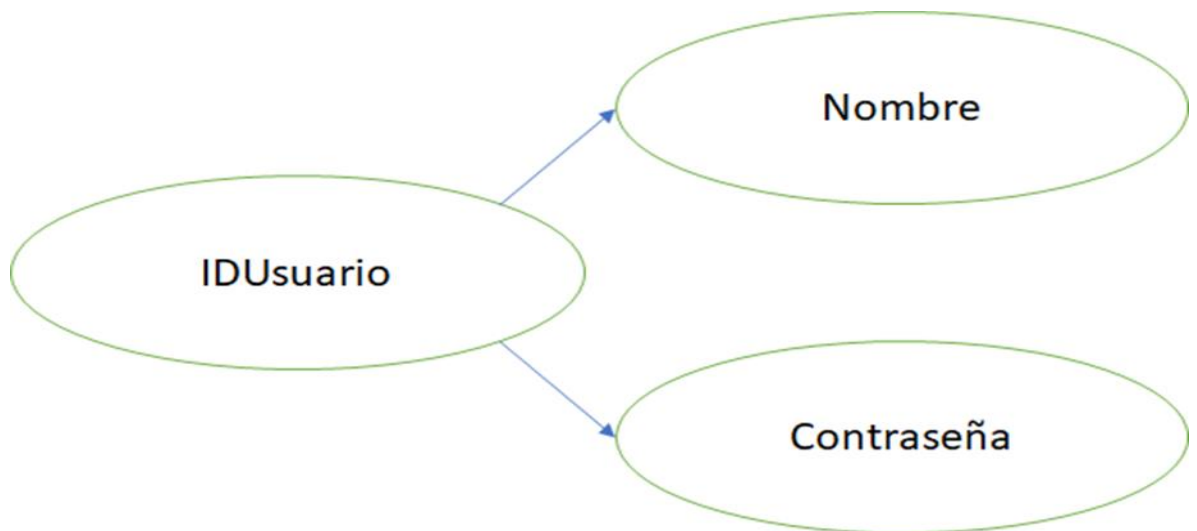


Ilustración 16. Relación entre los campos de la tabla usuario.

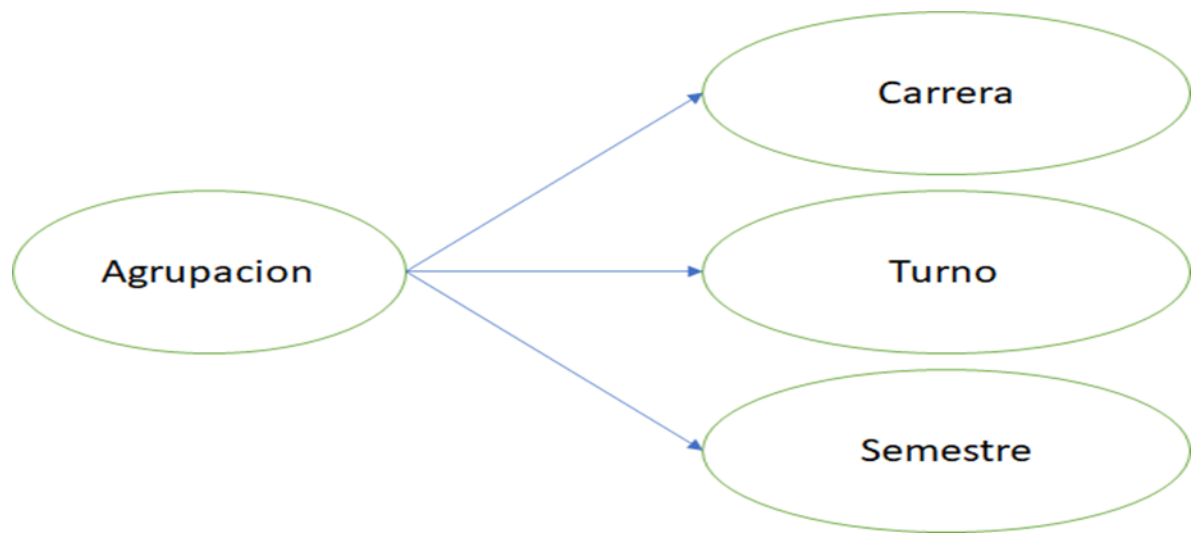


Ilustración 17. Relación de los campos en la tabla agrupación.

Modelo conceptual del sistema

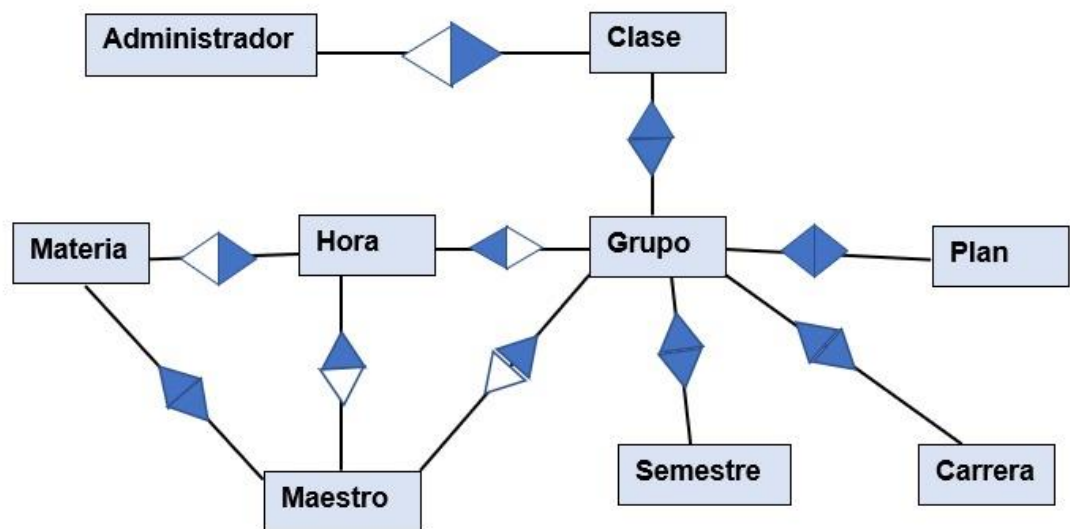


Ilustración 18. Modelo conceptual.

Diagrama UML



Ilustración 19. Diagrama UML.

Diccionario de datos

Materia:

Nombre	Tipo de dato	Schema
ID_Materia	auto numeración	ID_materia "ID"
Clave	string	Clave "string"
Semestre	string	Semestre "string"
Nombre de materia	string	Nombre Materia "string"

Tabla 9. Diccionario de datos tabla materia.

Clase:

Nombre	Tipo de dato	Schema
ID Clase	auto numeración	ID_clase "ID"
Plan de estudios	string	Plan_de_estudios "string"
Grupo	string	Grupo "string"
Hora	string	Hora "string"
Día	string	Día "string"
Salón	string	Salón "string"
Numero de agrupación	string	Agrupación "string"
Empleado (Maestro)	string	Empleado "string"
Id materia	text	ID_materia "text"

Tabla 10. Diccionario de datos tabla clase.

Agrupación:

Nombre	Tipo de dato	Schema
ID Agrupación	auto numeración	ID_agrupación "ID"
Carrera	string	Carrera "string"
Turno	string	Turno "string"
Semestre	string	Semestre "string"

Tabla 11. Diccionario de datos tabla agrupación.

Usuario:

Nombre	Tipo de dato	Schema
ID Usuario	auto numeración	ID_usuario "ID"
Nombre	string	Nombre "string"
Contraseña	string	Contraseña "string"

Tabla 12. Diccionario de datos tabla usuario.

Interfaces gráficas (entrada, salida, combinadas)

Interfaz: Inicio de sesión

La siguiente pantalla muestra en el inicio de sesión, la cual consta de usuario y contraseña. Se maneja un usuario y contraseña establecidos en el código fuente de la aplicación. El diseño cuenta con una imagen de la institución en el fondo, los colores representativos de la misma y los logos de las instituciones pertenecientes.

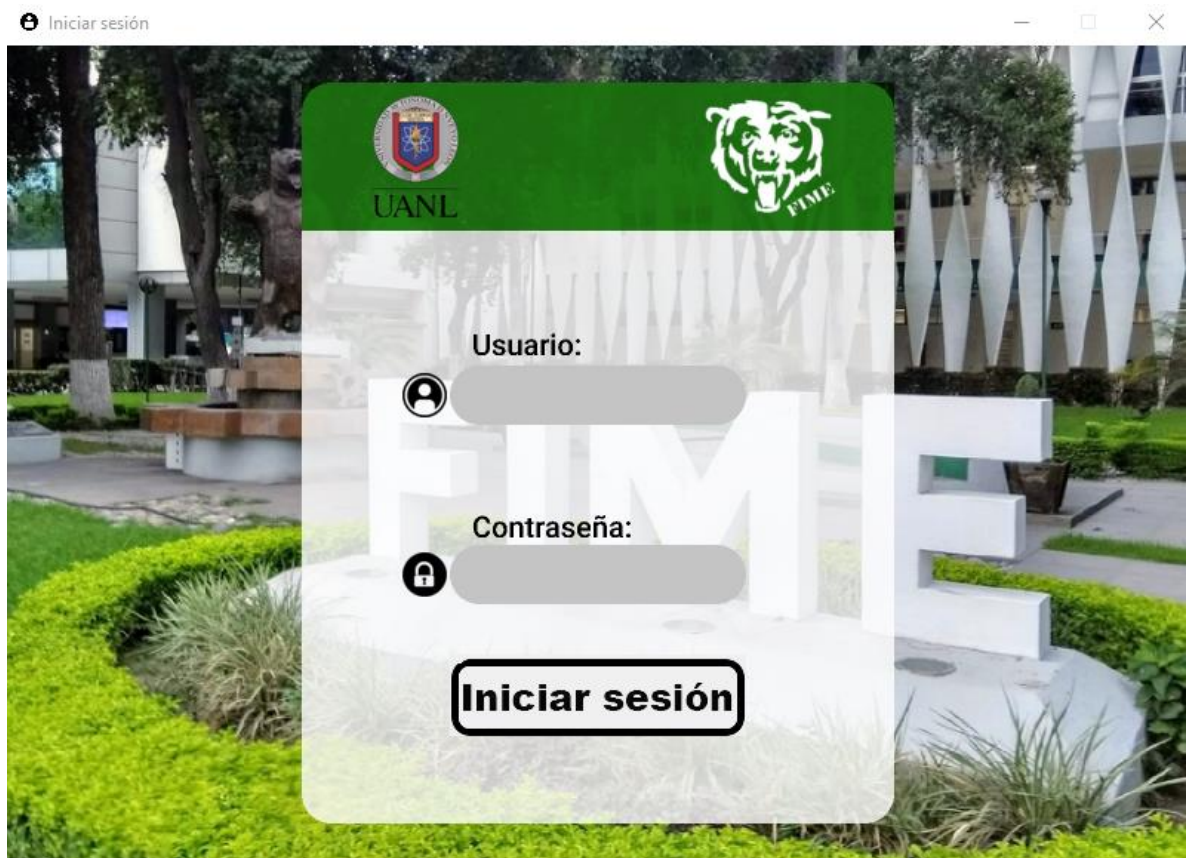


Ilustración 20. Pantalla de inicio de sesión.

Interfaz: Visualización

La siguiente pantalla abarca la visualización de los datos relacionados a las agrupaciones de las materias hechas. Como se observa, principalmente cuenta con dos tablas Treeview relacionadas entre sí.

La tabla “Grupos organizados” contiene todas las agrupaciones hechas en el proceso de la creación de Agrupaciones, al seleccionarlo se mostrará en la tabla “Materias del grupo” todas las materias que conforman dicha Agrupación con sus datos correspondientes.

Justo arriba de la tabla “Grupos organizados” se encuentran tres ListBox tipo ComboBox relacionados conjuntamente entre sí para filtrar las Agrupaciones según su carrera, semestre y turno, y a un lado a la izquierda el botón “Mostrar todo” para reiniciar los ListBox y mostrar todas las Agrupaciones.

En la parte de arriba a la derecha se encuentra el botón “Editar grupos” para ir a la ventana del CRUD de las materias y llenar la base de datos.

El diseño se concentra en la barra superior de la aplicación con los colores representativos, figuras que proporcionan diseño y los logos correspondientes a la institución. De fondo se encuentra el escudo de la Facultad de Ingeniería Mecánica y Eléctrica.



Ilustración 21. Pantalla interfaz de Visualización de datos.

Interfaz: CRUD

La interfaz CRUD como su propio nombre lo indica contiene las funciones para crear, leer, actualizar y eliminar registros de la base de datos. A la izquierda está el formulario “Agregar grupo” que cuenta con los apartados Plan, Clave, Materia, Empleado y Salón como Entry’s de forma escrita y los apartados Carrera, Hora, Semestre y Día como ComboBox de forma seleccionable.

La clave es individual de cada materia, ósea no es repetible, en caso contrario el sistema notificará para que sea cambiada, también existe la comprobación para que ningún empleado pueda tener dos grupos con la misma hora.

A la derecha se encuentra la tabla Treeview “Materias registradas” para verificar los registros en la base de datos.

Para eliminar un registro se requiere seleccionar un registro de la tabla Treeview y a continuación oprimir el botón “Eliminar registro”, si no se selecciona ningún registro el propio sistema lo notificará.

Agregar registro

UANL

Menú Actualizar grupos

Agregar grupo

Plan:

Clave:

Materia:

Carrera: IME Hora: M1

Semestre: 1 Día: L-M-V

Empleado:

Salón:

Aceptar

Cantidad de registros: 15

Materias registradas

ID Grupo	Plan	Clave	Materia	Carrera	Semestre	Empleado	Hora	Día	Salón
1	401	232	Química	IME	1	213	M1	L-M-V	2-323
2	401	123	Física	IME	1	123	M2	L-M-V	321
3	401	33	Matematicas	IME	1	123	M3	L-M-V	2-123
4	401	231	Algebra	IME	1	1233	M4	L-M-V	1-2123
5	401	2334	Física	IAS	3	2123	M1	L-M-V	23333
7	401	4445	Química	IME	1	245	V1	L-M-V	213
8	401	45	Física 3	IMT	3	445	M3	M-J	2-321
9	401	455	Física 4	ITS	7	252	N1	M-J	3-134
10	401	3245	Dibujo	IME	3	451	M3	M-J	24-123
11	401	355	Dibujo2	IME	1	42123	M3	M-J	2-123
12	401	5213	NO	IME	3	43129	M1	M-J	4123

Editar registro Eliminar registro

Ilustración 22. CRUD modo Create.

Para actualizar un registro, al igual que al eliminar, se requiere seleccionar uno de la tabla Treeview y presionar el botón “Editar registro”. A continuación, se abrirá una ventana nueva tipo TopLevel con la información correspondiente del registro. Se realizan los cambios requeridos y se presiona el botón “Actualizar” para completar la edición del registro.



Ilustración 23. CRUD modo Update.

Herramientas CASE

Las herramientas CASE son los programas o aplicaciones informáticas utilizadas para aumentar la productividad del desarrollo de software y a su vez disminuyen los costos de tiempo y dinero, para la realización de esta aplicación se utilizaron tres herramientas case: Python para la codificación, Access Database para la base de datos y Proxlight Designer para la creación de las interfaces.

No	Tipo de herramienta case	Nombre	Versión	Descripción
1	Alto nivel	Python	3.9.0	Lenguaje de programación utilizado para la codificación del proyecto.
2	Alto nivel	AccessDatabase	16.0	Base de datos utilizada para almacenar la información
3	Alto nivel	Proxlight Designer	3.9	Utilizado para la creación de las interfaces de usuario en Python

Tabla 13. Herramientas CASE.

Casos de uso

Primer caso	
Objetivo:	Validar usuario y contraseña para log in
Nombre:	Log in
Descripción:	Validar el acceso a plataforma por parte del usuario al ingresar usuario y contraseña
Pasos:	
Ingresar usuario	Ingresar valores en campo usuario
Ingresar contraseña	Ingresar valores en campo contraseña
Log In	Clic en botón confirmar datos
Log in fallido	El usuario o contraseña son incorrectas por lo cual es necesario volver a intentar
Resultados	Acceso a plataforma

Tabla 14. Tabla del primer caso de uso.

Segundo caso	
Objetivo:	Agregar una materia
Nombre:	Asignación grupo
Descripción:	Se agrega a un grupo tomando en consideración diferentes criterios
Pasos:	
Ingresar información	Llenar campos en CRUD para crear el nuevo grupo
Confirmar información	Validar los campos ingresados.
Registrar	Clic en botón confirmar datos
Resultados	Se ha registrado la materia

Tabla 15. Tabla del segundo caso de uso.

Tercer caso	
Objetivo:	Actualizar registro ya en plataforma desde tabla treeview
Nombre:	Actualizar registro
Descripción:	Modifica y actualiza registro
Pasos:	
Seleccionar registro	Colocar indicador sobre registro en tabla treeview
Confirmar	Confirmar operación mediante botón “Editar registro”
Validar información	Muestra en pantalla ventana tipo topleven con la información correspondiente al campo seleccionado
Ingresar modificaciones	Posterior a validar los campos existentes ingresar los cambios al registro
Confirmar cambios	Aceptar cambios en registros en botón “Actualizar”
Resultados	Realizar modificaciones a registro ya existente.

Tabla 16. Tabla del tercer caso de uso.

Cuarto caso	
Objetivo:	Eliminar registro de grupo en tabla treeview
Nombre:	Eliminar registro
Descripción:	Se busca eliminar un registro no deseado o cuando cometes un error
Pasos:	
Seleccionar registro	Colocar indicador sobre registro en tabla treeview
Confirmar	Confirmar operación mediante botón “Eliminar registro”

Eliminar fallido	Ocurre cuando se selecciona botón eliminar registro y no esta seleccionado un registro.
Resultados	Elimina de plataforma registro seleccionado para no mostrarlo más.

Tabla 17. Tabla del cuarto caso e uso.

Quinto caso	
Objetivo:	El usuario Filtrar / visualiza grupos
Nombre:	Filtro
Descripción:	Filtrar las agrupaciones ya sea por carrera, semestre o turno
Pasos:	
Seleccionar listbox	Situarse en listas para ingresar filtro: carrera, turno, semestre
Selecciona botón Mostrar todo	Muestra todos los registros de agrupaciones.
Resultado	Muestra en pantalla filtro con información de acuerdo con lo solicitado por el usuario

Tabla 18. Tabla del quinto caso de uso.

Sexto caso	
Objetivo:	Cierre de sesión usuario
Nombre:	Logout
Descripción:	Validar cierre y salida de plataforma por parte del usuario
Pasos:	
Seleccionar botón para cerrar	Manda un mensaje de alerta para confirmar el exit.
Resultado	Cierra la aplicación

Tabla 19. Sexto caso de uso.

Conclusiones del proyecto

Conclusiones:

Finalmente se realizó una aplicación factible, con usabilidad, acorde a los requerimientos solicitados, para así satisfacer las necesidades de la Coordinación de Administración y Sistemas de la Facultad de Ingeniería Mecánica y Eléctrica. La aplicación esta lista para ser empleada y cumple con la función para que el usuario elija horarios que sean adecuados para los semestres próximos.

Lo que se logró conseguir con el uso de la aplicación:

1. Reducción en tiempos. Se invierte menos tiempo en la gestión de buscar o crear paquetes de materias.
2. Seguridad en la información. Con la implementación de claves y contraseñas se tiene solamente accesos autorizados.
3. Consulta de datos. Se puede visualizar las agrupaciones hechas por el proceso de creación de paquetes de materias.
4. Diseño alusivo a la institución. La interfaz cuenta con los colores de la institución y escudo de la Facultad de ingeniería Mecánica y Eléctrica
5. Funciones para crear, actualizar y eliminar registros de la base de datos. Esto ayudara a las actualizaciones que el usuario necesite realizar en los paquetes de materias.

Pronóstico del tiempo

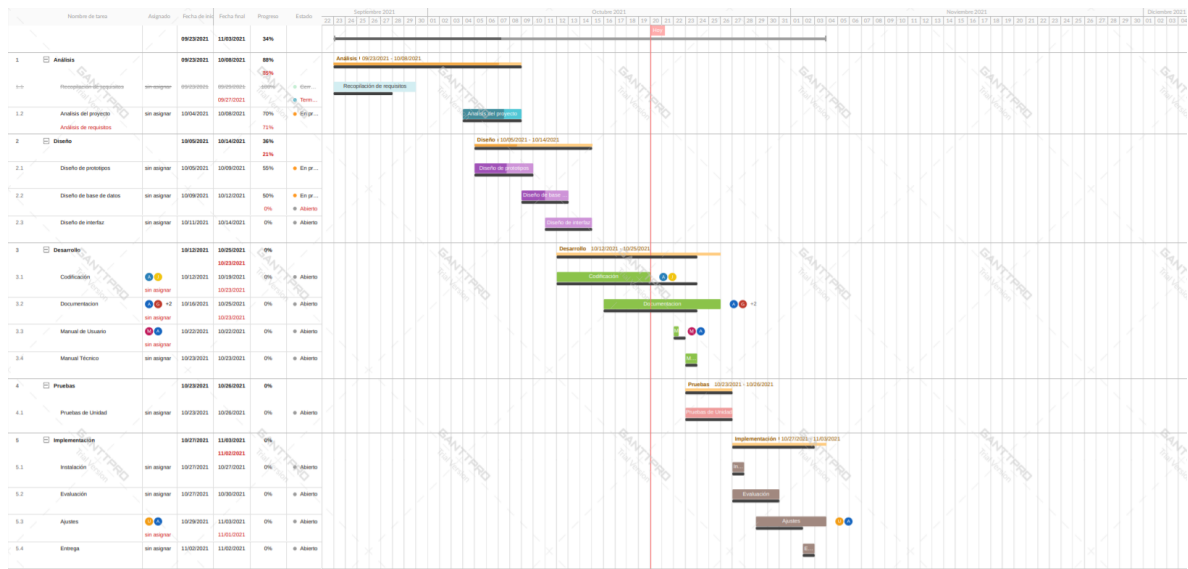


Ilustración 24. Diagrama de Gantt previsto

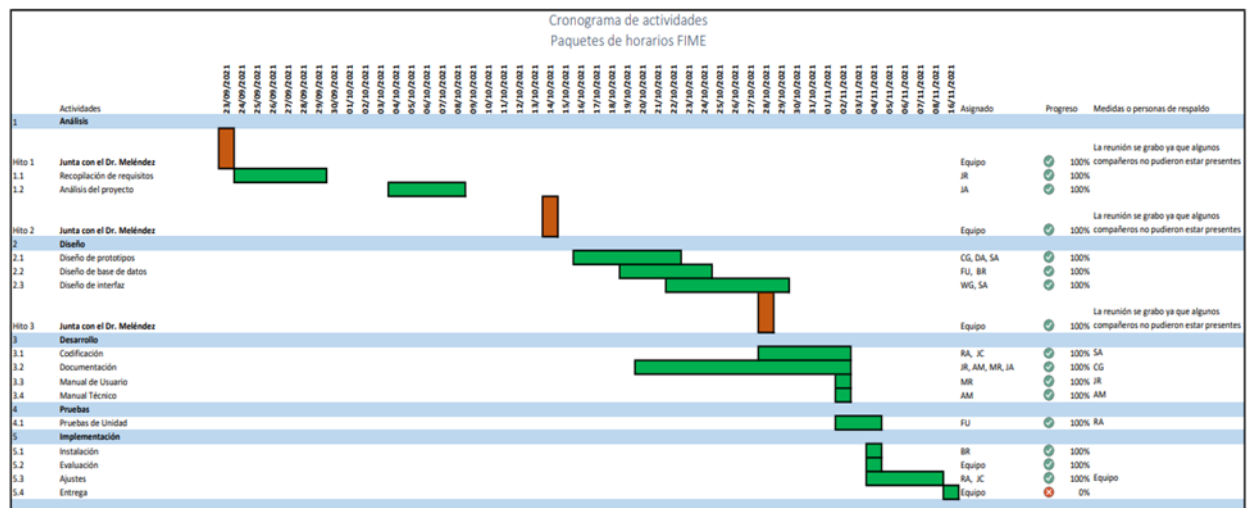


Ilustración 25. Diagrama de Gantt en tiempo real

Recursos		
Iniciales	Nombre	Rol
JR	Jenifer Alejandra Rodríguez Méndez	Análisis
AM	Alondra Monsserrat Ramírez Proa	Análisis
CG	Cynthia Guadalupe Vega Villalba	Diseño
SA	Selene Aydee Abrego Bonilla	Diseño
MR	Maricarmen Regalado Espinoza	Análisis
FU	Fernando Uriel Zapata Rivas	Codificación
RA	Roberto Alan Rodríguez Monroy	Codificación
JA	José Alberto Vázquez Martínez	Análisis
JC	Juan Carlos Romero Ortiz	Codificación
BR	Braulio Rafael Torrez Ruiz	Codificación
DA	Diego Alan Sánchez Partida	Diseño
WG	William Gilberto Vargas Delgado	Diseño
	Equipo	

Ilustración 26. Abreviación de nombres asignados al cronograma.

Anexos

Manual técnico

Propósito

El propósito del proyecto es la creación de una aplicación para ordenar en agrupaciones las materias que llevará un alumno en un cierto semestre, carrera y turno en la Facultad de Ingeniería Mecánica y Eléctrica.

Alcance

El uso de la aplicación está dado solo para un usuario el cual es el coordinador relacionado al área que proporciona las materias a los alumnos.

Aunque la aplicación es un sencillo programa para gestión de una base de datos, da oportunidad a crecer de acuerdo con las necesidades del usuario también debido al lenguaje en el que está construido que es fácil de aprender.

Programas utilizados

El lenguaje en el que fue desarrollada la aplicación es Python 3.9, como editor se utilizó SublimeText pero eso es opcional para el equipo de trabajo. Además, se

utilizó el editor de diseño de prototipos FIGMA con la ayuda de Proxlight Designer para desarrollar los diseños del sistema.

Construcción del proyecto

Interfaz inicio de sesión:



Ilustración 27. Pantalla de inicio de sesión.

La parte de la interfaz de inicio de sesión está constituida por la función ingreso() la cual nos permite comprobar el usuario y contraseña con los establecidos dentro de la aplicación comparándolos.

```
def ingreso(self):
    NombreUsuarioAdmin = "admin"
    PasswordAdmin = "ROOT1747264"
    self.db = Conexion()
    try:
        self.db.conectar()
    except:
        print("Error al conectar")

    self.recogerinformacion=self.db.cursor.execute(f"SELECT * FROM Usuarios").fetchall()
    count1=0
    for row in self.recogerinformacion:
        count1+=1
    if count1 == 0:
        pass
    else:
        if self.nombreIn.get() != "" and self.passwordIn.get() != "":
            for i in self.recogerinformacion:
                valor=i
                usuario=valor[1]
                contraseña=valor[2]
                if usuario == self.nombreIn.get() and contraseña == self.passwordIn.get():
                    contador_de_salida=1
                    break
                elif NombreUsuarioAdmin == self.nombreIn.get() and PasswordAdmin == self.passwordIn.get():
                    contador_de_salida=1
                    break
                else:
                    contador_de_salida=0
            if contador_de_salida==1:
                messagebox.showinfo("Bienvenida", f"Bienvenido {self.nombreIn.get()}")
                self.window.destroy()
                #window= Tk()
                window = ThemedTk(theme="adapta")
                entrar_menu=Menu_organizacion(window,"Menú",1350,670)
                window.mainloop()
            else:
                messagebox.showinfo("Error", "Usuario o contraseña incorrectos.")
        else:
            messagebox.showinfo("Error", f"Se deben llenar todos los campos")
```

Ilustración 28. Código de función: ingreso()

Interfaz Visualización:

The interface features a green-themed menu bar with options: 'Base de datos', 'Editar', and 'Ayuda'. Below the menu, there are filters for 'Carrera: IME', 'Semestre: 1', and 'Turno: Matutino'. The main content area is divided into two panels:

- Grupos organizados:** A table with columns 'Agrupación', 'Carrera', 'Turno', and 'Semestre'. It displays one row: 470, IME, Matutino, 1.
- Materias del grupo:** A table with columns 'Agrupación', 'Plan', 'Clave', 'Materia', 'Carrera', 'Semestre', 'Empleado', 'Hora', 'Día', and 'Salón'. It displays five rows of course data.

Agrupación	Plan	Clave	Materia	Carrera	Semestre	Empleado	Hora	Día	Salón
470	401	232	Quimica	IME	1	213	M1	L-M-V	2-323
470	401	123	Fisica	IME	1	123	M2	L-M-V	321
470	401	33	Matematicas	IME	1	123	M3	L-M-V	2-123
470	401	231	Algebra	IME	1	1233	M4	L-M-V	1-2123
470	401	355	Dibujo2	IME	1	42123	M3	M-J	2-123

Ilustración 29. Pantalla interfaz de Visualización de datos.

La interfaz visualización como su propio nombre indica es la visualización de los datos.

Funciones:

- Mostrar materias de la agrupación seleccionada (def click_treeview)
- Mostrar agrupaciones seleccionadas en la ListBox (def selección_del_combobox)
- Mostrar todas las agrupaciones (def mostrarTodo)
- Ir al CRUD (def ir_agregar)

Función click_treeview

Esta función nos permite mostrar las materias que se encuentran en la agrupación seleccionada dentro de la tabla “Grupos organizados”. Toma la variable “values” para registrar lo seleccionado dentro de la tabla, values[0] corresponde al ID del registro, a continuación recoge los datos de las Agrupaciones y compara esta ID con aquellos registros que contenga el mismo ID de registro de agrupación. Para finalizar incorpora los datos de dichos registros en la tabla “Materias del grupo”.

```
def click_treeview(self,e):
    seleccion = self.tabla_grupos.focus()
    values = self.tabla_grupos.item(seleccion,"values")

    self.tabla.delete(*self.tabla.get_children())

    self.recogerinformacion=self.db.cursor.execute(f"SELECT * FROM Grupos_ordenados").fetchall()
    if len(self.recogerinformacion) != 0:
        for i in self.recogerinformacion:
            if values[0] == i[1]:
                value1=i[11]
                value2=i[2]
                value3=i[3]
                value4=i[4]
                value5=i[5]
                value6=i[6]
                value7=i[7]
                value8=i[8]
                value10=i[10]
                value11=i[11]
            self.tabla.insert(parent="",index="end", text="", values=(value1,value2,value3,value4,value5,value6,value7,value8,value10,value11))
```

Ilustración 30. Código de la función: click_treeview

Función seleccion_del_combobox

Esta función es utilizada para filtrar las agrupaciones por carrera, semestre y turno. Los combobox están enlazados con condicionales para cualquier situación que se dé y se requiera encontrar la información seleccionada en ellos. En cada caso se reinicia la tabla con los nuevos valores comparados. Los posibles valores son prácticamente si se ha seleccionado o no el combobox.

```
def seleccion_del_combobox(self, event):
    self.recogerinformacion_grupos=self.db.cursor.execute(f"SELECT * FROM Agrupacion").fetchall()
    if len(self.recogerinformacion_grupos) != 0:
        self.tabla_grupos.delete(*self.tabla_grupos.get_children())
        for i in self.recogerinformacion_grupos:
            value0=i[0];value2=i[2];value3=i[3];value5=i[5]

            # si estan seleccionados los 3 combobox
            if value2 == self.comb_carreras.get() and value5 == self.comb_semestres.get() and value3 == self.comb_turnos.get():
                self.tabla_grupos.insert(parent="",index="end", text="", values=(value0,value2,value3,value5))

            # si estan seleccionados solo los 2 ultimos
            elif self.comb_carreras.get() == "" and value5 == self.comb_semestres.get() and value3 == self.comb_turnos.get():
                self.tabla_grupos.insert(parent="",index="end", text="", values=(value0,value2,value3,value5))

            # si estan seleccionados solo los 2 primeros
            elif self.comb_carreras.get() == value2 and self.comb_semestres.get() == value5 and self.comb_turnos.get() == "":
                self.tabla_grupos.insert(parent="",index="end", text="", values=(value0,value2,value3,value5))

            # si estan seleccionados el primero
            elif self.comb_carreras.get() == value2 and self.comb_semestres.get() == "" and self.comb_turnos.get() == "":
                self.tabla_grupos.insert(parent="",index="end", text="", values=(value0,value2,value3,value5))

            # si estan seleccionados solo el ultimo
            elif self.comb_carreras.get() == "" and self.comb_semestres.get() == "" and value3 == self.comb_turnos.get():
                self.tabla_grupos.insert(parent="",index="end", text="", values=(value0,value2,value3,value5))

            # si estan seleccionados el segundo
            elif self.comb_carreras.get() == "" and self.comb_semestres.get() == value5 and self.comb_turnos.get() == "":
                self.tabla_grupos.insert(parent="",index="end", text="", values=(value0,value2,value3,value5))
```

Ilustración 31. Código de la función seleccion_del_combobox

Función mostrarTodo

Esta función sencillamente muestra todas las agrupaciones en existencia tomándolas de la base de datos y reinicia los combobox.

```
# _____ Seleccion de los Combobox _____ #
def mostrarTodo(self):
    self.recogerinformacion_grupos=self.db.cursor.execute(f"SELECT * FROM Agrupacion").fetchall()
    if len(self.recogerinformacion_grupos) != 0:
        self.tabla_grupos.delete(*self.tabla_grupos.get_children())
        for i in self.recogerinformacion_grupos:
            value0=i[0]
            value2=i[2]
            value3=i[3]
            value5=i[5]
            self.tabla_grupos.insert(parent="",index="end", text="", values=(value0,value2,value3,value5))

    self.comb_carreras.current(0)
    self.comb_semestres.current(0)
    self.comb_turnos.current(0)
```

Ilustración 32. Código de la función mostrarTodo.

Función ir_agregar

Esta función cierra la ventana actual y llama a la función para ir a la ventana CRUD.

```
def ir_agregar(self):  
    self.window.destroy()  
    window=ThemedTk(theme="adapta")  
    llamada = Agregar(window, "Agregar registro", 1350, 670)  
    window.mainloop()
```

Ilustración 33. Código de función ir_agregar.

Interfaz CRUD

ID Grupo	Plan	Clave	Materia	Carrera	Semestre	Empleado	Hora	Día	Salón
1	401	232	Quimica	IME	1	213	M1	L-M-V	2-323
2	401	123	Fisica	IME	1	123	M2	L-M-V	321
3	401	33	Matematicas	IME	1	123	M3	L-M-V	2-123
4	401	231	Algebra	IME	1	1233	M4	L-M-V	1-2123
5	401	2334	Fisica	IAS	3	2123	M1	L-M-V	23333
7	401	4445	Quimica	IME	1	245	V1	L-M-V	213
8	401	45	Fisica 3	IMT	3	445	M3	M-J	2-321
9	401	455	Fisica 4	ITS	7	252	N1	M-J	3-134
10	401	3245	Dibujo	IME	3	451	M3	M-J	24-123
11	401	355	Dibujo2	IME	1	42123	M3	M-J	2-123
12	401	5213	NO	IME	3	43129	M1	M-J	4123

Ilustración 34. CRUD modo Create.

La interfaz CRUD es la relacionada con la base de datos, tanto para crear, leer, actualizar y eliminar registros.

Funciones:

- Agregar registro (agregar_elementos)
- Eliminar registro (eliminar_registro)
- Actualizar registro (actualizar)

- Crear los grupos (crear_grupos)

Función agregar_elementos

Esta función es la utilizada para agregar los elementos del formulario de la interfaz a la base de datos.

Primero se realiza la comprobación de que todos los apartados del formulario han sido llenados. A continuación, se hace la confirmación del registro con el usuario, después se llama a una función auxiliar que verifica si la clave proporcionada ya está registrada en la base de datos y otra que verifica que el empleado (maestro) ya tiene una clase a una hora asignada.

Si se cumplen las condiciones adecuadas se usa otra función auxiliar para asignar el turno de la materia y a continuación se agregan los datos a la tabla en la base de datos. Para terminar, se manda llamar la función auxiliar para actualizar la tabla de la interfaz.

```
def agregar_elementos(self):
    if self.entry0.get() != "" and self.entry1.get() != "" and self.entry2.get() != "" and self.entry3.get() != "" and self.entry4.get() != "" and self.entry5.get() != "":
        if messagebox.askokcancel(message="¿Deseas completar el registro?", title="Confirmar registro"):
            self.checar_clave()
            self.checar_empleado_hora()
            if self.ComprobacionClave==True:
                messagebox.showinfo("Error", "El valor Clave ya esta registrado en la base de datos.")
            elif self.variable_checar_empleado_hora==True:
                messagebox.showinfo("Error", "El valor Hora ya esta registrado en la base de datos con ese Empleado.")
            else:
                self.entry7 = Entry()
                self.definir_turno()
                self.db.cursor.execute(f"INSERT INTO Grupos_desordenados (Plan, Materia, Carrera, Semestre, Empleado, Hora, Clave, Turno, Dias, Salon) VALUES ('{self.entry0.get()},{self.entry1.get()},{self.entry2.get()},{self.entry3.get()},{self.entry4.get()},{self.entry5.get()},{self.entry6.get()},{self.entry7.get()},{self.entry8.get()},{self.entry9.get()}')")
                self.db.cursor.commit()
                self.entry0.delete(0, END)
                self.entry1.delete(0, END)
                self.entry2.delete(0, END)
                self.entry3.delete(0, END)
                self.entry4.delete(0, END)
                self.entry5.delete(0, END)
                self.entry6.delete(0, END)
                self.entry7.delete(0, END)
                self.entry8.delete(0, END)
                self.entry9.delete(0, END)
                messagebox.showinfo("Completado", "Actualizacion de datos completada.")
                self.actualizar_treeview()
        else:
            messagebox.showinfo("Error", "Debe llenar todos los apartados.")
```

Ilustración 35. Código de función agregar_elementos.

Función eliminar_registro

Esta función elimina el registro seleccionado en la tabla de la interfaz. Primero se pide la confirmación al usuario, a continuación, se registra el valor del registro seleccionado de la tabla en "values". Se compara values[0] con la base de datos y cuando lo encuentre será eliminado el registro y se actualizará la tabla de la interfaz.

```
def eliminar_registro(self):
    decision2=messagebox.askquestion("Confirmar","¿Seguro que quieres eliminar el registro?")
    if decision2 == "yes":
        seleccion = self.tabla.focus()
        values = self.tabla.item(seleccion,"values")

        self.infoC=self.db.cursor.execute(f"SELECT * FROM Grupos_desordenados").fetchall()
        if len(self.infoC) != 0:
            if values == "":
                messagebox.showinfo("Error","No ha seleccionado un registro")
            else:
                valor_eliminar = values[0]
                self.db.cursor.execute(f"DELETE FROM Grupos_desordenados WHERE Id = ?", valor_eliminar)
                self.db.cursor.commit()
                messagebox.showinfo("Completado","Registro eliminado.")
                self.actualizar_treeview()
```

Ilustración 36. Código de función eliminar_registro

Función actualizar

Esta función actualiza el registro seleccionado por la tabla de la interfaz. Es simple y recoge los valores guardados en la ventana extra para editar un registro. Y como todas las demás funciones, actualiza la tabla de la interfaz.

```
# FUNCION PARA ACTUALIZAR VALORES EDITADOS #
def actualizar(self):
    decision2=messagebox.askquestion("Confirmar","¿Seguro que quieres actualizar el registro?")
    if decision2 == "yes":
        self.extra_combobox_turno = Entry()
        self.definir_turno2()

        self.db.cursor.execute("UPDATE Grupos_desordenados SET Plan = ? WHERE Id = ?",self.extra_plan.get(),self.valor_extra0)
        self.db.cursor.execute("UPDATE Grupos_desordenados SET Materia = ? WHERE Id = ?",self.extra_materia.get(),self.valor_extra0)
        self.db.cursor.execute("UPDATE Grupos_desordenados SET Carrera = ? WHERE Id = ?",self.extra_combobox_carrera.get(),self.valor_extra0)
        self.db.cursor.execute("UPDATE Grupos_desordenados SET Semestre = ? WHERE Id = ?",self.extra_combobox_semestre.get(),self.valor_extra0)
        self.db.cursor.execute("UPDATE Grupos_desordenados SET Empleado = ? WHERE Id = ?",self.extra_empleado.get(),self.valor_extra0)
        self.db.cursor.execute("UPDATE Grupos_desordenados SET Hora = ? WHERE Id = ?",self.extra_combobox_horas.get(),self.valor_extra0)
        self.db.cursor.execute("UPDATE Grupos_desordenados SET Clave = ? WHERE Id = ?",self.extra_clave.get(),self.valor_extra0)
        self.db.cursor.execute("UPDATE Grupos_desordenados SET Turno = ? WHERE Id = ?",self.extra_combobox_turno.get(),self.valor_extra0)
        self.db.cursor.execute("UPDATE Grupos_desordenados SET Dias = ? WHERE Id = ?",self.extra_combobox_dias.get(),self.valor_extra0)
        self.db.cursor.execute("UPDATE Grupos_desordenados SET Salon = ? WHERE Id = ?",self.extra_salon.get(),self.valor_extra0)

        self.db.cursor.commit()
        self.top.destroy()
        messagebox.showinfo("Completado","Actualizacion de datos completada.")
        self.actualizar_treeview()
```

Ilustración 37. Código función actualizar.

Función crear_grupos

Esta función es la encargada de crear las agrupaciones. No es una función tan compleja, pero requiere mucha atención al usar los datos debido a las condiciones a las que se tienen que someter para crear una Agrupación.

Primeramente, se comprueba que haya al menos algunos valores en la base de datos, a continuación, recorre la base de datos y comprueba si el registro ya se encuentra o no en una agrupación.

```
def crear_grupos(self):
    self.nueva_tabla_access()
    self.info_crear_grupos=self.db.cursor.execute(f"SELECT * FROM Tabla_temporal").fetchall()

    if len(self.info_crear_grupos) != 0:
        for i in self.info_crear_grupos:
            comprobar_grupos_ordenados=self.db.cursor.execute(f"SELECT * FROM Grupos_ordenados").fetchall()
            comprobation=False
            for g6 in comprobar_grupos_ordenados:
                v1=g6[0]
                v2=i[0]
                if v2 == v1 or i==None:
                    comprobation=True
```

Ilustración 38. Código de función crear_grupos (parte1).

La siguiente sección separa cada dato del registro en variables y añade el registro en una tabla donde se encuentran las materias que han sido añadidas a una agrupación, además de añadir esa primera materia como referencia para la agrupación.

```
comprobation=True
if comprobation==False:
    valorA0= i[0]; valorA1= i[1]; valorA2= i[2]; valorA3= i[3]; valorA4= i[4]; valorA5= i[5]; valorA6= i[6]; valorA7= i[7]; valorA8= i[8];valorA9= i[9];v
    self.db.cursor.execute(f"INSERT INTO Agrupacion (Plan, Carrera, Turno,id_primer_materia,semestre) VALUES ('{valorA1}','{valorA4}','{valorA8}','{valor
    self.db.cursor.commit()

    self.info_agrupaciones=self.db.cursor.execute(f"SELECT * FROM Agrupacion").fetchall()
    for a in self.info_agrupaciones: #Consiguir la info de agrupaciones
        valorliadando = a[0] #ID DEL GRUPO
        VALORC4 = a[4] # ID DE LA MATERIA CON LA QUE SE DIO DE ALTA LA AGRUPACION

        if int(valorA0) == int(VALORC4):
            VALORC0 = valorliadando
            self.db.cursor.execute(f"INSERT INTO Grupos_ordenados (Id,Grupo_asignado,Plan, Clave,Materia, Carrera, Semestre, Empleado, Hora,Turno,Dias,Sa
            self.db.cursor.execute(f"DELETE FROM Tabla_temporal WHERE Id = ?", valorA0)
            self.db.cursor.commit()
```

Ilustración 39. Código de función crear_grupos (parte 2).

La siguiente sección recorre los demás registros y los asigna a variables para ser comparados al primer registro asignado. Como se observa se van comparando las variables de acuerdo con lo requerido en los grupos.

Las condiciones son:

- Las materias **deben** tener asignado el mismo plan de estudio, carrera, semestre y turno para pertenecer al mismo grupo.
- Las materias **no deben** tener asignado el mismo nombre de materia y hora clase (excluyendo si son en diferentes días).

```

for x in self.info_crear_grupos:
    valorB0= x[0]; valorB1= x[1]; valorB2= x[2]; valorB3= x[3]; valorB4= x[4]; valorB5= x[5]; valorB6= x[6]; valorB7= x[7]; valorB8= x[8]; valorB9= x[9]; valorB10= x[10];
    if valorA0 == valorB0: #comprueba si el id de la primera lista es igual al del segundo
        pass
    elif valorA9 == valorB9:
        if valorA1 == valorB1 and valorA2 != valorB2 and valorA3 != valorB3 and valorA4 == valorB4 and valorA5 == valorB5 and valorA7 != valorB7 and valorA8 == valorB8:
            self.ValorChequeo = valorB7
            self.ValorChequeoA = VALORC0
            self.checar_turno()
            if self.Veredicto == False:
                self.db.cursor.execute(f"INSERT INTO Grupos_ordenados (Id,Grupo_asignado,Plan, Clave,Materia, Carrera, Semestre, Empleado, Hora,Turno,Dias,Salon) VALUES ({valorB0},{valorB1},{valorB2},{valorB3},{valorB4},{valorB5},{valorB6},{valorB7},{valorB8},{valorB9},{valorB10},{valorB11})")
                self.db.cursor.execute(f"DELETE FROM Tabla_temporal WHERE Id = ?", valorB0)
                self.db.cursor.commit()
                self.info_crear_grupos=self.db.cursor.execute(f"SELECT * FROM Tabla_temporal").fetchall()
                self.comprobar_grupos_ordenados=self.db.cursor.execute(f"SELECT * FROM Grupos_ordenados").fetchall()

```

Ilustración 40. Código de función crear_grupos (parte3).

Manual de usuario

Inicio de sesión

La primera ventana que se observa al iniciar la aplicación es la del login, en esta ingresamos los datos que se solicitan, en este caso únicamente pide el nombre de usuario y la contraseña, previamente registrados para poder acceder al sistema.



Ilustración 41. Pantalla inicio de sesión.

Menú de opciones

Después de ingresar al sistema se encuentra la ventana de menú, donde podemos ver la sección de grupos organizados y materias del grupo, así como también los botones de “Editar grupo”, “No disponible” y un apartado donde muestra la carrera, el semestre y el turno mediante estas opciones es posible filtrar los registros.



Ilustración 42. Pantalla Visualización.

En “Mostrar todo” es un grupo de filtros en donde podemos escoger filtrar los campos de “Grupos organizados” por: carrera, semestre y turno, o en su defecto mostrar la lista completa.



Ilustración 43. Opciones de filtro.

En “Grupos organizados” se muestra una lista de los grupos que se dieron de alta previamente, en los registros se pueden observar datos como: Agrupación, Carrera, Turno y Semestre.



Agrupación	Carrera	Turno	Semestre
470	IME	Matutino	1
471	IAS	Matutino	3
472	IME	Vespertino	1
473	IMT	Matutino	3
474	ITS	Nocturno	7
475	IME	Matutino	3
476	IME	Vespertino	2
477	IME	Vespertino	3
478	IME	Vespertino	4
479	IAS	Vespertino	3

Ilustración 44. Tabla de grupos dentro de la aplicación.



Ilustración 45. Tabla de las materias que constituyen la agrupación.

En la sección “Materias del grupo” se puede ver una lista de las materias que tiene asignado cada grupo, con los campos:

- Agrupación es el grupo o paquete de horario seleccionado.
- El plan al que pertenece la materia.
- Clave de la materia.
- Nombre de la materia.
- La carrera a la que pertenece.
- El semestre en el cual se cursa esta materia.
- En empleado se muestra el número de maestro que imparte la materia.
- La hora a la cual se imparte la clase.
- El día en el que se da la clase.
- El salón.



Ilustración 46. Funciones de la pantalla Visualizar.

Agregar grupo

Al agregar grupo se ingresan los datos solicitados de la materia para añadirla a un grupo. Los datos solicitados son:

- Plan al que pertenece
- Clave de la materia
- Nombre de la materia
- Carrera a la que pertenece
- La hora a la que se imparte la clase
- El semestre del grupo
- La frecuencia de la materia
- El número de empleado que imparte la materia
- Y el salón donde se dará la clase

Clic en menú para regresar al menú principal

Cantidad de registros: 15

Agregar grupo

Plan:

Clave:

Materia:

Carrera: Hora:

Semestre: Día:

Frecuencia:

Empleado:

Salón:

Aceptar

Clic en aceptar para guardar los cambios.

Materias registradas

ID Grupo	Plan	Clave	Materia	Carrera	Semestre	Empleado	Hora	Día	Salas
1	401	120	Química	QAB	I	215	MA	1-AA-V	2-122
2	401	120	Física	QAB	I	125	MA	1-AA-V	21
3	401	31	Matemáticas	QAB	I	125	MA	1-AA-V	2-122
4	401	231	Algebra	QAB	I	1159	MA	1-AA-V	1-2123
5	401	234	Física	QAB	I	2123	MA	1-AA-V	2123
7	401	440	Química	QAB	I	208	VI	1-AA-V	212
8	401	41	Física 3	QAB	I	440	MA	MA-I	2-122
9	401	525	Física 4	QAB	I	222	MA	MA-I	3-124
10	401	5245	Química	QAB	I	401	MA	MA-I	3-123
11	401	221	(Química)	QAB	I	40123	MA	MA-I	3-123
12	401	1110	NO	QAB	I	40123	MA	MA-I	4123

Editar registro Eliminar registro

Ilustración 47. Funciones de la pantalla CRUD (parte 1).

Posteriormente el registro aparecerá en una lista en el apartado de materias registradas.

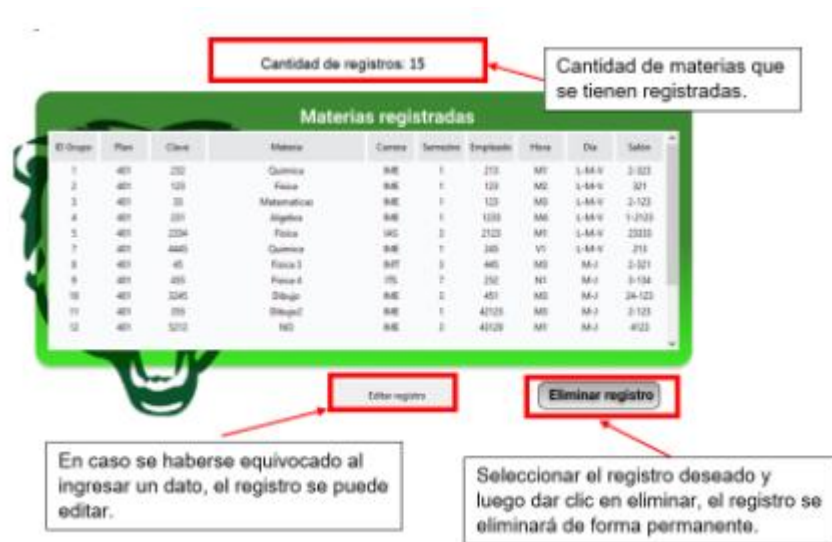


Ilustración 48. Funciones de la pantalla CRUD (parte 2).

Si se desea editar algún dato únicamente se selecciona el registro deseado y se da clic en el botón “Editar registro”, aparecerá la siguiente pantalla, donde se ingresan nuevamente los datos.



Ilustración 49. Función de la pantalla Actualizar.