PAC

MEMORIA

PROYECTO

INTEGRADOR

Paula Moure, Amelia Mª Coca y Juan Cortés

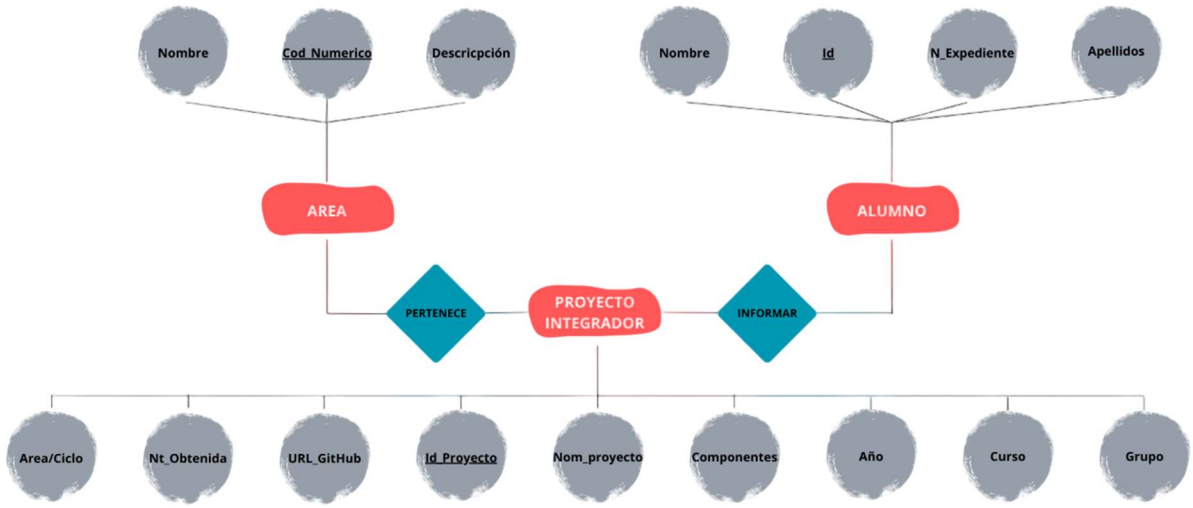
BASES DE DATOS

Entidad relación

La creación de un modelo entidad-relación (ER) en una base de datos es esencial para el diseño conceptual, la comunicación, la identificación de requisitos, la normalización y el mantenimiento del sistema.

El modelo ER proporciona una representación visual de las entidades y relaciones, permitiendo a los diseñadores comprender la estructura y las interacciones de los datos. Facilita la colaboración entre usuarios y diseñadores, ayudando a establecer requisitos de información precisos. Además, el modelo ER sirve como base para el diseño lógico de la base de datos, garantizando una estructura coherente y eficiente. Finalmente, el modelo ER permite gestionar el mantenimiento y la evolución del sistema, brindando una referencia visual para realizar cambios de manera efectiva.

En la entidad relación hemos creado tres entidades Área, Alumno y Proyecto Integrador. Alumno tiene la relación 'realizar' con Proyecto Integrador, que a su vez tiene la relación 'informar' con Área. Cada una de las entidades tiene sus atributos propios.



Normalización

Al hacer la definición de las entidades y los atributos, hemos identificado las entidades que forman parte del sistema y hemos establecido los atributos que las caracterizan. Ahora vamos a explicar todo con más detalle lo que he hecho:

Identificación de entidades: Identificamos las tres entidades principales que son: "Área o ciclo", "Alumno" y "Proyecto Integrador".

Definición de atributos: Por cada entidad, identificamos los atributos relevantes que describen cada entidad de manera única. Por ejemplo, en "Área o ciclo", los atributos son "Código numérico único", "Nombre" y "Descripción".

Clave primaria (PK): Para cada entidad, hemos indicado el atributo que se utilizará como clave primaria (PK). La clave primaria es un atributo único que identifica de manera exclusiva cada registro en una tabla.

Clave foránea (FK): En el caso de las relaciones entre entidades, hemos mencionado a las claves foráneas (FK) para establecer conexiones entre las tablas. Las claves foráneas permiten relacionar los registros de una tabla con los registros de otra tabla.

Forma Normal 1

Tabla Área/ Ciclo

COD_AREA	NOMBRE	DESCRIPCIÓN
1	DAW	Desarrollo de Aplicaciones Web
2	DAM	Desarrollo de Aplicaciones Multimedia
3	ASIR	Administración de Sistemas Informáticos en Red

Tabla Alumnos

NUM_EXPED	ID_NOMBRE	APELLIDOS
10001	Juan	Pérez
10002	Ana	Gómez
10003	Pedro	Sánchez
10004	María	López
10005	Francisco	Rodríguez

Tabla Proyecto Integrador

ID_PROYECTO	NOMBRE_PROYECTO	URL_GITHUB	NOTA_OBTENIDA	AÑO	CURSO	GRUPO	***COD_AREA***
1	Proyecto 1	https://github.com/proyecto1	9	2022	1	M11	1
2	Proyecto 2	https://github.com/proyecto1	8,5	2022	1	M22	2
3	Proyecto 3	https://github.com/proyecto1	7	2022	2	T1	1

Tabla AlumnosProyecto

NUM_EXPED	***ID_PROYECTO***
10001	1
10002	1
10003	1
10004	2
10005	3

Forma Normal 2

Tabla Alumnos

NUM_EXPED	**ID_NOMBRE**	APELLIDOS

Tabla Proyecto Integrador

ID_PROYECTO	NOMBRE_PROYECTO	URL_GITHUB	NOTA_OBTENIDA	AÑO	CURSO	GRUPO	***COD_AREA***

Tabla Área/ Ciclo

COD_AREA	NOMBRE	DESCRIPCIÓN

Forma Normal 3

Tabla Alumnos

NUM_EXPED	**ID_NOMBRE**	APELLIDOS

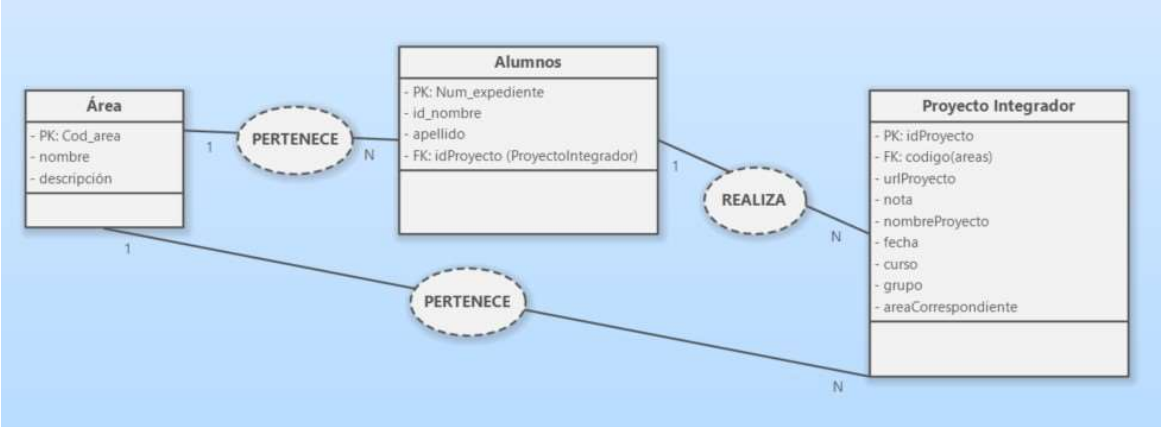
Tabla Proyecto Integrador

ID_PROYECTO	NOMBRE_PROYECTO	URL_GITHUB	NOTA_OBTENIDA	AÑO	CURSO	GRUPO	***COD_AREA***

Tabla Área/ Ciclo

COD_AREA	NOMBRE	DESCRIPCIÓN

Modelo Relacional



Hemos hecho este modelo relacional donde mostramos que a un **área** pueden *pertenecer* varios **alumnos** (*relación 1 a N*), un **alumno** puede *realizar* varios **proyectos integradores** (relación 1 a N) y cada proyecto integrador pertenece a un área específica (relación N a 1).

Creación de la BBDD

Hemos hecho un script SQL donde creamos y llenamos varias tablas relacionadas en una base de datos. A continuación, se explica cada parte del código:
Se inicia con la instrucción "*DROP TABLE IF EXISTS*" para eliminar las tablas existentes en caso de que ya estén creadas en la base de datos. Esto se hace para evitar errores si se ejecuta el script más de una vez.
A continuación, se crean tres tablas: "areas", "proyectoIntegrador" y "alumno".

La tabla "***areas***" tiene tres columnas: "*codigo*" (entero de 5 dígitos y clave primaria), "*nombre*" (cadena de hasta 30 caracteres) y "*descripcion*" (cadena de hasta 50 caracteres). Esta tabla representa las áreas de estudio relacionadas con los proyectos integradores.

La tabla "***proyectoIntegrador***" tiene varias columnas: "*idProyecto*" (entero de 5 dígitos y clave primaria), "*nombreProyecto*" (cadena de hasta 30 caracteres), "*urlProyecto*" (cadena de hasta 100 caracteres), "*nota*" (flotante con 2 decimales), "*fecha*" (fecha), "*curso*" (entero de 4 dígitos), "*grupo*" (cadena de hasta 50 caracteres) y "*areaCorrespondiente*" (entero de 5 dígitos que se relaciona con el código de área en la tabla "areas").

La tabla "***alumno***" tiene cinco columnas: "*codigo*" (entero de 5 dígitos y clave primaria), "*numExpediente*" (entero de 5 dígitos), "*nombre*" (cadena de hasta 20 caracteres), "*apellidos*" (cadena de hasta 20 caracteres) e "*idProyecto*" (entero de 5 dígitos).
La columna "*idProyecto*" se relaciona con la clave primaria "idProyecto" de la tabla "proyectoIntegrador" mediante una restricción de clave externa ("idProyecto_fk").

Y por último creamos la tabla "**administrado**". Esta tabla tiene tres columnas: "*idAdministrador*" (entero de 5 dígitos y clave primaria), "*username*" (cadena de hasta 20 caracteres), y "*pass*" (cadena de hasta 20 caracteres).

```sql
drop table if exists alumno;
drop table if exists proyectoIntegrador;
drop table if exists areas;
drop table if exists administrado;


create table areas (
    codigo int(5) primary key,
    nombre varchar(30),
    descripcion varchar(50)
);

create table proyectoIntegrador (
    idProyecto int(5) primary key not null,
    nombreProyecto varchar(30),
    urlProyecto varchar(100),
    nota float(2),
    fecha date,
    curso int(4),
    grupo varchar(50),
    areaCorrespondiente int(5),
    constraint proyecto_area_fk foreign key (areaCorrespondiente) references areas(codigo)
);

create table alumno (
    numExpediente int(5)primary key,
    codigo int(5) ,
    nombre varchar(20),
    apellidos varchar(20),
    idProyecto int (5),
    constraint idProyecto_fk foreign key (idProyecto) references proyectoIntegrador (idProyecto)
);

create table administrado (
    idAdministrador int(5) primary key,
    username varchar(20),
    pass varchar(20)
);
```

Después de crear las tablas, se ejecutan instrucciones "*INSERT INTO*" para insertar datos en las tablas.
Se insertan filas en la tabla "areas" con los códigos, nombres y descripciones correspondientes a tres áreas distintas.

Se insertan filas en la tabla "*proyectoIntegrador*" con los detalles de varios proyectos integradores. Cada proyecto tiene un ID único, nombre, URL, nota, fecha, curso, grupo y área correspondiente.

Finalmente, se insertan filas en la tabla "*alumno*" con detalles de varios alumnos, incluyendo su código, número de expediente, nombre, apellidos e ID del proyecto integrador al que están asignados.

```sql
INSERT INTO areas (codigo, nombre, descripcion) VALUES (1, 'DAW', 'Desarrollo de Aplicaciones Web');
INSERT INTO areas (codigo, nombre, descripcion) VALUES (2, 'DAM', 'Desarrollo de Aplicaciones Multiplataforma');
INSERT INTO areas (codigo, nombre, descripcion) VALUES (3, 'ASIR','Administración de Sistemas Informáticos en Red');
```

```sql
INSERT INTO proyectoIntegrador (idProyecto, nombreProyecto, urlProyecto, nota, fecha, curso, grupo, areaCorrespondiente)
VALUES (001, 'Proyecto Integrador 1', 'https://github.com/Proyecto_Integrador1.git', 8.5, CURRENT_DATE(), 2, 'Grupo PAC', 1);
INSERT INTO proyectoIntegrador (idProyecto, nombreProyecto, urlProyecto, nota, fecha, curso, grupo, areaCorrespondiente)
VALUES (002, 'Proyecto Integrador 2', 'https://github.com/Proyecto_Integrador2.git', 9, CURRENT_DATE(), 1, 'Grupo Energía Creativa', 2);
INSERT INTO proyectoIntegrador (idProyecto, nombreProyecto, urlProyecto, nota, fecha, curso, grupo, areaCorrespondiente)
VALUES (003, 'Proyecto Integrador 3', 'https://github.com/Proyecto_Integrador3.git', 7.5, CURRENT_DATE(), 2, 'Grupo Mentes Alquímicas', 3);
INSERT INTO proyectoIntegrador (idProyecto, nombreProyecto, urlProyecto, nota, fecha, curso, grupo, areaCorrespondiente)
VALUES (004, 'Proyecto Integrador 4', 'https://github.com/Proyecto_Integrador4.git', 5, CURRENT_DATE(), 2, 'Grupo Los Visionarios', 2);
INSERT INTO proyectoIntegrador (idProyecto, nombreProyecto, urlProyecto, nota, fecha, curso, grupo, areaCorrespondiente)
VALUES (005, 'Proyecto Integrador 5', 'https://github.com/Proyecto_Integrador5.git', 5, CURRENT_DATE(), 1, 'Grupo PAC', 1);
INSERT INTO proyectoIntegrador (idProyecto, nombreProyecto, urlProyecto, nota, fecha, curso, grupo, areaCorrespondiente)
VALUES (006, 'Proyecto Integrador 6', 'https://github.com/Proyecto_Integrador6.git', 4, CURRENT_DATE(), 2, 'Grupo Los Cazadores de Tesoros', 2);
INSERT INTO proyectoIntegrador (idProyecto, nombreProyecto, urlProyecto, nota, fecha, curso, grupo, areaCorrespondiente)
VALUES (007, 'Proyecto Integrador 7', 'https://github.com/Proyecto_Integrador7.git', 6, CURRENT_DATE(), 1, 'Grupo Los Soñadores Perdidos', 3);
INSERT INTO proyectoIntegrador (idProyecto, nombreProyecto, urlProyecto, nota, fecha, curso, grupo, areaCorrespondiente)
VALUES (008, 'Proyecto Integrador 8', 'https://github.com/Proyecto_Integrador8.git', 9.5, CURRENT_DATE(), 2, 'Grupo Los Buscadores de la Verdad', 2);
INSERT INTO proyectoIntegrador (idProyecto, nombreProyecto, urlProyecto, nota, fecha, curso, grupo, areaCorrespondiente)
VALUES (009, 'Proyecto Integrador 9', 'https://github.com/Proyecto_Integrador9.git', 8.5, CURRENT_DATE(), 2, 'Grupo Los Maestros del Sonido', 1);
INSERT INTO proyectoIntegrador (idProyecto, nombreProyecto, urlProyecto, nota, fecha, curso, grupo, areaCorrespondiente)
VALUES (010, 'Proyecto Integrador 10', 'https://github.com/Proyecto_Integrador10.git', 9, CURRENT_DATE(), 1, 'Grupo Los Aventureros del Bosque Encantado', 2);
```

```sql
INSERT INTO alumno (codigo, numExpediente, nombre, apellidos, idProyecto) VALUES (1, 11122, 'Amelia María', 'Coca López', 001);
INSERT INTO alumno (codigo, numExpediente, nombre, apellidos, idProyecto) VALUES (2, 22233, 'Juan', 'Cortés Cortés', 001);
INSERT INTO alumno (codigo, numExpediente, nombre, apellidos, idProyecto) VALUES (3, 33344, 'Paula', 'Moure Prado', 001);
INSERT INTO alumno (codigo, numExpediente, nombre, apellidos, idProyecto) VALUES (6, 66677, 'James Andrew', 'BEHAN', 002);
INSERT INTO alumno (codigo, numExpediente, nombre, apellidos, idProyecto) VALUES (7, 77788, 'Ismael', 'Bodas Díaz', 002);
INSERT INTO alumno (codigo, numExpediente, nombre, apellidos, idProyecto) VALUES (8, 88899, 'Jorge', 'Burgos Barrera', 002);
INSERT INTO alumno (codigo, numExpediente, nombre, apellidos, idProyecto) VALUES (9, 99900, 'Ali', 'Chalak', 003);
INSERT INTO alumno (codigo, numExpediente, nombre, apellidos, idProyecto) VALUES (10, 10011, 'JAVIER', 'Chicano Miguel' ,003);
INSERT INTO alumno (codigo, numExpediente, nombre, apellidos, idProyecto) VALUES (11, 11022, 'Santiago Andres', 'Daza Villamizar',003);
INSERT INTO alumno (codigo, numExpediente, nombre, apellidos, idProyecto) VALUES (12, 12233, 'Amir Mahdi', 'Dorrani',004);
INSERT INTO alumno (codigo, numExpediente, nombre, apellidos, idProyecto) VALUES (13, 13344, 'Aarón Juan', 'Escudero Navas',004);
INSERT INTO alumno (codigo, numExpediente, nombre, apellidos, idProyecto) VALUES (14, 14455, 'Daniel', 'Garrido Nuñez',004);
INSERT INTO alumno (codigo, numExpediente, nombre, apellidos, idProyecto) VALUES (15, 15566, 'Daniel', 'Herrero Martínez',005);
INSERT INTO alumno (codigo, numExpediente, nombre, apellidos, idProyecto) VALUES (16, 16677, 'Alejandro', 'Junyent Romani',005);
INSERT INTO alumno (codigo, numExpediente, nombre, apellidos, idProyecto) VALUES (17, 17788, 'Aris', 'Maximilian Kuhs',005);
INSERT INTO alumno (codigo, numExpediente, nombre, apellidos, idProyecto) VALUES (18, 18899, 'Lucca', 'Manfredotti García',006);
INSERT INTO alumno (codigo, numExpediente, nombre, apellidos, idProyecto) VALUES (19, 19900, 'Ismael', 'Moreno Moral',006);
INSERT INTO alumno (codigo, numExpediente, nombre, apellidos, idProyecto) VALUES (20, 20011, 'Juan Diego', 'Motta Moncada',006);
INSERT INTO alumno (codigo, numExpediente, nombre, apellidos, idProyecto) VALUES (21, 21122, 'Pablo', 'Naranjo Cid',007);
INSERT INTO alumno (codigo, numExpediente, nombre, apellidos, idProyecto) VALUES (22, 22234, 'David', 'Oñate Sánchez',007);
INSERT INTO alumno (codigo, numExpediente, nombre, apellidos, idProyecto) VALUES (23, 23344, 'Mario', 'Robles Padua',007);
INSERT INTO alumno (codigo, numExpediente, nombre, apellidos, idProyecto) VALUES (24, 24455, 'Gabriel Enrique', 'Rodríguez Padrón',008);
INSERT INTO alumno (codigo, numExpediente, nombre, apellidos, idProyecto) VALUES (25, 25566, 'David', 'Rojo Villalba',008);
INSERT INTO alumno (codigo, numExpediente, nombre, apellidos, idProyecto) VALUES (26, 26677, 'Celia', 'Rubio Pais',008);
INSERT INTO alumno (codigo, numExpediente, nombre, apellidos, idProyecto) VALUES (27, 27788, 'Miguel', 'Sánz Villanueva',009);
INSERT INTO alumno (codigo, numExpediente, nombre, apellidos, idProyecto) VALUES (28, 28899, 'Elena', 'Saugar Mayoral',009);
INSERT INTO alumno (codigo, numExpediente, nombre, apellidos, idProyecto) VALUES (29, 29900, 'Álvaro', 'Serrano Colomer',009);
INSERT INTO alumno (codigo, numExpediente, nombre, apellidos, idProyecto) VALUES (30, 30011, 'Josep', 'Serrano Rayó',010);
INSERT INTO alumno (codigo, numExpediente, nombre, apellidos, idProyecto) VALUES (31, 31122, 'Wei', 'Xu',010);
INSERT INTO alumno (codigo, numExpediente, nombre, apellidos, idProyecto) VALUES (32, 32233, 'Guoanqi', 'Yin',010);
```

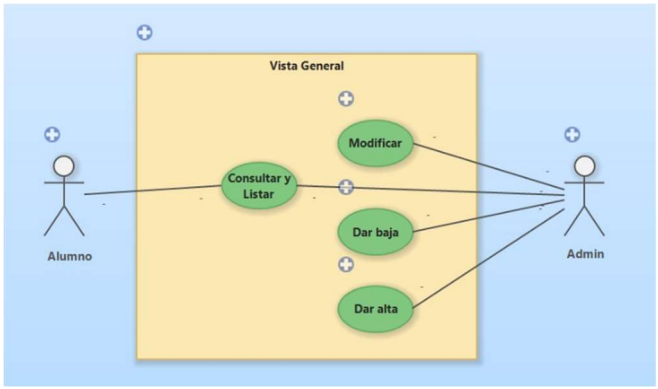
```sql
INSERT INTO administrado (idAdministrador, username, pass) VALUES (0512, 'SaraVillanueva', 'SaraProfesora');
INSERT INTO administrado (idAdministrador, username, pass) VALUES (4321, 'RaquelCerda', 'RaquelProfesora');
```

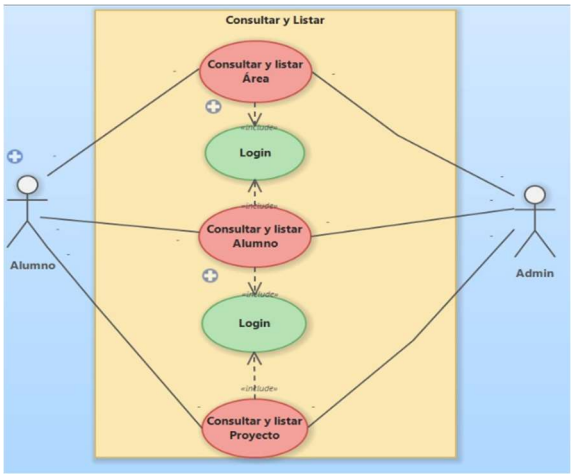
ENTORNOS DEL DESARROLLO

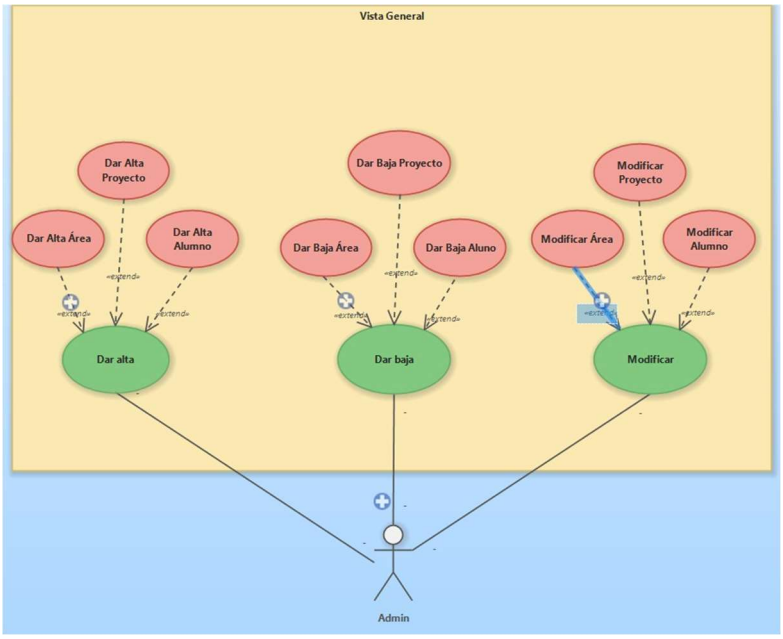
Diagrama de Casos de Uso

Los diagramas de casos de uso se crean para representar visualmente las interacciones entre un sistema y sus actores, identificar los casos de uso relevantes y capturar los requisitos funcionales. Ayudan a definir los actores principales, mostrar las funcionalidades del sistema, visualizar las interacciones entre actores y casos de uso, y capturar los requisitos funcionales del sistema. En resumen, los diagramas de casos de uso son herramientas visuales para comprender y comunicar los requerimientos funcionales de un sistema.

- Creamos tres vistas, en la Vista General el Alumno podrá hacer consultas y lisar, y el Admin, podrá modificar, dar baja, dar alta y consultar y listar.
- En la vista Consultar y Listar tanto el Admin como el alumno, podrá consultar y listar, tendrán que hacer un Login para poder acceder.
- Vista Admin, podrá acceder altas, bajas y modificación, dentro de esos campos las opciones serán Proyecto Integrador, Alumno y Área.





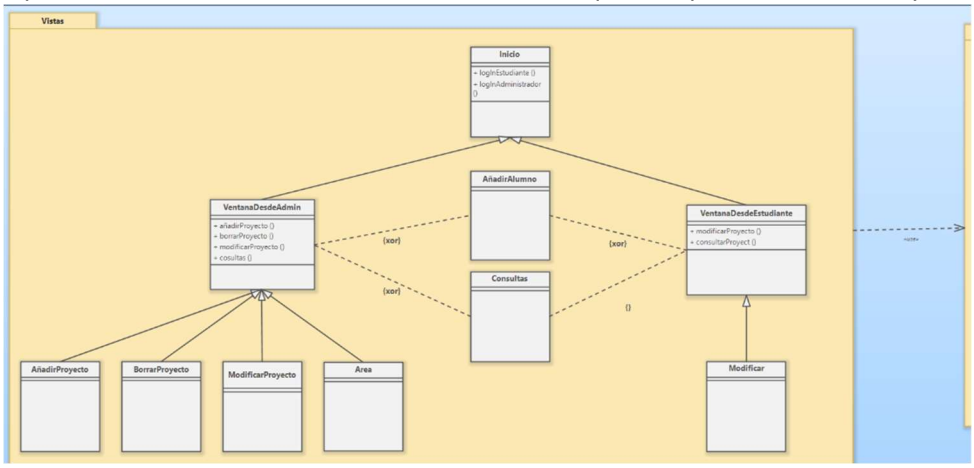


Generación del diagrama de clases

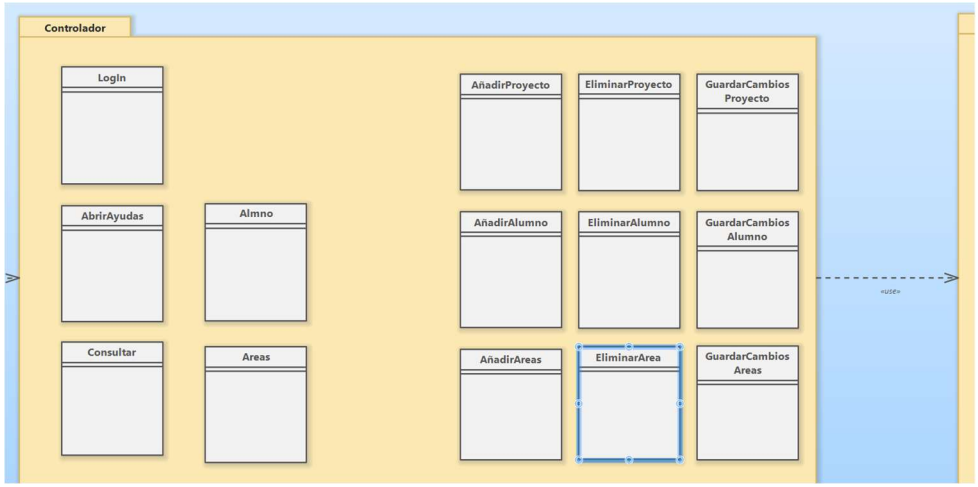
La generación del diagrama de clases es una etapa fundamental en el diseño orientado a objetos y se utiliza para representar visualmente la estructura y las relaciones entre las clases de un sistema.

Hemos creado tres paquetes, Controlador, Vista y Modelo.

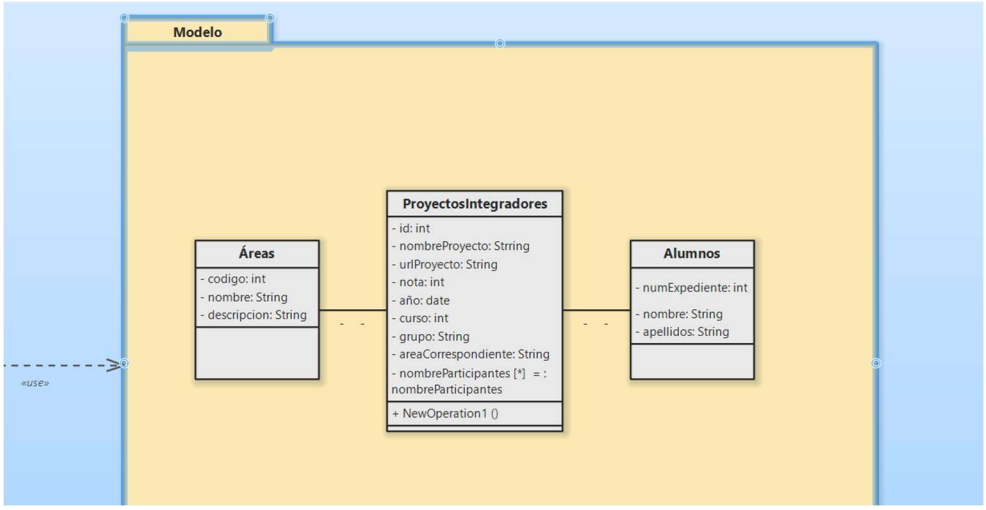
En el paquete Vista añadimos todas las todas las ventas que componen nuestro Proyecto



En el paquete Controlador, añadimos los botones que tendrán las interfaces gráficas, los cuales nos llevarán a distintas ventanas.



El paquete Modelo estarán las clases principales, de las que se compondrá el Proyecto, que son Áreas, ProyectoIntegrador y Alumnos.

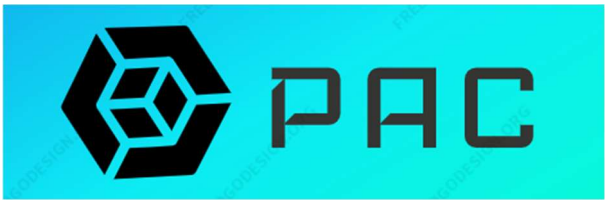


Diseño del logo: Sprint 2.

Lo primero es el nombre del Proyecto Integrador, PAC, refiriéndose a las iniciales de los integrantes, Paula, Amelia y Cortés.

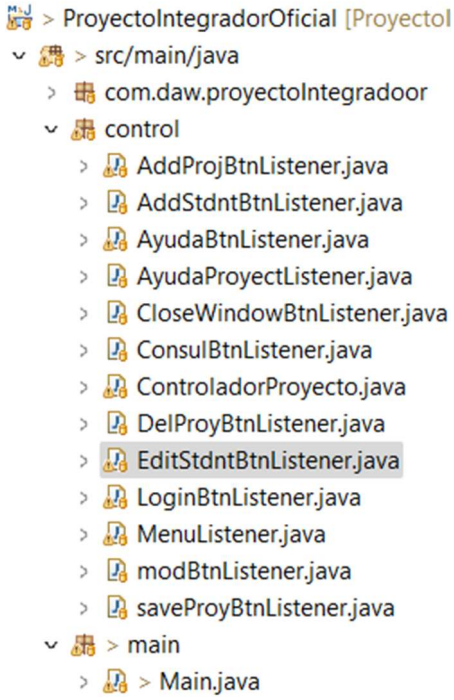
Añadimos una un hexágono compuesto por tres partes, ya que el número de integrantes es de 3, y un último hexágono con otras tres formas, pero distintas a las tres primeras. Haciendo la unión de estos dos hexágonos creamos el logo.

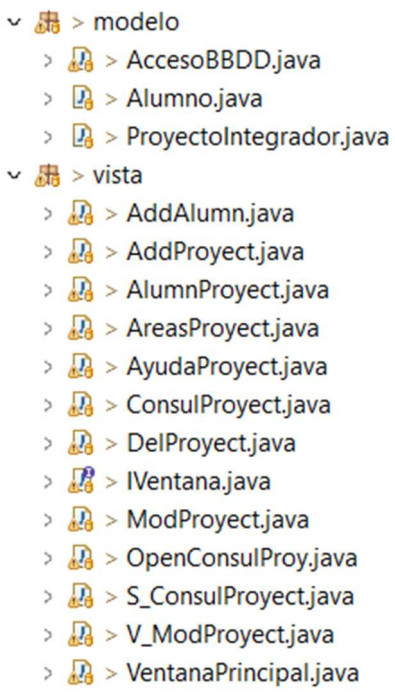
Por último, añadimos un fondo, compuesto de una gama de azules, haciendo un degradado de tonos más oscuros desde la esquina izquierda superior, hasta la opuesta.



PROGRAMACIÓN

En la clase de control, se han creado listeners y botones que se utilizan para interactuar con la vista y garantizar el funcionamiento del proyecto. Estos listeners están vinculados a eventos específicos, como hacer clic en un botón, y ejecutan las acciones correspondientes en el sistema. Los botones son importados a la vista para permitir al usuario interactuar con las funcionalidades ofrecidas por el proyecto.





INTERFACES:

LOGIN:

Esta interfaz representa la pantalla de inicio de sesión de la aplicación.
Permite a los usuarios ingresar sus credenciales para acceder al sistema.
Los administradores tienen la capacidad de crear una nueva venta y llamarla.
Los usuarios normales también pueden crear una venta, pero no tienen acceso a todas las funcionalidades administrativas.
Existe un botón de "Login" que, al ser presionado, activa el "LISTERNER LOGIN". Este listener se encarga de verificar los datos introducidos por el usuario con los almacenados en la base de datos. Si los datos son correctos, se permite el acceso a la interfaz correspondiente.

PROYECTO:

ADMIN-CONSULTAR Proyecto:

Esta interfaz está diseñada para los administradores y les permite consultar la información de los proyectos existentes.
En esta interfaz, se muestra una lista (JList) que contiene los nombres de los proyectos disponibles.
Al hacer clic en el botón "Consultar", que está asociado con el "LISTERNER CONSUTBOTTON", se abre una nueva interfaz llamada "OPENCONSULT" que muestra información detallada sobre el proyecto seleccionado. Por ejemplo, puede mostrar detalles como el nombre del proyecto, descripción, fecha de inicio, etc.

ADMIN-AÑADIR Proyecto:
Esta interfaz permite a los administradores agregar nuevos proyectos al sistema.
Muestra una etiqueta (JLabel) con información relacionada con el proyecto.
Proporciona un campo de texto (JTextField) donde los administradores pueden ingresar la información del proyecto, como el nombre, la descripción, la fecha de inicio, etc.
Existe un botón de "Añadir" que, cuando se presiona, activa el "LISTERNER AÑADIR". Este listener utiliza el método "a" del acceso a la base de datos para insertar los datos del proyecto en la base de datos.

ADMIN-ELIMINAR Proyecto:
Esta interfaz permite a los administradores eliminar proyectos existentes del sistema.
Muestra una lista (JList) que contiene los nombres de los proyectos disponibles.
Al seleccionar un proyecto y hacer clic en el botón "Eliminar", que está asociado con el "LISTENER ELIMINAR", se activa el método "b" del acceso a la base de datos para eliminar el proyecto seleccionado de la base de datos.

ADMIN-EDITAR Proyecto:
Esta interfaz permite a los administradores editar la información de los proyectos existentes.
Muestra una lista (JList) que contiene los nombres de los proyectos disponibles.
Al seleccionar un proyecto y hacer clic en el botón "Editar", que está asociado con el "LISTERNER EDIT", se abre una nueva interfaz llamada "OPENEDIT". Esta interfaz permite realizar cambios en la información del proyecto seleccionado, como modificar el nombre, la descripción, la fecha de inicio, etc.

ALUMNO:

ADMIN-CONSULTAR Alumno:
Esta interfaz permite a los administradores consultar la información de los alumnos registrados en el sistema.
Muestra una lista (JList) que contiene los nombres de los alumnos disponibles.
Al hacer clic en el botón "Añadir", que está asociado con el "LISTERNER ADDBOTTON", se abre una nueva interfaz que permite agregar información adicional del alumno. Esta interfaz se llama "OPENAEDITARALUMNO". En esta interfaz, se pueden ingresar datos como el nombre del alumno, la dirección, la fecha de nacimiento, etc.
También hay un botón de "Editar" asociado con el "LISTERNER EDITBOTTON". Al hacer clic en este botón, se abre una nueva interfaz para editar la información del alumno seleccionado.
CONSULTAR:

ALUMNO-CONSULTAR Proyectos:
Esta interfaz está diseñada para que los alumnos puedan consultar los proyectos disponibles en el sistema.
Muestra una lista (JList) que contiene los nombres de los proyectos disponibles.
Al hacer clic en el botón "Consultar", que está asociado con el "LISTERNER ADDBOTTON", se abre una nueva interfaz llamada "OPENCONSULTARPROYECTOS". Esta interfaz muestra información detallada sobre el proyecto seleccionado, como su descripción, los miembros del equipo, la fecha de inicio, etc.

AYUDA:

Ayuda-Usuarios:

Esta interfaz proporciona ayuda a los usuarios respondiendo preguntas frecuentes o consultas relacionadas con el sistema.

Muestra una lista (JList) que contiene las preguntas o temas de ayuda disponibles.

Al hacer clic en el botón "Ayuda", que está asociado con el "AYUDABTNLISTENNER", se muestra un mensaje de advertencia que contiene la respuesta o información relevante a la pregunta seleccionada.