



Ingeniería de Software II

Manual de Usuario

Grupo: 8CC2

Semestre: Agosto - Diciembre

Profesor: Cordero de los Ríos Perla Ivonne

Equipo 1#

Martín Eduardo Chacón Orduño - 351840

Allan Hall Solorio - 358909

Marco Iram López Contreras - 367689

Jorge Alejandro Beltran Rosales - 348635

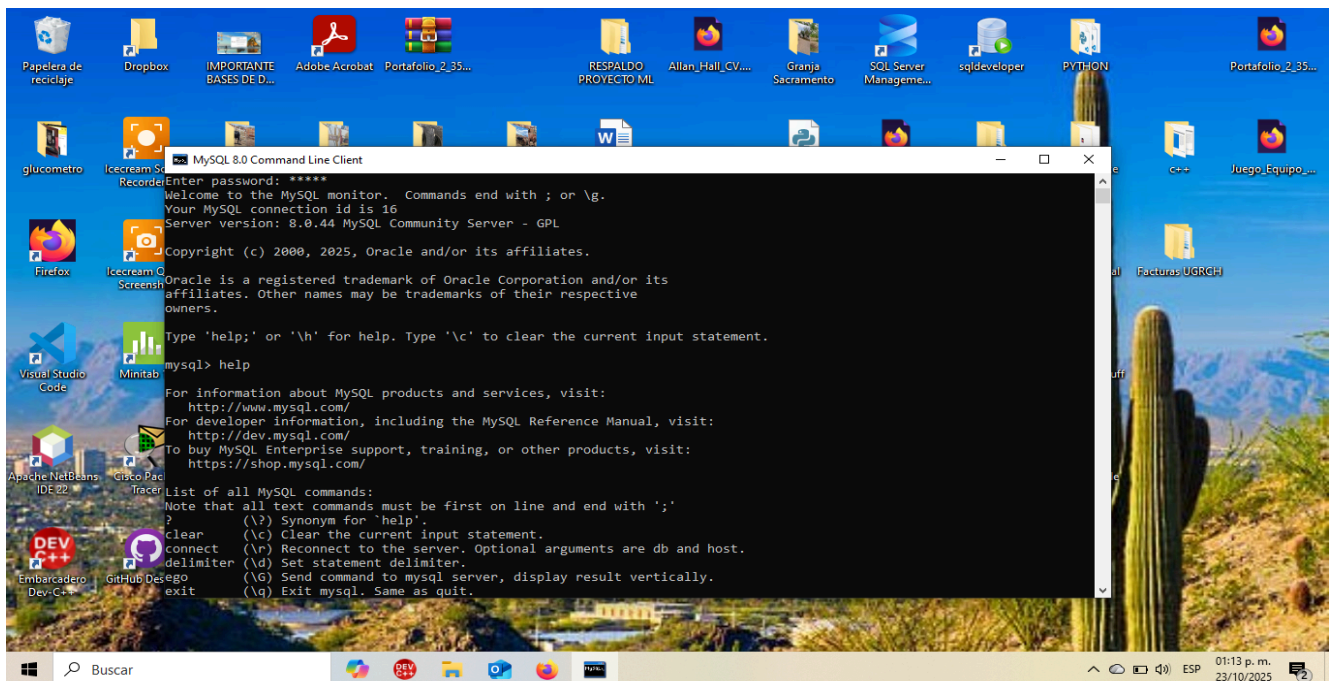
Fecha de entrega: 27/10/2025

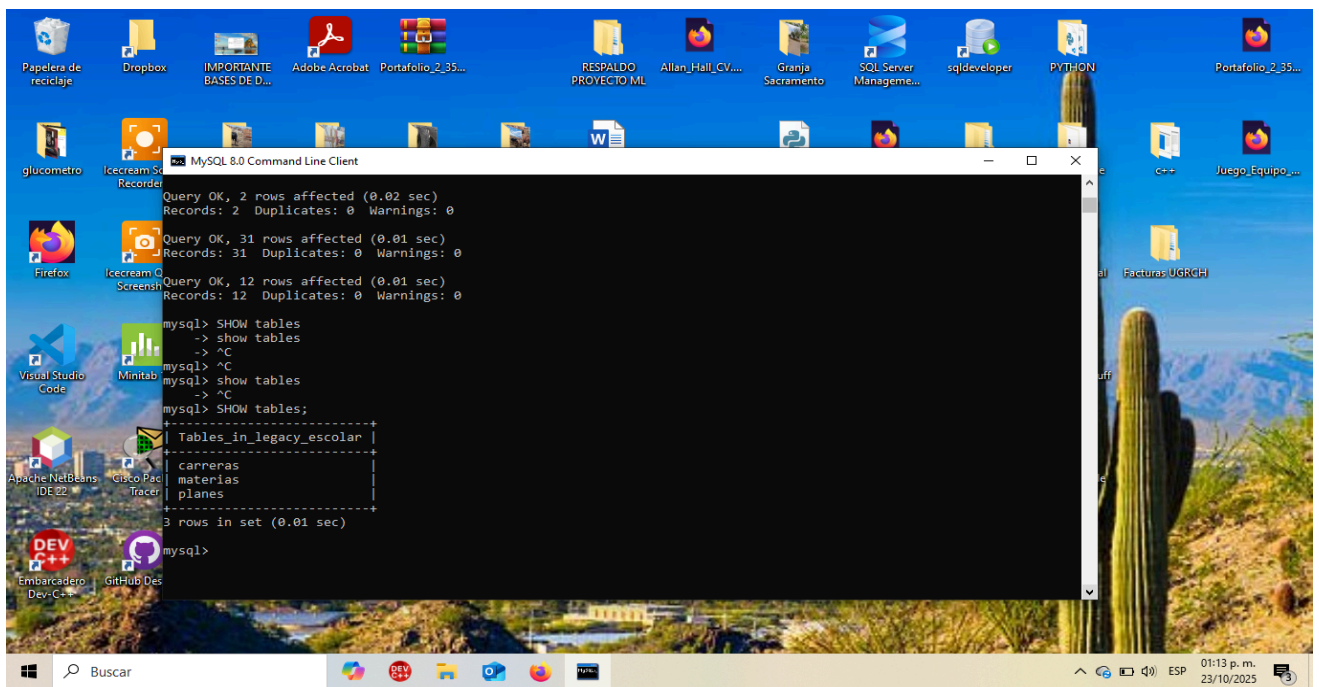
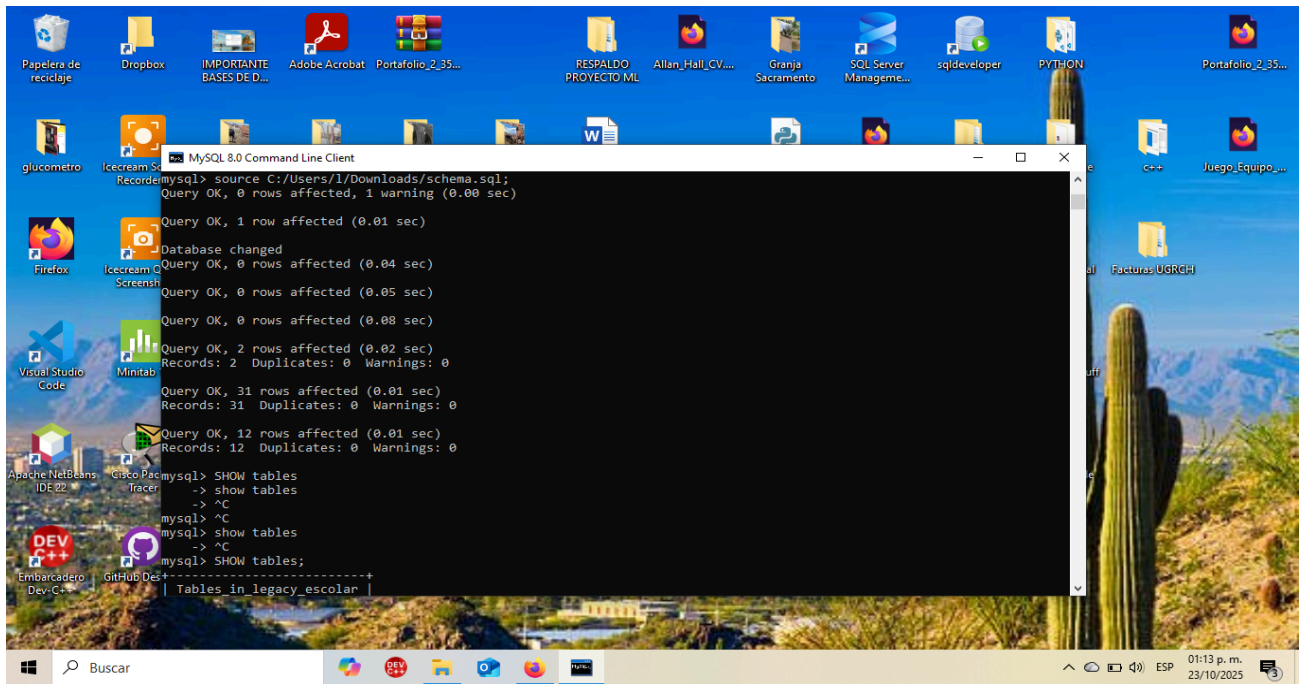
Manual de Usuario

Este manual de usuario es sobre cómo se aplicó reingeniería de sistemas en base al archivo .rar proporcionado

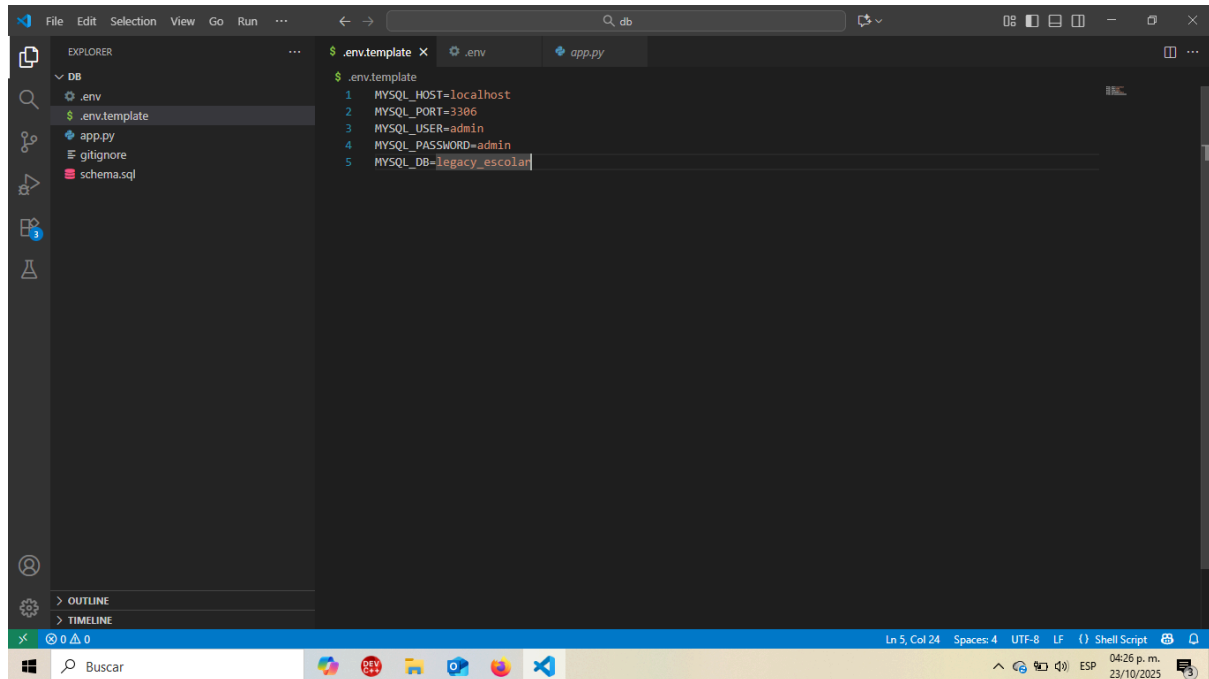
Creación y ejecución de la base de datos (mysql)

Aquí lo que se hizo fue pasar los datos proporcionados a una base de datos de mysql





Ejecución



Implementación en Python

```
import os
import re
import sys
from typing import Optional
from mysql.connector.connection import MySQLConnection
from dotenv import load_dotenv
import mysql.connector
from mysql.connector import Error
from tabulate import tabulate

load_dotenv()

DATABASE_CONFIG = {
    "host": os.getenv("MYSQL_HOST", "localhost"),
    "port": int(os.getenv("MYSQL_PORT", 3306)),
    "user": os.getenv("MYSQL_USER", "root"),
    "password": os.getenv("MYSQL_PASSWORD", ""),
    "database": os.getenv("MYSQL_DB", "legacy_escolar"),
    "autocommit": False,
}
```

```

# conexion sql
def conectar_db() -> MySqlConnection:
    """
    Crear conexion con mysql
    """
    return mysql.connector.connect(**DATABASE_CONFIG)

# Helpers
def formtatear_semestre(valor:str) -> Optional[str]:
    """
    Valida y formatea el semestre (01 - 10)
    """
    if re.fullmatch(r"(0?[1-9]|10)", valor):
        return f"{int(valor):02d}"
    return None

def formatear_fecha(cadena:str) -> Optional[str]:
    """
    Convierte fechas con formato DD/MM/YYYY a YYYY-MM-DD
    """
    cadena = cadena.strip()
    if not cadena or cadena.upper() == "NULL":
        return None

    if re.fullmatch(r"\d{4}-\d{2}-\d{2}", cadena):
        return cadena

    match = re.fullmatch(r"(\d{2})/(\d{2})/(\d{4})", cadena)
    if match:
        dia, mes, anio = match.groups()
        return f"{anio}-{mes}-{dia}"

    print("Formato de fecha invalido. Usa YYYY-MM-DD o DD/MM/YYYY")
    return None

# Consulta a bd
def carrera_existe(conn: MySqlConnection, clave_carrera: str) -> bool:
    """
    Verifica si una carrera existe en la base de datos
    """
    query = "SELECT 1 FROM carreras WHERE clave = %s"

```

```

        with conn.cursor() as cursor:
            cursor.execute(query, (clave_carrera,))
            return cursor.fetchone() is not None

def mostrar_carreras(conn: MySQLConnection):
    """
    Muestra todas las carreras
    """
    query = "SELECT clave, nombre FROM carreras ORDER BY clave"
    with conn.cursor() as cursor:
        cursor.execute(query)
        carreras = cursor.fetchall()

    print("\n=== Carreras Disponibles ===")
    print(tabulate(carreras, headers=["Clave", "Nombre"],
tablefmt="github"))

def obtener_materias(conn: MySQLConnection):
    """
    Obtiene todas las materias disponibles y las muestra en una tabla
    """
    query = "SELECT clave, descri FROM materias ORDER BY clave"
    with conn.cursor() as cursor:
        cursor.execute(query)
        materias = cursor.fetchall()

    print("\n=== MATERIAS DISPONIBLES ===")
    print(tabulate(materias, headers=["Clave", "Descripción"],
tablefmt="github"))
    return {str(fila[0]) for fila in materias}

def mostrar_planes(conn: MySQLConnection):
    """
    Muestra todos los planes
    """
    query = "SELECT * FROM planes ORDER BY id"
    with conn.cursor() as cursor:
        cursor.execute(query)
        resultados = cursor.fetchall()
        encabezados = ["ID", "Carrera", "Materia", "semestre", "Fecha
alta", "Fecha baja"]

    print("\n=== PLANES DE ESTUDIO ===")

```

```

print(tabulate(resultados, headers=encabezados, tablefmt="github"))

# Operaciones CRUD
def agregar_plan(conn: MySQLConnection):
    """
    Agrega nuevas materias a un plan de estudios
    """
    mostrar_carreras(conn)
    clave_carrera = input("\nClave de carrera (ej. 05, 07): ").strip()
    if not carrera_existe(conn, clave_carrera):
        print("La carrera no existe en la base de datos.")

    semestre = formtatear_semestre(input("Semestre (01 - 10): "))
    if not semestre:
        print("Semestre inválido. Debe ser entre 01 y 10.")
        return

    materias_disponibles = obtener_materias(conn)
    materias_ingresada = input("Clave de materias a agregar (separadas
por coma): ")
    materias_validas = [m for m in materias_ingresada.split(",") if
m.strip() in materias_disponibles]

    if not materias_validas:
        print("Ninguna materia valida seleccionada.")
        return

    fecha_alta = formatear_fecha(input("Fecha de alta (YYYY-MM-DD o
vacío): ") or "")
    fecha_baja = formatear_fecha(input("Fecha de baja (YYYY-MM-DD o
vacío): ") or "")

    query_insert = """
        INSERT INTO planes (carrer, materi, semest, fecalt, fecbaj)
        VALUES (%s, %s, %s, %s, %s)
    """

    try:
        with conn.cursor() as cursor:
            for materia in materias_validas:
                cursor.execute(query_insert, (clave_carrera,
materia.strip(), semestre, fecha_alta, fecha_baja))

```

```

        conn.commit()

        print(f"Se agregaron {len(materias_validas)} materias al plan
de estudios de la carrera {clave_carrera}")
    except Error as err:
        conn.rollback()
        print(f"Error al insertar materias al plan de estudios. Error:
{err}")

def eliminar_plan(conn: MySQLConnection):
    """
    Elimina un plan por su ID
    """

    mostrar_planes(conn)

    try:
        id_plan = int(input("ID del plan a eliminar: ").strip())
    except ValueError:
        print("ID Inválido")
        return

    query_eliminar = "DELETE FROM planes WHERE id = %s"
    with conn.cursor() as cursor:
        cursor.execute(query_eliminar, (id_plan, ))
        conn.commit()

        if cursor.rowcount:
            print("Plan eliminado correctamente")
        else:
            print("No se encontró plan con ese ID.")

def editar_plan(conn: MySQLConnection):
    """
    Edita los datos de un plan (semestre y fechas)
    """

    mostrar_planes(conn)

    try:
        id_plan = int(input("ID del plan a editar: ").strip())
    except ValueError:
        print("ID inválido")
        return

```



```

        nuevo_semestre = input("Nuevo semestre (01 - 10 o enter para
mantener): ")

    if nuevo_semestre:
        nuevo_semestre = formtatear_semestre(nuevo_semestre)
        if not nuevo_semestre:
            print("Semestre inválido. Operación cancelada")
            return

        nueva_fecha_alta = formatear_fecha(input("Nueva fecha de alta
(YYYY-MM-DD o vacío): ") or "")
        nueva_fecha_baja = formatear_fecha(input("Nueva fecha de baja
(YYYY-MM-DD o vacío): ") or "")

    query_update = """
        UPDATE planes
        SET semest = COALESCE(%s, semest),
            fecalt = %s,
            fecbaj = %s
        WHERE id = %s
    """

    try:
        with conn.cursor() as cursor:
            cursor.execute(query_update, (nuevo_semestre,
nueva_fecha_alta, nueva_fecha_baja, id_plan))
            conn.commit()

            if cursor.rowcount:
                print("Plan actualizado correctamente.")
            else:
                print("No se encontró ningún plan con ese ID.")
    except Error as err:
        print(f"Error al actualizar: {err}")
        conn.rollback()

# Menú principal
def menu_principal():
    """
    Menú interactivo
    """
    try:

```

```

        conexion = conectar_db()
except Error as err:
    print(f"Error al conectar con la base de datos: {err}")
    sys.exit(1)

try:
    while True:
        print("\n==== MENÚ PRINCIPAL: PLANES DE ESTUDIO ====")
        print("1. Listar todos los planes.")
        print("2. Agregar materias a un plan.")
        print("3. Editar un plan existente.")
        print("4. Eliminar un plan.")
        print("5. Salir")

        opcion = input("selecciona una opción: ").strip()

        match opcion:
            case "1":
                mostrar_planes(conexion)
            case "2":
                agregar_plan(conexion)
            case "3":
                editar_plan(conexion)
            case "4":
                eliminar_plan(conexion)
            case "5":
                print("Saliendo del sistema...")
                break
            case _:
                print("Opción invalida. Intenta de nuevo.")

    finally:
        conexion.close()

if __name__ == "__main__":
    menu_principal()

```

Evidencia de Funcionamiento:

The screenshot shows a Visual Studio Code editor with a Python file named `app.py`. The code defines a `DATABASE_CONFIG` dictionary with the following values: `host: os.getenv("MYSQL_HOST", "localhost")`, `port: int(os.getenv("MYSQL_PORT", 3306))`, `user: os.getenv("MYSQL_USER", "admin")`, `password: os.getenv("MYSQL_PASSWORD", "admin")`, `database: os.getenv("MYSQL_DB", "legacy_escolar")`, and `autocommit: False`. The terminal output shows the command `python db/app.py` being executed, resulting in an error: `Error al conectar con la base de datos: 1045 (28000): Access denied for user 'admin'@'localhost' (using password: YES)`. The interface includes a sidebar with the Explorer, Search, and Run and Debug views, and a bottom status bar showing the file path, encoding, and language.

```
app.py > ...
2 import re
3 import sys
4 from typing import Optional
5 from mysql.connector.connection import MySQLConnection
6 from dotenv import load_dotenv
7 import mysql.connector
8 from mysql.connector import Error
9 from tabulate import tabulate
10
11 load_dotenv()
12
13 DATABASE_CONFIG = {
14     "host": os.getenv("MYSQL_HOST", "localhost"),
15     "port": int(os.getenv("MYSQL_PORT", 3306)),
16     "user": os.getenv("MYSQL_USER", "admin"),
17     "password": os.getenv("MYSQL_PASSWORD", "admin"),
18     "database": os.getenv("MYSQL_DB", "legacy_escolar"),
19     "autocommit": False,
20 }
21
22
```

PS C:\Users\I\Downloads\db> & C:\Users\I\AppData\Local\Microsoft\WindowsApps/python3.11.exe c:\Users\I\Downloads/db/app.py
Error al conectar con la base de datos: 1045 (28000): Access denied for user 'admin'@'localhost' (using password: YES)
PS C:\Users\I\Downloads\db> & C:\Users\I\AppData\Local\Microsoft\WindowsApps/python3.11.exe c:\Users\I\Downloads/db/app.py

==== MENÚ PRINCIPAL: PLANES DE ESTUDIO ====
1. Listar todos los planes.
2. Agregar materias a un plan.
3. Editar un plan existente.
4. Eliminar un plan.
5. Salir

The screenshot shows the same Visual Studio Code editor with the `app.py` file. The terminal output now displays the application's menu and a table of study plans. The menu options are: 1. Listar todos los planes, 2. Agregar materias a un plan, 3. Editar un plan existente, 4. Eliminar un plan, 5. Salir. The table, titled "PLANES DE ESTUDIO", has columns for ID, Carrera, Materia, semestre, Fecha alta, and Fecha baja. The data is as follows:

ID	Carrera	Materia	semestre	Fecha alta	Fecha baja
1	05	101	01	1982-06-24	
2	05	102	01	1982-06-25	
3	05	103	01	1982-06-24	
4	05	201	02	1982-06-28	
5	05	202	02	1982-06-28	
6	05	203	02	1982-06-28	
7	07	105	01	1982-06-28	
8	07	106	01	1982-06-28	
9	07	204	02	1982-06-28	
10	07	206	02	1982-06-28	
11	07	207	02	1982-06-28	
12	07	208	02	1982-06-28	

The application is currently displaying the menu options again, with the prompt "selecciona una opción: 1".

```
app.py > ...
2 import re
3 import sys
4 from typing import Optional
5 from mysql.connector.connection import MySQLConnection
6 from dotenv import load_dotenv
7 import mysql.connector
8 from mysql.connector import Error
9 from tabulate import tabulate
10
11 load_dotenv()
12
13 DATABASE_CONFIG = {
14     "host": os.getenv("MYSQL_HOST", "localhost"),
15     "port": int(os.getenv("MYSQL_PORT", 3306)),
16     "user": os.getenv("MYSQL_USER", "admin"),
17     "password": os.getenv("MYSQL_PASSWORD", "admin"),
18     "database": os.getenv("MYSQL_DB", "legacy_escolar"),
19     "autocommit": False,
20 }
21
22
```

PS C:\Users\I\Downloads\db> & C:\Users\I\AppData\Local\Microsoft\WindowsApps/python3.11.exe c:\Users\I\Downloads/db/app.py

==== MENÚ PRINCIPAL: PLANES DE ESTUDIO ====
1. Listar todos los planes.
2. Agregar materias a un plan.
3. Editar un plan existente.
4. Eliminar un plan.
5. Salir
selecciona una opción: 1


==== PLANES DE ESTUDIO ====

ID	Carrera	Materia	semestre	Fecha alta	Fecha baja
1	05	101	01	1982-06-24	
2	05	102	01	1982-06-25	
3	05	103	01	1982-06-24	
4	05	201	02	1982-06-28	
5	05	202	02	1982-06-28	
6	05	203	02	1982-06-28	
7	07	105	01	1982-06-28	
8	07	106	01	1982-06-28	
9	07	204	02	1982-06-28	
10	07	206	02	1982-06-28	
11	07	207	02	1982-06-28	
12	07	208	02	1982-06-28	

==== MENÚ PRINCIPAL: PLANES DE ESTUDIO ====
1. Listar todos los planes.
2. Agregar materias a un plan.
3. Editar un plan existente.
4. Eliminar un plan.
5. Salir
selecciona una opción: 1

Diseño de la base de datos

materias	
clave	INT * pk
descri	VARCHAR(120) *
nsesio	TINYINT
durses	DECIMAL(4,1)
taller	BOOLEAN
fecalt	DATE
fecbaj	DATE
tipo	CHAR(1)

planes	
id	INT *  pk
carrer	CHAR(2) *
materi	INT *
semest	CHAR(2) *
fecalt	DATE
fecbaj	DATE

carreras	
clave	CHAR(2) * pk
nombre	VARCHAR(120) *
fecalt	DATE
fecbaj	DATE