

**Trabajo de desarrollo**

**Lm,mlm,m,m,m,m,m,m,m,m,m,m,m,m,m,**

# Tabla de contenidos

Contenido

[Indice 1](#_Toc166968623)

[Introducción. 1](#_Toc166968624)

[Desarrollo 2](#_Toc166968625)

[Modelo de base de datos. 2](#_Toc166968626)

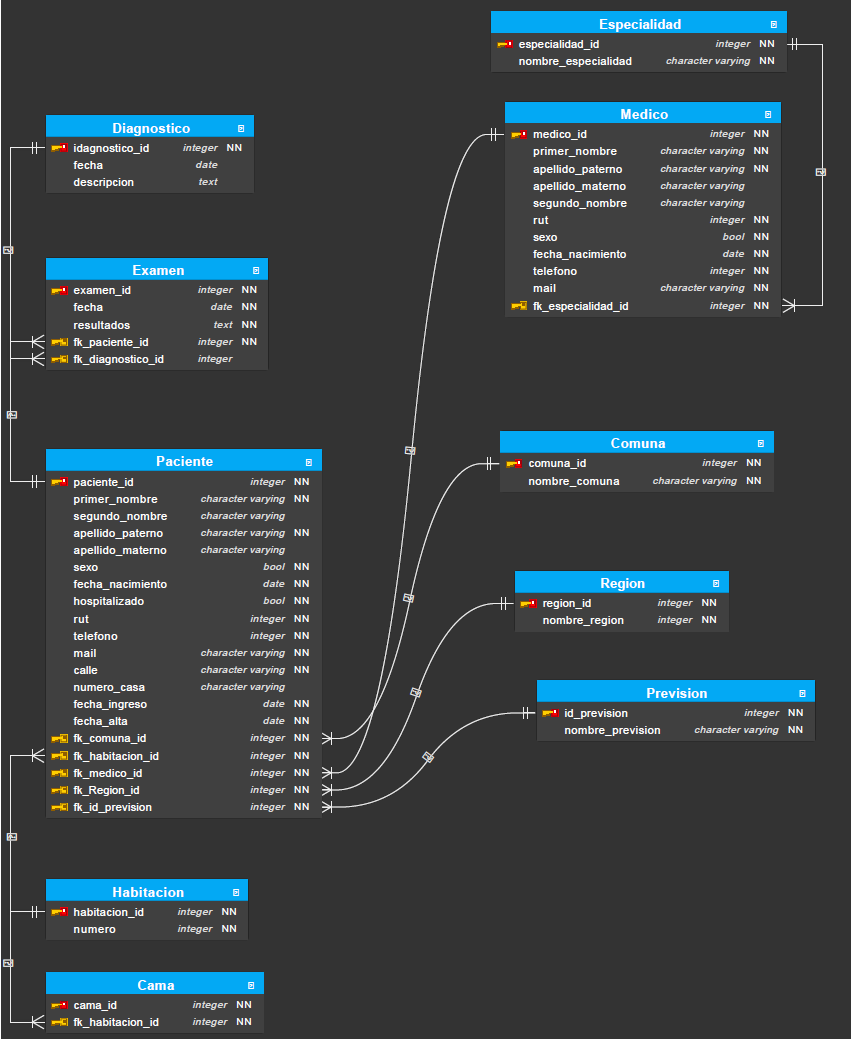
# Introducción.

En el contexto de una clínica médica, se requiere desarrollar una aplicación de consola que permita gestionar la información relacionada con pacientes, camas, médicos, exámenes y diagnósticos. La aplicación se implementará utilizando el lenguaje de programación Python y se integrará con una base de datos PostgreSQL para almacenar y recuperar los datos.

El objetivo principal de esta aplicación es proporcionar una herramienta eficiente y fácil de usar para el personal médico y administrativo de la clínica, permitiéndoles llevar un registro preciso de los pacientes, sus asignaciones de camas, los médicos responsables y los diagnósticos realizados.

# Desarrollo

## Modelo de base de datos.



## SQL

|  |
| --- |
| CREATE TABLE "Cama"(cama\_id integer NOT NULL GENERATED BY DEFAULT AS IDENTITY, fk\_habitacion\_id integer NOT NULL, CONSTRAINT "Cama\_pkey" PRIMARY KEY(cama\_id));  CREATE TABLE "Comuna"(comuna\_id integer NOT NULL GENERATED BY DEFAULT AS IDENTITY, nombre\_comuna character varying NOT NULL, CONSTRAINT "Comuna\_pkey" PRIMARY KEY(comuna\_id));  CREATE TABLE "Diagnostico"(  idagnostico\_id integer NOT NULL GENERATED BY DEFAULT AS IDENTITY,  fecha date,  descripcion text,  CONSTRAINT "Diagnostico\_key" UNIQUE(idagnostico\_id)  );  CREATE TABLE "Especialidad"(especialidad\_id integer NOT NULL GENERATED BY DEFAULT AS IDENTITY, nombre\_especialidad character varying NOT NULL, CONSTRAINT "Especialidad\_pkey" PRIMARY KEY(especialidad\_id));  CREATE TABLE "Examen"(  examen\_id integer NOT NULL GENERATED BY DEFAULT AS IDENTITY,  fecha date NOT NULL,  resultados text NOT NULL,  fk\_paciente\_id integer NOT NULL,  fk\_diagnostico\_id integer,  CONSTRAINT "Examen\_pkey" PRIMARY KEY(examen\_id)  );  CREATE TABLE "Habitacion"(habitacion\_id integer NOT NULL GENERATED BY DEFAULT AS IDENTITY, numero integer NOT NULL, CONSTRAINT "Habitacion\_pkey" PRIMARY KEY(habitacion\_id));  CREATE TABLE "Medico"(  medico\_id integer NOT NULL GENERATED BY DEFAULT AS IDENTITY,  primer\_nombre character varying NOT NULL,  apellido\_paterno character varying NOT NULL,  apellido\_materno character varying,  segundo\_nombre character varying,  rut integer NOT NULL,  sexo bool NOT NULL,  fecha\_nacimiento date NOT NULL,  telefono integer NOT NULL,  mail character varying NOT NULL,  fk\_especialidad\_id integer NOT NULL,  CONSTRAINT "Medico\_pkey" PRIMARY KEY(medico\_id)  );  CREATE TABLE "Paciente"(  paciente\_id integer NOT NULL GENERATED BY DEFAULT AS IDENTITY,  primer\_nombre character varying NOT NULL,  segundo\_nombre character varying,  apellido\_paterno character varying NOT NULL,  apellido\_materno character varying,  sexo bool NOT NULL,  fecha\_nacimiento date NOT NULL,  hospitalizado bool NOT NULL,  rut integer NOT NULL,  telefono integer NOT NULL,  mail character varying NOT NULL,  calle character varying NOT NULL,  numero\_casa character varying,  fecha\_ingreso date NOT NULL,  fecha\_alta date NOT NULL,  fk\_comuna\_id integer NOT NULL,  fk\_habitacion\_id integer NOT NULL,  fk\_medico\_id integer NOT NULL,  "fk\_Region\_id" integer NOT NULL,  fk\_id\_prevision integer NOT NULL,  CONSTRAINT "Paciente\_pkey" PRIMARY KEY(paciente\_id)  );  CREATE TABLE "Prevision"(id\_prevision integer NOT NULL GENERATED BY DEFAULT AS IDENTITY, nombre\_prevision character varying NOT NULL, CONSTRAINT "Prevision\_pkey" PRIMARY KEY(id\_prevision));  CREATE TABLE "Region"(region\_id integer NOT NULL GENERATED BY DEFAULT AS IDENTITY, nombre\_region integer NOT NULL, CONSTRAINT "Region\_pkey" PRIMARY KEY(region\_id));  ALTER TABLE "Paciente" ADD CONSTRAINT "Paciente\_fk\_comuna\_id\_fkey" FOREIGN KEY (fk\_comuna\_id) REFERENCES "Comuna" (comuna\_id);  ALTER TABLE "Paciente" ADD CONSTRAINT "Paciente\_fk\_habitacion\_id\_fkey" FOREIGN KEY (fk\_habitacion\_id) REFERENCES "Habitacion" (habitacion\_id);  ALTER TABLE "Cama" ADD CONSTRAINT "Cama\_fk\_habitacion\_id\_fkey" FOREIGN KEY (fk\_habitacion\_id) REFERENCES "Habitacion" (habitacion\_id);  ALTER TABLE "Examen" ADD CONSTRAINT "Examen\_fk\_paciente\_id\_fkey" FOREIGN KEY (fk\_paciente\_id) REFERENCES "Paciente" (paciente\_id);  ALTER TABLE "Paciente" ADD CONSTRAINT "Paciente\_fk\_medico\_id\_fkey" FOREIGN KEY (fk\_medico\_id) REFERENCES "Medico" (medico\_id);  ALTER TABLE "Paciente" ADD CONSTRAINT "Paciente\_fk\_Region\_id\_fkey" FOREIGN KEY ("fk\_Region\_id") REFERENCES "Region" (region\_id);  ALTER TABLE "Paciente" ADD CONSTRAINT "Paciente\_fk\_id\_prevision\_fkey" FOREIGN KEY (fk\_id\_prevision) REFERENCES "Prevision" (id\_prevision);  ALTER TABLE "Medico" ADD CONSTRAINT "Medico\_fk\_especialidad\_id\_fkey" FOREIGN KEY (fk\_especialidad\_id) REFERENCES "Especialidad" (especialidad\_id);  ALTER TABLE "Examen" ADD CONSTRAINT "Examen\_fk\_diagnostico\_id\_fkey" FOREIGN KEY (fk\_diagnostico\_id) REFERENCES "Diagnostico" (idagnostico\_id); |

## Python.

## Main.py

|  |
| --- |
| from funciones import lista\_pacientes, MostrarPacienteViaRut, CambioDeCama, CambioDeMedico,insertar\_cama,agregar\_habitaciones  while True:      print("\nMenú:")      print("1. Lista de pacientes")      print("2. Mostrar paciente por RUT")      print("3. Cambiar paciente de habitación")      print("4. Cambiar paciente de médico")      print("5. Insertar cama")      print("6. Agregar habitaciones")      print("7. Salir del programa")      opcion = input("Elige una opción (1-7): ")      if opcion == "1":          lista\_pacientes()      elif opcion == "2":          MostrarPacienteViaRut()      elif opcion == "3":          CambioDeCama()      elif opcion == "4":          CambioDeMedico()      elif opcion =="5":          insertar\_cama()      elif opcion=="6":          agregar\_habitaciones()      elif opcion == "7":          print("¡Hasta luego!")          break      else:          print("Opción inválida. Elige una opción del 1 al 5.") |

Funciones.py

|  |
| --- |
| import psycopg2  def lista\_pacientes():      # Parámetros de conexión a la base de datos      db\_params = {          "host": "localhost",          "database": "Clinica",          "user": "postgres",          "password": "adobe16"      }      try:          # Conectarse a la base de datos          conn = psycopg2.connect(\*\*db\_params)          print("Conexión exitosa a la base de datos.")          # Crear un cursor para ejecutar la consulta          cursor = conn.cursor()          # Ejecutar la consulta SQL          cursor.execute('SELECT \              pa.rut, pa.primer\_nombre || \' \' || pa.apellido\_paterno AS nombre\_paciente, \              di.descripcion AS diagnostico, \              m.primer\_nombre || \' \' || m.apellido\_paterno AS nombre\_medico, \              ha.numero AS numero\_habitacion \          FROM \              public."Paciente" pa \          INNER JOIN \              public."Diagnostico" di ON pa.fk\_id\_diagnostico = di.idagnostico\_id \          INNER JOIN \              public."Medico" m ON pa.fk\_medico\_id = m.medico\_id \          INNER JOIN \              public."Habitacion" ha ON pa.fk\_habitacion\_id = ha.habitacion\_id')          # Mostrar los resultados          print("A continuación se encuentra la lista de pacientes:")          print("RUT     |   Nombre paciente   | Diagnóstico | Medico | Habitación")          for rut, nombre\_paciente, diagnostico, nombre\_medico, numero\_habitacion in cursor.fetchall():              print(rut, "|", nombre\_paciente, "|", diagnostico, "|", nombre\_medico, "|", numero\_habitacion)      except (Exception, psycopg2.DatabaseError) as error:          print("Error al conectar o ejecutar la consulta:", error)      finally:          # Cerrar la conexión          cursor.close()          conn.close()    def MostrarPacienteViaRut():      while True:          # Solicitar el RUT del paciente          rut\_paciente = input("Por favor, ingresa el RUT del paciente: ")          # Parámetros de conexión a la base de datos          db\_params = {              "host": "localhost",              "database": "Clinica",              "user": "postgres",              "password": "adobe16"          }          try:              # Conectarse a la base de datos              conn = psycopg2.connect(\*\*db\_params)              print("Conexión exitosa a la base de datos.")              # Crear un cursor para ejecutar la consulta              cursor = conn.cursor()              # Ejecutar la consulta SQL              cursor.execute("""                  SELECT                      pa.rut,                      pa.primer\_nombre || ' ' || pa.apellido\_paterno AS nombre\_paciente,                      di.descripcion AS diagnostico,                      me.primer\_nombre || ' ' || me.apellido\_paterno AS nombre\_medico,                      ex.fecha AS fecha\_ultimo\_examen,                      ha.numero AS numero\_habitacion                  FROM                      public."Paciente" pa                  INNER JOIN                      public."Diagnostico" di ON pa.fk\_id\_diagnostico = di.idagnostico\_id                  INNER JOIN                      public."Medico" me ON pa.fk\_medico\_id = me.medico\_id                  INNER JOIN                      public."Examen" ex ON pa.paciente\_id = ex.fk\_paciente\_id                  INNER JOIN                      public."Habitacion" ha ON pa.fk\_habitacion\_id = ha.habitacion\_id                  WHERE                      pa.rut = %s              """, (rut\_paciente,))              # Obtener los resultados              records = cursor.fetchall()              # Imprimir los resultados              for row in records:                  print(row)              # Salir del bucle si se encontraron resultados              if records:                  break              else:                  print("No se encontró ningún paciente con ese RUT. Inténtalo nuevamente.")          except (Exception, psycopg2.DatabaseError) as error:              print("Error al conectar o ejecutar la consulta:", error)          finally:              # Cerrar la conexión              cursor.close()              conn.close()    def CambioDeCama():      # Parámetros de conexión a la base de datos      db\_params = {          "host": "localhost",          "database": "Clinica",          "user": "postgres",          "password": "adobe16"      }      try:          # Conectarse a la base de datos          conn = psycopg2.connect(\*\*db\_params)          print("Conexión exitosa a la base de datos.")          # Crear un cursor para ejecutar las consultas          cursor = conn.cursor()          while True:              # Solicitar el RUT del paciente y el nuevo número de habitación              rut\_paciente = input("Ingresa el RUT del paciente: ")              numero\_habitacion = input("Ingresa el nuevo número de habitación: ")              # Verificar si el paciente existe en la base de datos              cursor.execute("SELECT 1 FROM public.\"Paciente\" WHERE rut = %s;", (rut\_paciente,))              if cursor.fetchone():                  # Ejecutar la consulta SQL de actualización                  cursor.execute("UPDATE public.\"Paciente\" SET fk\_habitacion\_id = %s WHERE rut = %s;", (numero\_habitacion, rut\_paciente))                  conn.commit()                  print(f"Se ha cambiado la habitación del paciente con RUT {rut\_paciente} al número {numero\_habitacion}.")                  break              else:                  print(f"No se encontró al paciente con RUT {rut\_paciente}. Ingresa un RUT válido.")      except (Exception, psycopg2.DatabaseError) as error:          print("Error al conectar o ejecutar la consulta:", error)      finally:          # Cerrar la conexión          cursor.close()          conn.close()    def CambioDeMedico():      # Solicitar la ID del nuevo médico y el RUT del paciente      print("--Para reasignar medico indique Rut depaciente y ID de médico--")      rut\_paciente = input("Por favor, ingresa el RUT del paciente: ")      nuevo\_medico\_id = input("Por favor, ingresa la ID del nuevo médico: ")        # Parámetros de conexión a la base de datos      db\_params = {          "host": "localhost",          "database": "Clinica",          "user": "postgres",          "password": "adobe16"      }      try:          # Conectarse a la base de datos          conn = psycopg2.connect(\*\*db\_params)          print("Conexión exitosa a la base de datos.")          # Crear un cursor para ejecutar la consulta          cursor = conn.cursor()          # Ejecutar la consulta SQL de actualización          cursor.execute("""              UPDATE                  public."Paciente"              SET                  fk\_medico\_id = %s              WHERE                  rut = %s;          """, (nuevo\_medico\_id, rut\_paciente))          # Confirmar los cambios en la base de datos          conn.commit()          # Verificar si la actualización tuvo éxito          if cursor.rowcount == 0:              print("No se encontró el paciente con el RUT proporcionado o el médico ya tiene asignado ese RUT.")          else:              print(f"El médico con ID {nuevo\_medico\_id} ha sido asignado al paciente con RUT {rut\_paciente}.")      except (Exception, psycopg2.DatabaseError) as error:          print("Error al conectar o ejecutar la consulta:", error)      finally:          # Cerrar la conexión          cursor.close()          conn.close()    def insertar\_cama():      # Parámetros de conexión a la base de datos      db\_params = {          "host": "localhost",          "database": "Clinica",          "user": "postgres",          "password": "adobe16"      }      try:          # Conectarse a la base de datos          conn = psycopg2.connect(\*\*db\_params)          print("Conexión exitosa a la base de datos.")          # Crear un cursor para ejecutar la consulta          cursor = conn.cursor()          # Imprimir las camas actuales          cursor.execute("SELECT \* FROM public.\"Cama\";")          camas\_actuales = cursor.fetchall()          print("Camas actuales:")          for cama in camas\_actuales:              print(cama)          while True:              # Solicitar el valor por teclado              fk\_habitacion\_id = input("Ingresa el ID de la habitación: ")              # Verificar si ya existe el valor en la tabla              cursor.execute("SELECT 1 FROM public.\"Cama\" WHERE fk\_habitacion\_id = %s;", (fk\_habitacion\_id,))              if cursor.fetchone():                  print(f"El valor {fk\_habitacion\_id} ya existe en la base de datos. Ingresa otro valor.")              else:                  # Ejecutar la consulta SQL de inserción                  cursor.execute("INSERT INTO public.\"Cama\" (fk\_habitacion\_id) VALUES (%s);", (fk\_habitacion\_id,))                  conn.commit()                  print(f"Se ha insertado el valor {fk\_habitacion\_id} correctamente.")                  break      except (Exception, psycopg2.DatabaseError) as error:          print("Error al conectar o ejecutar la consulta:", error)      finally:          # Cerrar la conexión          cursor.close()          conn.close()    def agregar\_habitaciones():      # Parámetros de conexión a la base de datos      db\_params = {          "host": "localhost",          "database": "Clinica",          "user": "postgres",          "password": "adobe16"      }      try:          # Conectarse a la base de datos          conn = psycopg2.connect(\*\*db\_params)          print("Conexión exitosa a la base de datos.")          # Crear un cursor para ejecutar la consulta          cursor = conn.cursor()          # Imprimir las habitaciones actuales          cursor.execute("SELECT \* FROM public.\"Habitacion\";")          habitaciones\_actuales = cursor.fetchall()          print("Habitaciones actuales:")          for habitacion in habitaciones\_actuales:              print(habitacion)          while True:              # Solicitar el valor por teclado              numero\_habitacion = input("Ingresa el número de la habitación: ")              # Verificar si ya existe el valor en la tabla              cursor.execute("SELECT 1 FROM public.\"Habitacion\" WHERE numero = %s;", (numero\_habitacion,))              if cursor.fetchone():                  print(f"El número {numero\_habitacion} ya existe en la base de datos. Ingresa otro valor.")              else:                  # Ejecutar la consulta SQL de inserción                  cursor.execute("INSERT INTO public.\"Habitacion\" (numero) VALUES (%s);", (numero\_habitacion,))                  conn.commit()                  print(f"Se ha insertado el número {numero\_habitacion} correctamente.")                  break      except (Exception, psycopg2.DatabaseError) as error:          print("Error al conectar o ejecutar la consulta:", error)      finally:          # Cerrar la conexión          cursor.close()          conn.close() |

# Conclusiones.

Este proyecto proporcionó una experiencia práctica en el desarrollo de aplicaciones que requieren interacción con bases de datos, lo que es fundamental para cualquier desarrollador de software.

La combinación de Python y PostgreSQL demostró ser una solución eficiente y escalable para las necesidades de gestión clínica. Continuar explorando y aplicando estos conocimientos en proyectos futuros será esencial para seguir mejorando las habilidades en el campo de la programación y la administración de bases de datos.