Universidad de La Laguna

ADBD.

Proyecto

Proyecto de la asignatura.

VÍCTOR RODRÍGUEZ DORTA
alu0101540153
Daniel Marhuenda Guillén
alu0101487731
MARIO GUERRA PÉREZ
alu0101395036

Índice.

Introducción.

En el presente proyecto se desarrolla un sistema de gestión para la clínica dental Sonrisas, con el objetivo de centralizar y optimizar tanto la gestión administrativa como la atención clínica de los pacientes. La necesidad de un sistema integral surge de la complejidad de los procesos diarios en la clínica, que incluyen la programación de citas, el seguimiento de los historiales clínicos, la administración del personal sanitario, el control de materiales y el seguimiento de encargos a laboratorios externos.

El sistema se ha diseñado considerando la importancia del paciente como entidad central, así como la correcta gestión de recursos humanos y materiales. Se contemplan distintas modalidades de financiación, relaciones jerárquicas entre el personal sanitario y la gestión de intervenciones y tratamientos complejos, incluyendo aquellos que requieren la intervención de laboratorios externos. El proyecto integra conceptos fundamentales de modelado de bases de datos como entidades fuertes y débiles, relaciones 1:N y N:M, relaciones IS_A exclusivas, y restricciones de exclusividad e integridad, asegurando así un diseño robusto y coherente.

Este trabajo pretende cubrir de manera completa todo el ciclo de diseño e implementación de una base de datos: desde la especificación de requisitos hasta la implementación en PostgreSQL, pasando por la carga de datos de ejemplo y el desarrollo de un API REST en Flask para la gestión de operaciones CRUD sobre los datos clínicos y administrativos.

Los objetivos del proyecto se pueden resumir en los siguientes puntos:

- Diseñar y modelar una base de datos relacional que refleje de manera fiel los procesos y necesidades de la clínica dental Sonrisas, contemplando todas las entidades y relaciones relevantes, incluyendo casos de exclusividad, jerarquías IS_A, relaciones N:M y triples.

- Implementar la base de datos en PostgreSQL, definiendo claves primarias, foráneas, dominios y restricciones de integridad, así como disparadores y checks que garanticen la consistencia de la información.

- Realizar una carga de datos de ejemplo que permita probar la funcionalidad del sistema y la correcta interacción entre las distintas entidades.

- Desarrollar un API REST con Flask que permita la gestión completa de los datos de la clínica, ofreciendo endpoints para operaciones CRUD sobre pacientes, citas, intervenciones, tratamientos, personal sanitario, materiales y encargos externos.

- Documentar y justificar el diseño mediante diagramas entidad-relación y relacionales, incluyendo supuestos semánticos, decisiones de modelado y ejemplos de consultas de prueba.

- Proporcionar un sistema funcional y escalable, capaz de soportar la evolución de la clínica y la incorporación de nuevos requisitos en el futuro, garantizando la integridad y consistencia de los datos en todas las operaciones

Descripción del contexto.

La clínica dental *Sonrisas* necesita un sistema centralizado que permita gestionar toda su actividad, tanto la relacionada con la atención a los pacientes como la parte administrativa. Con este sistema se pretende llevar el control de los expedientes clínicos de los pacientes, organizar la agenda diaria, administrar al personal y los materiales utilizados en los procedimientos, y además supervisar los trabajos que se encargan a servicios externos.

El elemento principal del sistema son los pacientes, ya que se almacena información tanto personal como sanitaria de cada uno. La clínica trabaja bajo dos formas de financiación que no pueden coincidir: un paciente puede ser atendido como cliente privado o como beneficiario de una mutua aseguradora, pero nunca bajo ambas modalidades al mismo tiempo en un mismo expediente. Cuando el paciente pertenece a una mutua, se debe registrar la compañía aseguradora y el número de póliza asociado, y los tratamientos disponibles quedan limitados a los que cubre dicha aseguradora.

La organización del trabajo diario se basa en citas reservadas en la agenda. Cada cita es un hueco asignado para la atención y no tiene sentido si no está asociada a un paciente concreto. Durante una misma cita, el personal puede añadir varias observaciones clínicas que queden registradas como parte del historial del paciente para facilitar su seguimiento en visitas posteriores.

El personal que trabaja en la clínica está compuesto por distintos profesionales sanitarios. Aunque todos comparten información administrativa común como identificación, nombre y datos de contacto, es necesario diferenciarlos por el rol que desempeñan, ya que sus funciones y competencias no son iguales. El equipo se clasifica de forma excluyente en tres grupos: dentistas, que cuentan con número de colegiado, higienistas y auxiliares.

Los servicios que ofrece la clínica se estructuran como tratamientos, que abarcan desde limpiezas hasta intervenciones más complejas como endodoncias o extracciones. Cuando se lleva a cabo un procedimiento, se registra como una intervención. Para que una intervención sea considerada válida, debe quedar reflejado a quién se atiende, qué profesional la realiza y en qué sala se lleva a cabo, además de la fecha y la hora exacta en la que se ejecuta.

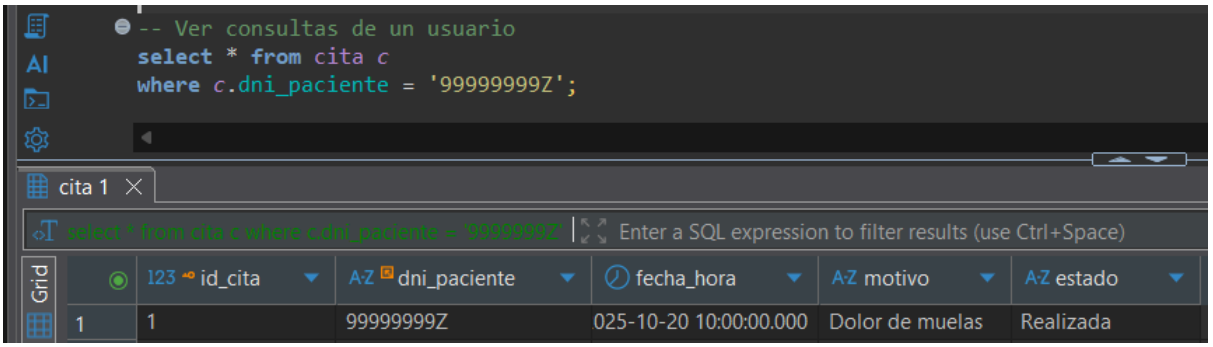
Una intervención puede incluir uno o varios tratamientos al mismo tiempo, dependiendo del caso clínico. En la realización de estos tratamientos se consumen distintos materiales desechables, como anestesia, guantes o resinas, que deben descontarse del inventario disponible en la clínica. Para cada intervención se debe anotar la cantidad exacta de material utilizado para llevar un control preciso de existencias.

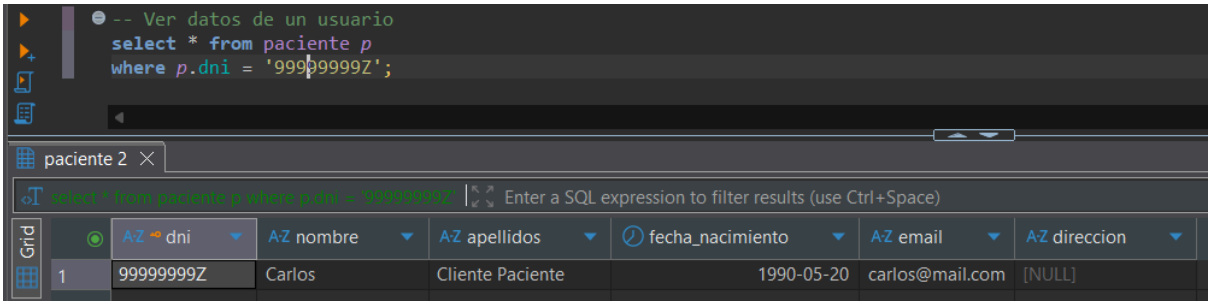
En determinados procedimientos más avanzados, como la colocación de coronas o puentes, la clínica necesita piezas que no fabrica internamente. En esos casos, se solicitan a laboratorios externos. El sistema debe gestionar estos encargos indicando a qué tratamiento corresponden y qué laboratorio los suministra. Además, debe registrarse la fecha en la que se envía el encargo, la fecha prevista de recepción y el coste facturado por el proveedor, de forma que se puedan controlar los gastos relacionados con trabajos externos.

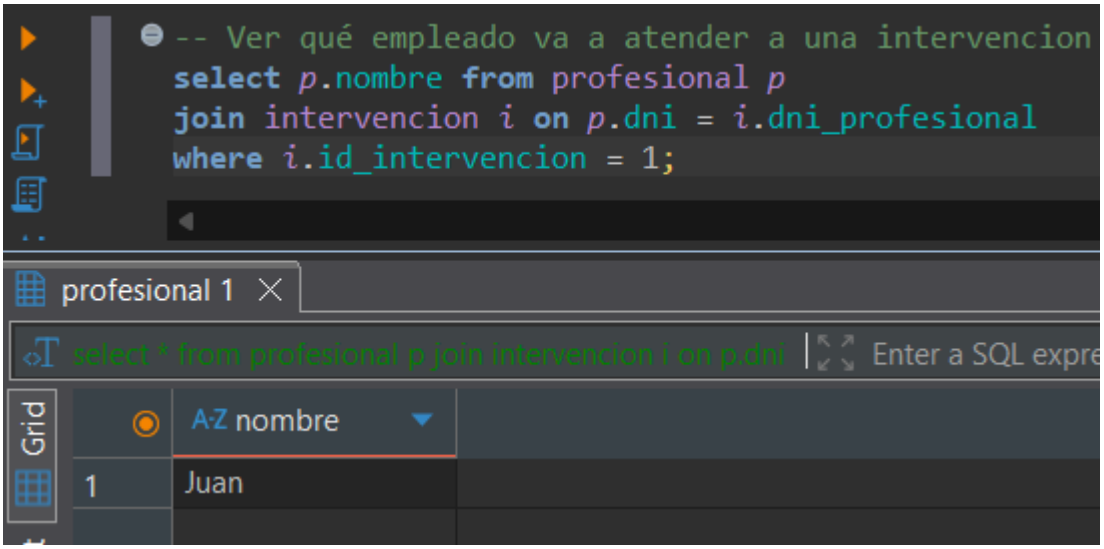
Diagramas.

Los diagramas se encuentran en el repositorio de Github, pero igualmente estos son los enlaces.
Diagrama ER:
https://drive.google.com/file/d/1ajMzULCu_v6VS9D3a5HMGU-tpN0sl-AD/view?usp=sharing
Diagrama Relacional:
https://drive.google.com/file/d/1aSB6zYlz5yzVTRNWpqmpxdRc4pRhtfMx/view?usp=sharing

Consultas de ejemplo.







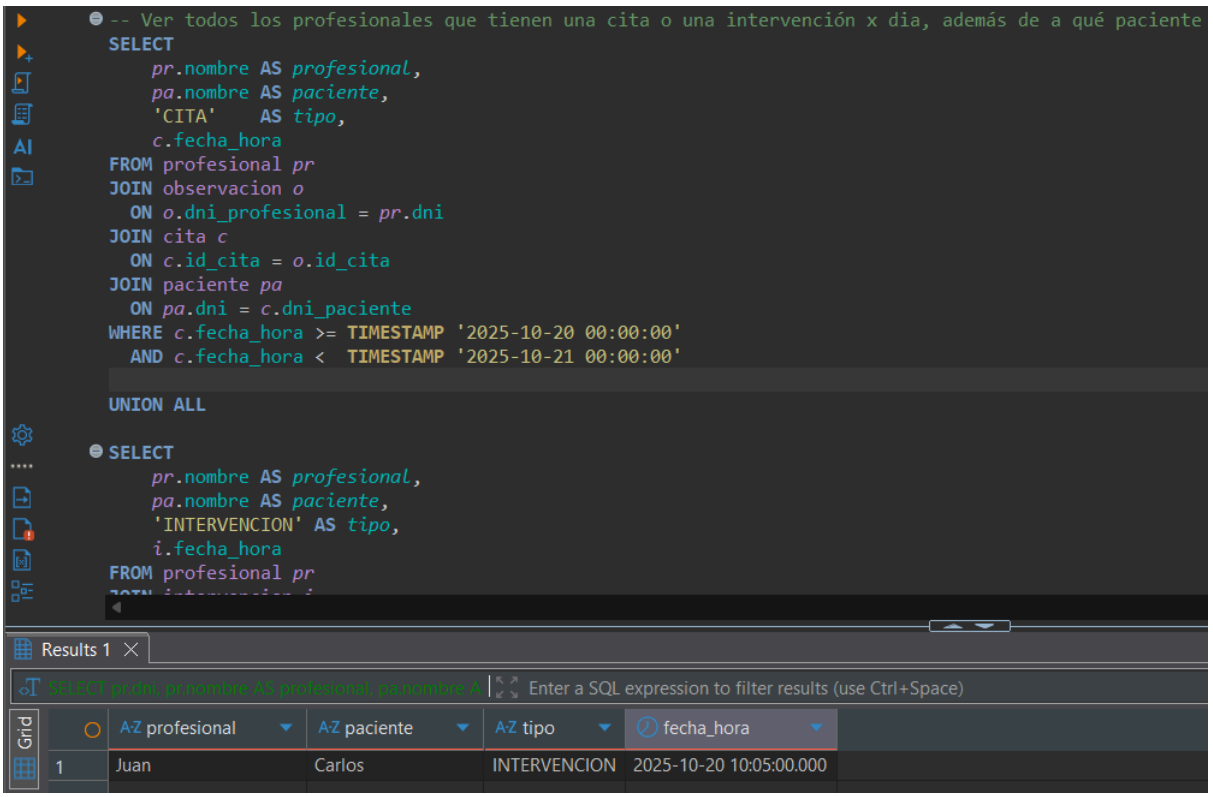
```sql
-- Ver todos los pacientes que tienen una cita o una intervención x dia
SELECT
    p.dni,
    p.nombre,
    'CITA' AS tipo,
    c.fecha_hora
FROM paciente p
JOIN cita c
  ON c.dni_paciente = p.dni
WHERE c.fecha_hora >= TIMESTAMP '2025-10-20 00:00:00'
  AND c.fecha_hora <  TIMESTAMP '2025-10-21 00:00:00'

UNION ALL

SELECT
    p.dni,
    p.nombre,
    'INTERVENCION' AS tipo,
    i.fecha_hora
FROM paciente p
JOIN intervencion i
  ON i.dni_paciente = p.dni
WHERE i.fecha_hora >= TIMESTAMP '2025-10-20 00:00:00'
  AND i.fecha_hora <  TIMESTAMP '2025-10-21 00:00:00';
```

Results 1

SELECT p.dni, p.nombre, 'CITA' AS tipo, c.fecha_hora FR | Enter a SQL expression to filter results (use Ct

Grid

	dni	nombre	tipo	fecha_hora
1	99999999Z	Carlos	CITA	2025-10-20 10:00:00.000
2	99999999Z	Carlos	INTERVENCION	2025-10-20 10:05:00.000



```sql
-- Ver todos los profesionales que tienen una cita o una intervención x dia, además de a qué paciente
SELECT
    pr.nombre AS profesional,
    pa.nombre AS paciente,
    'CITA'    AS tipo,
    c.fecha_hora
FROM profesional pr
JOIN observacion o
  ON o.dni_profesional = pr.dni
JOIN cita c
  ON c.id_cita = o.id_cita
JOIN paciente pa
  ON pa.dni = c.dni_paciente
WHERE c.fecha_hora >= TIMESTAMP '2025-10-20 00:00:00'
  AND c.fecha_hora <  TIMESTAMP '2025-10-21 00:00:00'

UNION ALL

SELECT
    pr.nombre AS profesional,
    pa.nombre AS paciente,
    'INTERVENCION' AS tipo,
    i.fecha_hora
FROM profesional pr
```

Results 1

SELECT pr.dni, pr.nombre AS profesional, pa.nombre AS | Enter a SQL expression to filter results (use Ctrl+Space)

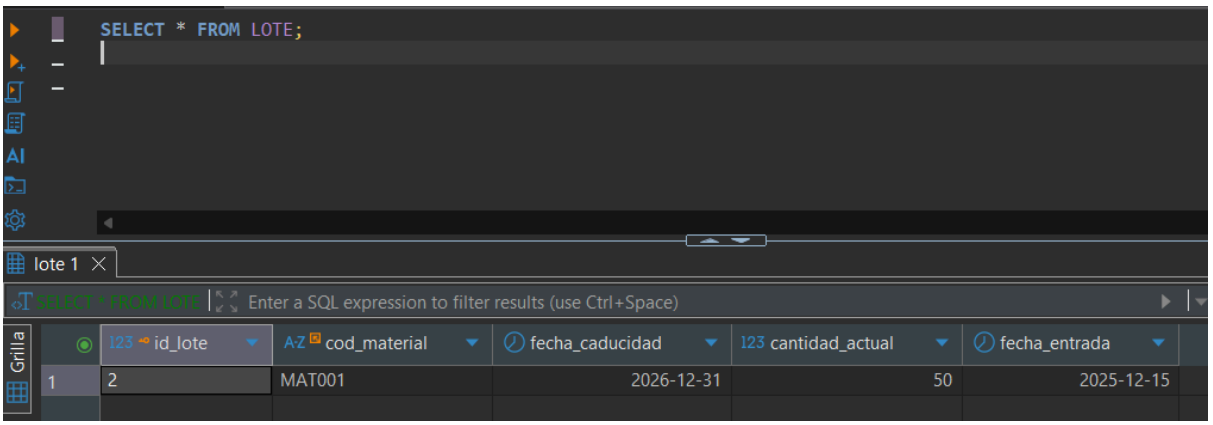
Grid

	profesional	paciente	tipo	fecha_hora
1	Juan	Carlos	INTERVENCION	2025-10-20 10:05:00.000

Trigger.

```sql
CREATE OR REPLACE FUNCTION actualizar_stock_material()
RETURNS TRIGGER AS $$
DECLARE
    stock_actual INT;
    lote_record RECORD;
BEGIN
    SELECT SUM(cantidad_actual) INTO stock_actual
    FROM LOTE
    WHERE cod_material = NEW.cod_material;

    IF stock_actual < NEW.cantidad_material THEN
        RAISE EXCEPTION 'No hay suficiente stock del material %', NEW.cod_material;
    END IF;

    FOR lote_record IN
        SELECT id_lote, cantidad_actual
        FROM LOTE
        WHERE cod_material = NEW.cod_material AND cantidad_actual > 0
        ORDER BY fecha_entrada
    LOOP
        IF lote_record.cantidad_actual >= NEW.cantidad_material THEN
            UPDATE LOTE
            SET cantidad_actual = cantidad_actual - NEW.cantidad_material
            WHERE id_lote = lote_record.id_lote;
            EXIT;
        ELSE
            NEW.cantidad_material := NEW.cantidad_material - lote_record.cantidad_actual;
            UPDATE LOTE
            SET cantidad_actual = 0
            WHERE id_lote = lote_record.id_lote;
        END IF;
    END LOOP;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Este trigger se encarga de controlar y actualizar automáticamente el stock de materiales cuando se registra su uso en una intervención. Antes de realizar el descuento, comprueba que existe cantidad suficiente del material en los lotes disponibles. En caso afirmativo, descuenta la cantidad utilizada siguiendo un criterio FIFO, consumiendo primero los lotes más antiguos. Si no hay stock suficiente, la operación se cancela para garantizar la integridad del inventario.



*<viveros> Script-6

```sql
SELECT cod_material, cantidad_actual
  FROM LOTE
  WHERE cod_material = 'MAT001';
```

lote 1 | lote 2

SELECT cod_material, cantidad_actual FR | Enter a SQL expression to filter results (u

Grilla

	cod_material	cantidad_actual
1	MAT001	45

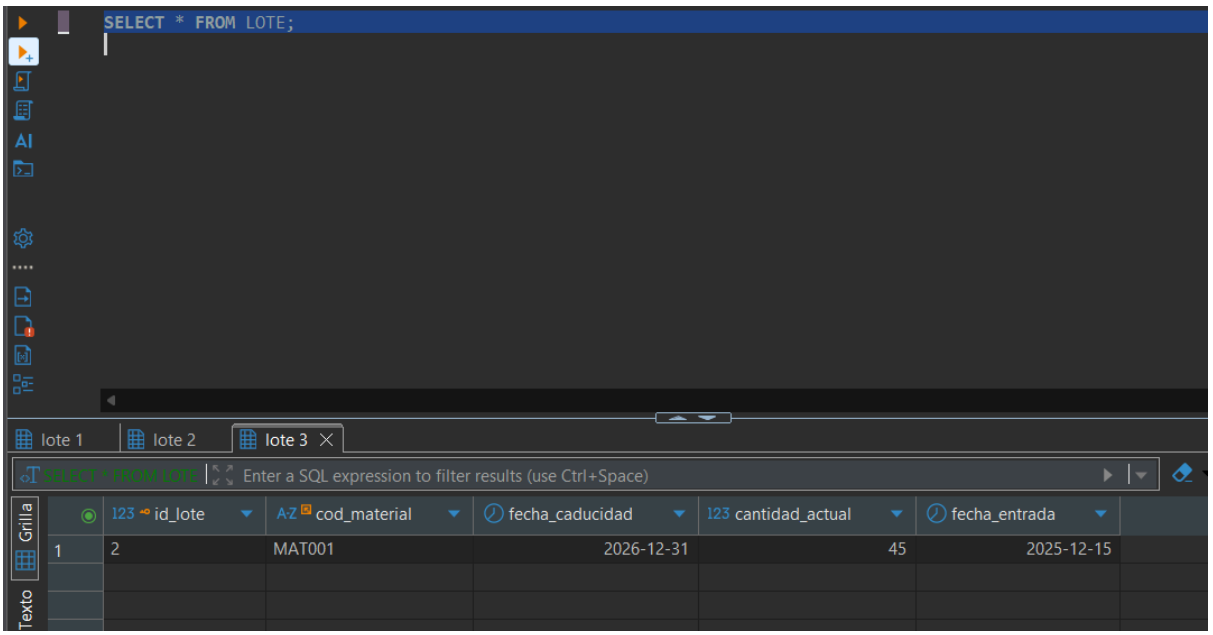
```sql
SELECT
    p.nombre || ' ' || p.apellidos AS paciente,
    t.nombre AS tratamiento,
    m.nombre AS material,
    tim.cantidad_material,
    i.fecha_hora
FROM TRATAMIENTO_INTERVENCION_MATERIAL tim
JOIN TRATAMIENTO t
    ON tim.id_tratamiento = t.id_tratamiento
JOIN MATERIAL m
    ON tim.cod_material = m.cod_material
JOIN INTERVENCION i
    ON tim.id_intervencion = i.id_intervencion
JOIN PACIENTE p
    ON i.dni_paciente = p.dni;
```

lote 1 | tratamiento(+) 2

SELECT p.nombre || ' ' || p.apellidos AS pa | Enter a SQL expression to filter results (use Ctrl+Space)

Grilla

	paciente	tratamiento	material	cantidad_material	fecha_hora
1	Víctor Rodríguez	Limpieza dental	Guantes desechables	5	2025-12-15 19:58:55.758



Peticiones de ejemplo a la API de Flask.

Aquí se muestran capturas de pantalla de la api desarrollada con Flask.
Ejecución de la api.

```
(venv) usuario@ubuntu:~/trabajofinal$ sudo systemctl start postgresql
[sudo] password for usuario:
(venv) usuario@ubuntu:~/trabajofinal$ sudo systemctl status postgresql
● postgresql.service - PostgreSQL RDBMS
     Loaded: loaded (/usr/lib/systemd/system/postgresql.service; enabled; preset: enabled)
     Active: active (exited) since Thu 2025-12-11 10:24:15 UTC; 4 days ago
    Process: 202431 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
   Main PID: 202431 (code=exited, status=0/SUCCESS)
        CPU: 1ms

dic 11 10:24:15 ubuntu systemd[1]: Starting postgresql.service - PostgreSQL RDBMS...
dic 11 10:24:15 ubuntu systemd[1]: Finished postgresql.service - PostgreSQL RDBMS.
(venv) usuario@ubuntu:~/trabajofinal$ python app2.py
 * Serving Flask app 'app2'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://10.6.130.242:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 623-031-122
127.0.0.1 - - [15/Dec/2025 18:44:57] "POST /patients HTTP/1.1" 500 -
127.0.0.1 - - [15/Dec/2025 18:45:24] "POST /patients HTTP/1.1" 201 -
127.0.0.1 - - [15/Dec/2025 18:47:03] "GET /historial/paciente/11111111A HTTP/1.1" 200 -
127.0.0.1 - - [15/Dec/2025 18:47:14] "GET /profesional/33333333C/intervenciones HTTP/1.1" 200 -
```

Post a un paciente.

```
usuario@ubuntu:~$ curl -X POST http://127.0.0.1:5000/patients \
-H "Content-Type: application/json" \
-d '{
  "dni": "22222222B",
  "nombre": "Miguel",
  "apellidos": "Sánchez Ruiz",
  "fecha_nacimiento": "1985-11-03",
  "email": "miguel.sanchez@gmail.com",
  "direccion": "Calle Mayor 18"
}'
{
  "dni": "22222222B",
  "message": "Patient created successfully"
}
```

Error por hacer Post a una consulta con misma clave primaria (DNI)

```
usuario@ubuntu:~$ curl -X POST http://127.0.0.1:5000/patients \
-H "Content-Type: application/json" \
-d '{
  "dni": "33333333D",
  "nombre": "Marta",
  "apellidos": "Sánchez Ruiz",
  "fecha_nacimiento": "1985-09-14",
  "email": "marta@mail.com",
  "direccion": "Avenida Sur 88"
}'
{
  "error": "duplicate key value violates unique constraint \"paciente_pkey\"\nDETAIL:  Key (dni)=(33333333D) already exists.\n"
}
```

Get intervenciones que ha hecho un profesional

```
usuario@ubuntu:~$ curl http://127.0.0.1:5000/profesional/33333333C/intervenciones
[
  {
    "apellidos_paciente": "G\u00f3mez Mart\u00edn",
    "fecha_hora": "Fri, 10 Jan 2025 11:00:00 GMT",
    "id_intervencion": 3,
    "intervencion": "Revisi\u00f3n anual",
    "nombre_paciente": "Laura"
  },
  {
    "apellidos_paciente": "G\u00f3mez Mart\u00edn",
    "fecha_hora": "Fri, 10 Jan 2025 11:00:00 GMT",
    "id_intervencion": 5,
    "intervencion": "Revisi\u00f3n anual",
    "nombre_paciente": "Laura"
  },
  {
    "apellidos_paciente": "G\u00f3mez Mart\u00edn",
    "fecha_hora": "Wed, 20 Nov 2024 10:30:00 GMT",
    "id_intervencion": 4,
    "intervencion": "Extracci\u00f3n muela del juicio",
    "nombre_paciente": "Laura"
  }
]
```

Get Historial de un paciente concreto.

```
usuario@ubuntu:~$ curl http://127.0.0.1:5000/historial/paciente/22222222B
[
  {
    "doctor_apellidos": "L\u00f3pez Aux.",
    "doctor_nombre": "Luis",
    "fecha_hora": "Mon, 15 Dec 2025 19:05:32 GMT",
    "intervencion": "Revisi\u00f3n peri\u00f3dica",
    "tratamiento": "Limpieza Dental"
  },
  {
    "doctor_apellidos": "L\u00f3pez Aux.",
    "doctor_nombre": "Luis",
    "fecha_hora": "Mon, 15 Dec 2025 19:05:27 GMT",
    "intervencion": "Empaste muela",
    "tratamiento": "Limpieza Dental"
  },
  {
    "doctor_apellidos": "L\u00f3pez Aux.",
    "doctor_nombre": "Luis",
    "fecha_hora": "Mon, 15 Dec 2025 19:05:22 GMT",
    "intervencion": "Limpieza dental",
    "tratamiento": "Limpieza Dental"
  },
  {
    "doctor_apellidos": "L\u00f3pez Aux.",
    "doctor_nombre": "Luis",
    "fecha_hora": "Mon, 15 Dec 2025 19:03:45 GMT",
    "intervencion": "Extracci\u00f3n muela",
    "tratamiento": "Limpieza Dental"
  }
]
usuario@ubuntu:~$
```

Get agenda futura de un Profesional concreto.

```
usuario@ubuntu:~$ curl http://127.0.0.1:5000/profesional/33333333C/agenda
[
  {
    "apellidos": "S\u00e1nchez Ruiz",
    "dni_paciente": "22222222B",
    "fecha_hora": "Sat, 20 Dec 2025 10:00:00 GMT",
    "intervencion_programada": "Extracci\u00f3n muela",
    "nombre": "Miguel",
    "sala": "Box 1 - General"
  },
  {
    "apellidos": "S\u00e1nchez Ruiz",
    "dni_paciente": "22222222B",
    "fecha_hora": "Sat, 20 Dec 2025 10:00:00 GMT",
    "intervencion_programada": "Limpieza dental",
    "nombre": "Miguel",
    "sala": "Box 1 - General"
  },
  {
    "apellidos": "S\u00e1nchez Ruiz",
    "dni_paciente": "22222222B",
    "fecha_hora": "Sat, 20 Dec 2025 10:00:00 GMT",
    "intervencion_programada": "Empaste muela",
    "nombre": "Miguel",
    "sala": "Box 1 - General"
  },
  {
    "apellidos": "S\u00e1nchez Ruiz",
    "dni_paciente": "22222222B",
    "fecha_hora": "Sat, 20 Dec 2025 10:00:00 GMT",
    "intervencion_programada": "Revisi\u00f3n peri\u00f3dica",
    "nombre": "Miguel",
    "sala": "Box 1 - General"
  },
  {
    "apellidos": "S\u00e1nchez Ruiz",
```

Put Actualización de un paciente concreto

```
usuario@ubuntu:~$ curl -X PUT http://127.0.0.1:5000/patients/11111111A \
-H "Content-Type: application/json" \
-d '{
  "nombre": "Laura Actualizada",
  "apellidos": "Gómez Pérez",
  "email": "laura.actualizada@gmail.com"
}'
{
  "dni": "11111111A",
  "message": "Patient updated successfully"
}
```

Delete de un paciente concreto

```
usuario@ubuntu:~$ curl -X DELETE http://127.0.0.1:5000/patients/11111111A
{
  "dni": "11111111A",
  "message": "Patient deleted successfully"
}
usuario@ubuntu:~$ curl -X GET http://127.0.0.1:5000/patients/11111111A
{
  "error": "Patient not found"
}
usuario@ubuntu:~$
```

Put, modificación de una intervención concreta

```
usuario@ubuntu:~$ curl -X PUT http://127.0.0.1:5000/intervenciones/1 \
-H "Content-Type: application/json" \
-d '{
  "nombre": "Extracción muela actualizada",
  "tipo_intervencion": "Cirugía avanzada",
  "id_sala": 2
}'
{
  "id_intervencion": 1,
  "message": "Intervenci\u00f3n updated successfully"
}
```

Delete de una intervención concretra

```
usuario@ubuntu:~$ curl -X DELETE http://127.0.0.1:5000/intervenciones/1
{
  "id_intervencion": 1,
  "message": "Intervenci\u00f3n deleted successfully"
}
```

Presupuesto.

La implementación del sistema de gestión de la clínica dental Sonrisas requiere la consideración de distintos factores que afectan al presupuesto total. Estos incluyen la adquisición de hardware y software, la configuración de la base de datos PostgreSQL, el desarrollo de la API REST con Flask, la carga de datos iniciales y las pruebas del sistema, así como el tiempo de trabajo estimado de los desarrolladores.

Se ha estimado que el desarrollo completo por parte de un equipo de dos personas con conocimientos en bases de datos y desarrollo backend puede repartirse en las siguientes tareas principales:

Concepto	Descripción	Horas estimadas	Coste/hora (€)	Subtotal (€)

Análisis y diseño de la base de datos	Modelado ER, tablas, relaciones, restricciones, integridad	12	20	240
Implementación de la base de datos	Creación de tablas en PostgreSQL, disparadores, checks	10	20	200
Carga de datos de ejemplo	Inserción de pacientes, profesionales, citas, materiales	6	20	120
Desarrollo de API REST en Flask	Endpoints CRUD, validaciones, manejo de errores	20	25	500
Pruebas y depuración	Pruebas unitarias y de integración, corrección de errores	10	20	200
Documentación	Diagramas, justificación de diseño, manual de uso	8	15	120

Subtotal estimado: 1.380 €

Adicionalmente, se incluyen costes de infraestructura y licencias necesarias para el desarrollo y prueba del sistema:

Concepto	Descripción	Coste (€)
Servidor de desarrollo	Máquina virtual o física para PostgreSQL y Flask	200
Software adicional	IDE, librerías, entornos virtuales	0 (software libre)
Backup y almacenamiento	Almacenamiento seguro para datos de prueba	50

Total adicional: 250 €

Total estimado: 1.630 €

Este presupuesto considera únicamente los recursos necesarios para desarrollar e implementar la base de datos y la API REST, incluyendo el tiempo de programación, pruebas y documentación. No se incluyen costes recurrentes de mantenimiento de la clínica ni personal administrativo o sanitario.