Universidad de La Laguna

Administración y Diseño de Base de Datos

Práctica Final: Proyecto Hospital

Escuela Superior de Ingeniería y Tecnología
Ingeniería Informática
2023-2024

Arturo Pestana Ortiz
Marcelo Daniel Choque Mamani

Índice

1. Introducción

En el campo de la informática la integración de tecnologías como las bases de datos relacionales y las interfaces de programación de aplicaciones (API) se ha vuelto esencial para potenciar la eficiencia y la accesibilidad en diversos contextos, de tal forma que se encuentra presente en un sinfín de áreas y campos. En este proyecto, nos sumergimos en el mundo de la gestión de la información médica, utilizando una base de datos relacional PostgreSQL y desarrollando una API REST con Flask para crear un Sistema de Gestión de Consultas Médicas para un hospital.

2. Objetivos

Los objetivos primordiales de nuestro proyecto de base de datos hospitalaria se estructuran en una serie de metas clave diseñadas para abordar las complejidades y necesidades inherentes a la gestión hospitalaria moderna:

- **Modelado Integral de Datos:** Nuestro primer objetivo es representar de manera precisa los requisitos de datos esenciales para gestionar las operaciones hospitalarias. Esto implica establecer un marco que permita registrar y rastrear pacientes, administrar clientes, coordinar consultas médicas y supervisar el inventario de materiales en los diversos departamentos del hospital. Aseguramos que cada entidad y relación sea cuidadosamente diseñada para reflejar con precisión las operaciones y flujos de trabajo del mundo real en un entorno hospitalario.
- **Identificación de Entidades y Atributos Críticos:** El siguiente paso es determinar las entidades centrales y sus atributos correspondientes que formarán parte de nuestra base de datos. Esto abarca desde la estructuración de departamentos y personal médico hasta la gestión de clientes y la información vital de sus contactos, como familiares. Al definir estos elementos, nos aseguramos de capturar todos los detalles relevantes que contribuyen al funcionamiento eficiente y eficaz del hospital.
- **Definición Precisa de Relaciones:** Una vez identificadas las entidades y atributos, es esencial establecer relaciones claras y coherentes entre ellas. Por ejemplo, estableceremos relaciones definidas entre médicos y pacientes para reflejar consultas, tratamientos y seguimientos adecuados. Esta etapa garantiza que cada interacción y conexión dentro del sistema se maneje de manera fluida y sin ambigüedades, optimizando así la entrega de atención médica y los procesos administrativos.
- **Normalización Efectiva de la Base de Datos:** Finalmente, para garantizar la integridad, eficiencia y coherencia de nuestra base de datos, llevaremos a cabo un proceso exhaustivo de normalización. Este proceso busca eliminar redundancias innecesarias, minimizar la dependencia de datos y asegurar que la información se

almacene de manera organizada y estructurada. Al hacerlo, no solo mejoramos la precisión y la fiabilidad de los datos, sino que también facilitamos futuras actualizaciones y expansiones del sistema hospitalario.

3. Supuesto

El objetivo principal de la base de datos del hospital es gestionar eficientemente la información relacionada con pacientes, clientes, empleados, departamentos, materiales, proveedores y pagos. La base de datos busca facilitar el seguimiento y la administración de pacientes, garantizando una atención médica adecuada y una gestión eficaz de los recursos hospitalarios.

- **Gestión de Pacientes:** Se pretende almacenar información detallada sobre cada paciente, incluyendo su nombre, fecha de nacimiento, género y relación con un cliente específico. Además, se busca registrar la información de familiares asociados a cada paciente.
- **Registro de Clientes:** La base de datos almacena datos de clientes que pueden estar asociados a uno o varios pacientes. Se incluyen detalles como nombre, dirección, número de teléfono y correo electrónico.
- **Gestión de Empleados:** Se mantiene un registro exhaustivo de los empleados del hospital, incluyendo médicos y auxiliares. Cada empleado está asociado a un departamento específico y puede pertenecer a un grupo de prácticas particular.
- **Administración de Departamentos y Materiales:** La base de datos gestiona los departamentos hospitalarios, asignando materiales específicos a cada departamento a través de proveedores. Se busca asegurar un suministro constante de materiales médicos esenciales para cada área.
- **Relación con Proveedores:** Se mantiene una relación estrecha con proveedores externos que suministran materiales médicos y otros recursos esenciales al hospital. Cada proveedor tiene asignado un conjunto específico de materiales y está asociado a uno o más departamentos.
- **Control de Pagos:** La base de datos registra los pagos realizados por los clientes, ya sea en efectivo o mediante tarjeta, asegurando una contabilidad precisa y una gestión financiera adecuada.

4. Entidades y relaciones

Las entidades que podemos distinguir en nuestro modelo son las siguientes:

- **Cliente:** Representa a la persona mayor de edad que trae a su hijo o dependiente para una consulta en el hospital.

- **Pago:** Esta entidad refleja el método de pago utilizado, junto con el importe asociado a la transacción. Sus atributos clave incluyen el identificador de pago y el monto.
- **Paciente:** Aquí se registra la información relacionada con el paciente que asiste a la consulta. Los atributos asociados a esta entidad son el identificador del paciente, edad, nombre, fecha de nacimiento y sexo.
- **Personal:** Esta entidad engloba a todos los empleados que trabajan dentro del hospital. Se distinguen dos tipos principales:
    - **Doctores:** Profesionales encargados de la atención médica especializada.
    - **Auxiliares:** Personal de apoyo en diversas áreas del hospital.
- **Departamento:** Representa las diferentes áreas o unidades especializadas dentro del hospital donde el personal puede estar asignado. Algunos ejemplos mencionados son cardiología, radiología, pediatría, entre otros.
- **Material:** Esta entidad hace referencia a los diversos insumos o materiales médicos utilizados en el hospital, como jeringas, bisturíes y otros instrumentos o suministros relevantes.
- **Proveedores:** Esta entidad representa a las empresas o entidades que suministran los productos o materiales al hospital. Estos proveedores son vitales para garantizar el abastecimiento adecuado de insumos a los diferentes departamentos y áreas del hospital.

5. Script Postgres SQL

Para la implementación de la base de datos, utilizamos un script sql, que conecta con el sistema gestor de bases de datos relacional PostgreSQL. Dicho fichero se puede consultar en el siguiente enlace

En un primer paso, se procede a la creación de la base de datos, estableciendo las bases sobre las cuales se construirán las tablas. La conexión a la base de datos recién creada se establece para preparar el terreno para la siguiente fase.
Posteriormente, se lleva a cabo la creación de las tablas, siguiendo el modelo relacional previamente definido. Se han diseñado tablas para manejar relaciones triples y se han introducido tablas específicas para detallar el tipo de empleado, pasa consulta y la información de pago. En el script, se incorporan condiciones durante la inserción de datos, como el uso de "NOT NULL" para asegurar que ciertos campos no queden vacíos.

En el proceso de diseño, se han aplicado restricciones a las claves ajenas para mantener la integridad referencial entre las tablas. Se ha adoptado la estrategia de eliminación en cascada para todas las tablas, garantizando que la eliminación de registros principales se refleje adecuadamente en las tablas relacionadas.

Esta fase inicial sienta las bases sólidas para la estructura de la base de datos, incorporando medidas que garantizan la consistencia y la integridad de los datos. El enfoque meticuloso en la creación de tablas y la gestión de relaciones sienta las bases para un sistema robusto y eficiente.

Tras la creación de las tablas se realizaron verificaciones o “checks” necesarias para las tablas oportunas. Las verificaciones implementadas fueron las siguientes:

- El email de los clientes debe ser único.
- El teléfono de los clientes debe ser único, tener 9 dígitos y comprendidos entre 100.000.000 y 999.999.999.
- El teléfono de los familiares debe ser único, tener 9 dígitos y comprendidos entre 100.000.000 y 999.999.999.
- En la tabla teléfonos, de igual manera se cumplen las premisas antes nombradas.
- La edad de los pacientes no puede ser negativo.
- La fecha de los pacientes debe ser superior a 01-01-1900.
- Los tipos de datos de género están restringidos a 2 posibles valores:
    - Masculino
    - Femenino
- El nombre de los departamentos debe de ser único.
- En los familiares el dni debe ser unico.
- El pago debe ser superior a 0 €.
- Restringimos el tipo de pago a efectivo o tarjeta.
- Del modo que hicimos con lo teléfonos hacemos los mismo con los teléfonos de proveedores.
- Finalmente, que la cantidad de materiales aprovisionados sea mayor a 0, es decir, al menos 1.

En cuanto a los disparadores, desarrollamos 4 triggers:

- El primero comprueba que al ingresar una tarjeta ha expirado o no.
- La segunda comprueba que el número de tarjeta no sea repetido.
- El tercero si dejamos el campo de fecha en la que pasa consulta, dejamos este campo vacío, se rellena automáticamente con la fecha del día actual.

6. Consulta Base de Datos

Cargamos la base de datos “bbdd_hospital”

```
usuario@ubuntu:~/practicas/final$ sudo -u postgres psql -d bbdd_hospital -a -f tablas.sql
-- Administración y Diseño de Bases de Datos
-- Proyecto de Hospital
--
-- Realizado por:
--  Marcelo Daniel Choque Mamani
--  Arturo Pestana Ortiz
--
-- Creacion de la base de datos
--
\c postgres
You are now connected to database "postgres" as user "postgres".
-- Elimina la base de datos si existe
DROP DATABASE IF EXISTS bbdd_hospital;
DROP DATABASE
-- Crea la base de datos
CREATE DATABASE bbdd_hospital WITH TEMPLATE = template0 ENCODING = 'UTF8';
CREATE DATABASE
```

```
bbdd_hospital=# select * from cliente;
 id_cliente |      nombre       |          direccion           | telefono  |         email
------------+-------------------+------------------------------+-----------+-------------------------
          1 | Carlos Tevez      | Salón de la Fama 1           | 611111111 | carlosTevez@email.com
          2 | Brad Pitt         | Miami, Número 2              | 623456789 | bradPitt@email.com
          3 | Tom Hanks         | Beverly Hills, Calle Principal | 655511122 | tomHanks@email.com
          4 | Leonardo DiCaprio | Malibu, Frente al Mar        | 677788899 | leoDiCapric@email.com
          5 | Dwayne Johnson    | Hollywood Blvd, Casa 5       | 633344455 | dwayneJohnson@email.com
          6 | Chris Hemsworth   | Sydney, Australia            | 699988877 | chrisHemsworth@email.com
          7 | Robert Downey Jr. | Los Angeles, Rodeo Drive     | 666777888 | robertDowney@email.com
          8 | Keanu Reeves      | New York City, Times Square  | 611122233 | keanuReeves@email.com
```

Vemos que no deja insertar el email repetido, se estaría violando una restricción check de unicidad.

```
bbdd_hospital=# INSERT INTO cliente (id_cliente, nombre, direccion, telefono, email)
VALUES
  (98, 'Sergio Ramos', 'Calle Madrid 1', 999888777, 'sergic@example.com'),
  (99, 'David Silva', 'Avenida España 2', 999888777, 'sergic@example.com');
ERROR:  duplicate key value violates unique constraint "check_email_unique"
DETAIL:  Key (email)=(sergic@example.com) already exists.
```

Lo mismo ocurre con teléfono.

```
  (98, 'Sergio Ramos', 'Calle Madrid 1', 999888777, 'sergic@example.com'),
  (99, 'David Silva', 'Avenida España 2', 999888777, 'sergio123@example.com');
ERROR:  duplicate key value violates unique constraint "check_telefono_cliente_unique"
DETAIL:  Key (telefono)=(999888777) already exists.
```

Del mismo modo, al introducir el género/sexo se una paciente este solo podra se [‘masculino’ / ’feminino’], rechazando cualquier otro valor.

```
bbdd_hospital=# INSERT INTO paciente (nombre, fecha_nacimiento, genero, id_cliente) VALUES
bbdd_hospital-#   ('Andrew Garfield', '1984-12-30', 'Hombre', 100);
ERROR:  invalid input value for enum genero_t: "Hombre"
LINE 2:   ('Andrew Garfield', '1984-12-30', 'Hombre', 100);
                                            ^
```

Realizamos consulta del empleado con codigo_empleado = 1, para ver a que paciente paso consulta y en que fechas, involucra las tablas pasa_consulta, empleado, y paciente.

codigo	fecha	diagnostico	id_paciente	codigo_empleado	nombre_empleado	nombre_paciente
1	2023-01-15	Consulta de rutina	1	1	Kurt Cobain	LeBron James
6	2023-06-10	Consulta dermatológica	6	1	Kurt Cobain	Kawhi Leonard
10	2023-11-15	Consulta de rutina	1	1	Kurt Cobain	LeBron James
11	2023-09-10	Consulta dermatológica	8	1	Kurt Cobain	Damian Lillard

Realizamos una consulta para ver los datos de los clientes que pagaron con tarjeta y los datos de la propia tarjeta

```sql
SELECT
  cliente.id_cliente,
  cliente.nombre AS nombre_cliente,
  cliente.direccion,
  cliente.telefono,
  cliente.email,
  tarjeta.num_card,
  tarjeta.fecha_caduca
FROM cliente
JOIN pago ON cliente.id_cliente = pago.id_cliente
JOIN tarjeta ON pago.id_pago = tarjeta.id_pago
WHERE pago.tipo = 'tarjeta';
```

id_cliente	nombre_cliente	direccion	telefono	email	num_card	fecha_caduca
2	Brad Pitt	Miami, Número 2	623456789	bradPitt@email.com	1234567890123456	2024-12-31
4	Leonardo DiCaprio	Malibu, Frente al Mar	677788899	leoDiCapric@email.com	9876543210987654	2028-12-30
5	Dwayne Johnson	Hollywood Blvd, Casa 5	633344455	dwayneJohnson@email.com	1111222233334444	2024-12-31
6	Chris Hemsworth	Sydney, Australia	699988877	chrisHemsworth@email.com	5555666677778888	2029-12-30
7	Robert Downey Jr.	Los Angeles, Rodeo Drive	666777888	robertDowney@email.com	9999888877776666	2025-01-15
8	Keanu Reeves	New York City, Times Square	611122233	keanuReeves@email.com	1234123412341234	2027-11-30

Realizamos una consulta sobre 4 tablas, obtenemos la información de los pacientes así como sus familiares de contacto (pueden tener más de 1 familiar) y seguidamente la información de la consulta (fecha, diagnostico) y finalmente el empleado que atendió la consulta.

```
SELECT
  pacienteVar.id_paciente,
  pacienteVar.nombre AS nombre_paciente,
  famVar.nombre AS nombre_familiar,
  consultaVar.codigo AS codigo_consulta,
  consultaVar.fecha AS fecha_consulta,
  consultaVar.diagnostico,
  empleadoVar.codigo_p AS codigo_empleado,
  empleadoVar.nombre AS nombre_empleado
FROM paciente pacienteVar
JOIN familiar famVar ON pacienteVar.id_paciente = famVar.id_paciente
JOIN pasa_consulta consultaVar ON pacienteVar.id_paciente = consultaVar.id_paciente
JOIN empleado empleadoVar ON consultaVar.codigo_p = empleadoVar.codigo_p;
```

id_paciente	nombre_paciente	nombre_familiar	codigo_consulta	fecha_consulta	diagnostico	codigo_empleado	nombre_empleado
1	LeBron James	Diego Maradona Jr.	1	2023-01-15	Consulta de rutina	1	Kurt Cobain
1	LeBron James	Lionel Messi	1	2023-01-15	Consulta de rutina	1	Kurt Cobain
1	LeBron James	Marcelo Tinelli	1	2023-01-15	Consulta de rutina	1	Kurt Cobain
1	LeBron James	Susana Gimenez	1	2023-01-15	Consulta de rutina	1	Kurt Cobain
5	Luka Dončić	Pampita Veron	5	2023-05-05	Vacunación anual	5	Dave Grohl
6	Kawhi Leonard	Juan Martín Del Potro	6	2023-06-10	Consulta dermatológica	1	Kurt Cobain
7	Anthony Davis	Lali Espósito	7	2023-07-15	Exámenes cardiológicos	2	Eddie Vedder
8	Damian Lillard	Javier Gerardo Milei	8	2023-08-20	Seguimiento postoperatorio	3	Chris Cornell
1	LeBron James	Diego Maradona Jr.	10	2023-11-15	Consulta de rutina	1	Kurt Cobain
1	LeBron James	Lionel Messi	10	2023-11-15	Consulta de rutina	1	Kurt Cobain
1	LeBron James	Marcelo Tinelli	10	2023-11-15	Consulta de rutina	1	Kurt Cobain
1	LeBron James	Susana Gimenez	10	2023-11-15	Consulta de rutina	1	Kurt Cobain
8	Damian Lillard	Javier Gerardo Milei	11	2023-09-10	Consulta dermatológica	1	Kurt Cobain

Realizamos una consulta de la cantidad de material suministrado.

```
bbdd_hospital=# SELECT
bbdd_hospital-#   a.id_material,
bbdd_hospital-#   m.nombre_material,
bbdd_hospital-#   SUM(a.cantidad) AS cantidad_total
bbdd_hospital-# FROM aprovisionamiento a
bbdd_hospital-# JOIN material m ON a.id_material = m.id_material
bbdd_hospital-# GROUP BY a.id_material, m.nombre_material;
 id_material | nombre_material  | cantidad_total
-------------+------------------+----------------
           3 | Quimioterapia    |            320
           4 | Bisturí          |             70
           2 | Monitor cardíaco |            125
           1 | Jeringas         |            160
           5 | Vacunas          |            240
```

Realizamos otra consulta de los materiales que hay en los distintos departamentos y de quienes fueron los proveedores de dichos materiales, así mismo, la cantidad de material.

```
bbdd_hospital=# SELECT
  aprovisionamiento.id_dpto AS id_departamento,
  aprovisionamiento.id_material,
  aprovisionamiento.id_proveedor,
  material.nombre_material,
  proveedores.nombre AS nombre_proveedor,
  aprovisionamiento.cantidad
FROM aprovisionamiento
JOIN material ON aprovisionamiento.id_material = material.id_material
JOIN proveedores ON aprovisionamiento.id_proveedor = proveedores.id_proveedor;
 id_departamento | id_material | id_proveedor | nombre_material |       nombre_proveedor        | cantidad
-----------------+-------------+--------------+-----------------+-------------------------------+----------
               1 |           1 |            1 | Jeringas        | Suministros Médicos S.A.      |      100
               2 |           2 |            2 | Monitor cardíaco | Equipos Médicos SL           |       50
               3 |           3 |            3 | Quimioterapia   | Farmacia ABC                  |      200
               4 |           4 |            4 | Bisturí         | Instrumentos Quirúrgicos Inc. |       30
               5 |           5 |            5 | Vacunas         | Vacunas Pro SLU               |      150
               1 |           2 |            3 | Monitor cardíaco | Farmacia ABC                 |       75
               2 |           3 |            4 | Quimioterapia   | Instrumentos Quirúrgicos Inc. |      120
               3 |           4 |            5 | Bisturí         | Vacunas Pro SLU               |       40
               4 |           5 |            1 | Vacunas         | Suministros Médicos S.A.      |       90
               5 |           1 |            2 | Jeringas        | Equipos Médicos SL            |       60
```

Forzamos la introducción de datos erroneos para poner de manifiesto uno de los disparadores en este caso una tarjeta con la fecha caducada.

```
bbdd_hospital=# INSERT INTO tarjeta (num_card, fecha_caduca, id_pago)
VALUES
   (1234567890123456, '2023-01-01', 2);
ERROR:  La tarjeta ha caducado ! ! !
CONTEXT:  PL/pgSQL function check_card_expiration() line 4 at RAISE
```

Forzamos la introducción de un empleado asignando un “id_dpto” y “id_grupoPracticas” en un mismo insert, cosa que no puede ser, debido a que un empleado solo puede tener un identificador.

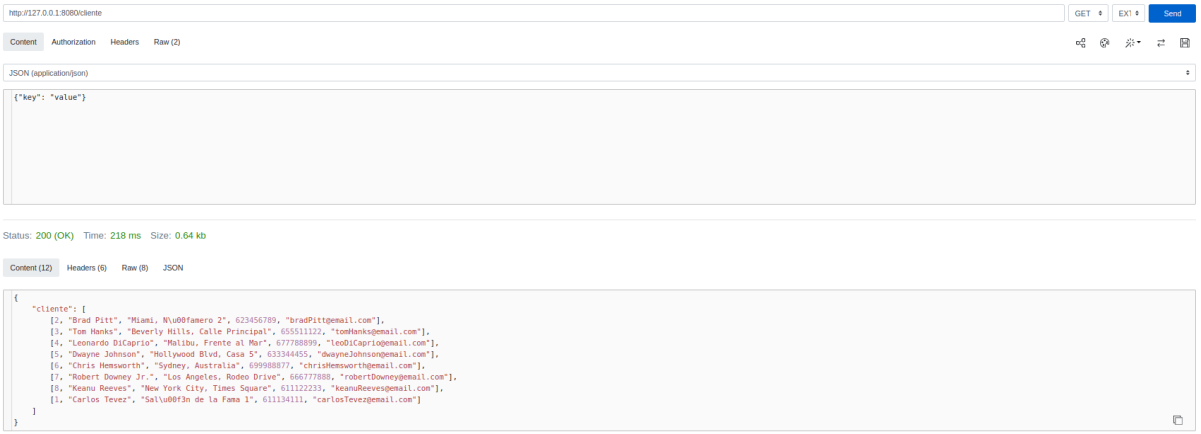
```
bbdd_hospital=# INSERT INTO empleado (codigo_p, nombre, dni, id_dpto, id_grupoPracticas) VALUES
  (6, 'Konan el Barbaro', '111222999', 99, 99);
ERROR:  Un empleado debe pertenecer a un departamento o a un grupo de prácticas.
CONTEXT:  PL/pgSQL function validar_pertenencia() line 5 at RAISE
```

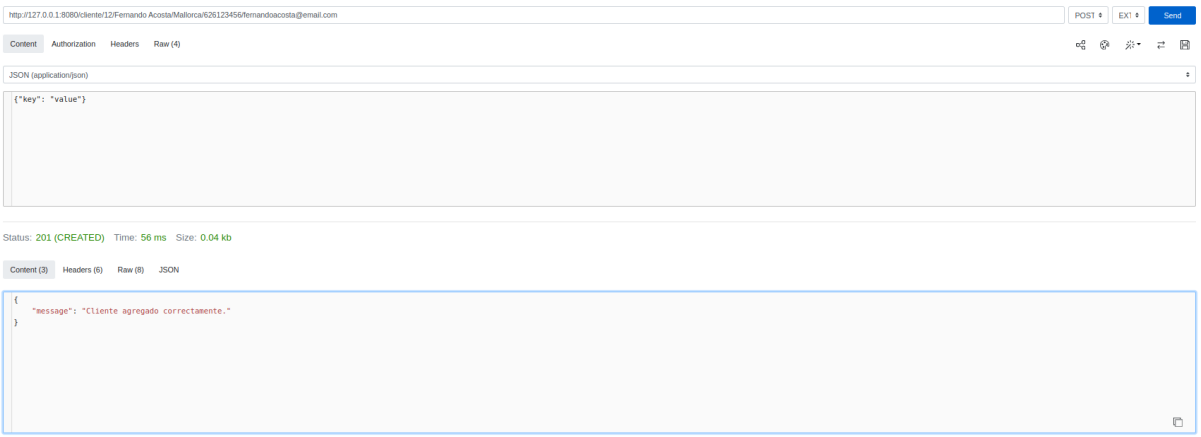
En la siguiente imagen se muestra la modificación de uno de los pacientes, concretamente a la fecha de nacimiento.

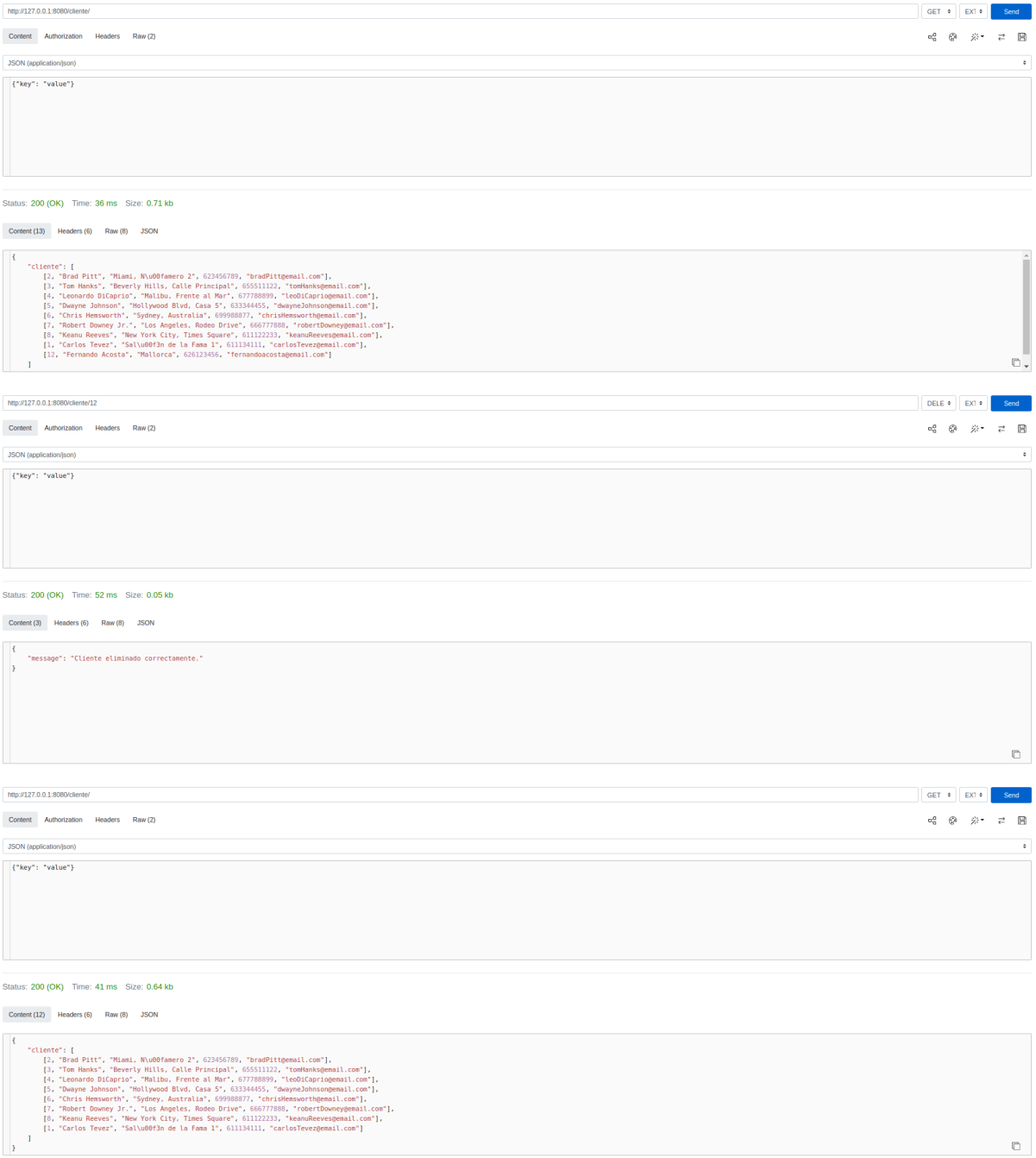
```
my_base2=# SELECT id_paciente, nombre, fecha_nacimiento, edad, genero
FROM paciente
LIMIT 3;
 id_paciente |        nombre         | fecha_nacimiento | edad |  genero
-------------+-----------------------+------------------+------+-----------
           3 | Stephen Curry         | 1988-03-14       |   35 | Masculino
           4 | Giannis Antetokounmpo | 1994-12-06       |   29 | Masculino
           5 | Luka Dončić           | 1999-02-28       |   24 | Masculino
(3 rows)

my_base2=# UPDATE paciente
SET fecha_nacimiento = '2000-01-01'
WHERE id_paciente = 2;
UPDATE 1

 id_paciente |        nombre         | fecha_nacimiento | edad |  genero   | id_cliente
-------------+-----------------------+------------------+------+-----------+------------
          14 | Vivianne Miedema      | 1996-07-15       |   27 | Femenino  |          1
           1 | LeBron James          | 2000-01-01       |   23 | Masculino |          1
           2 | Kevin Durant          | 2000-01-01       |   23 | Masculino |          2
```

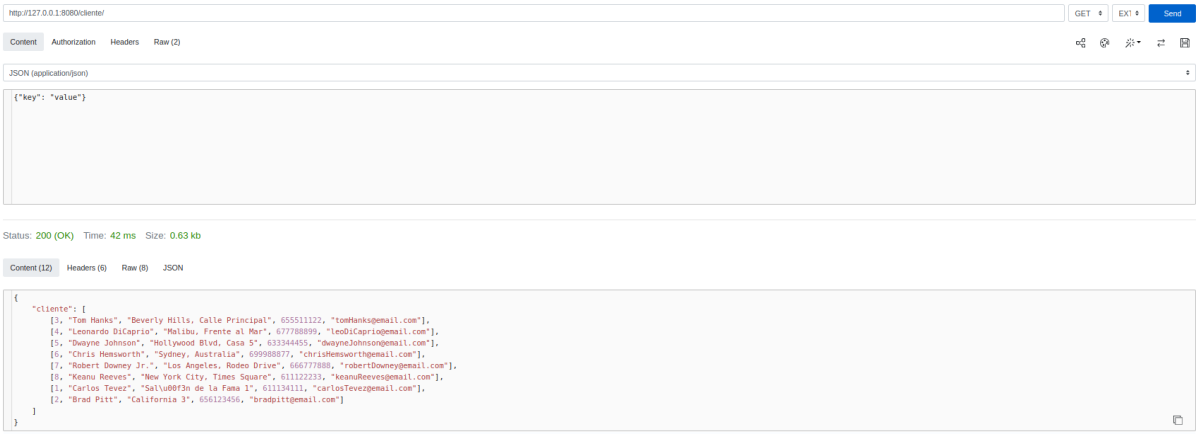
7. API REST con Flask (CRUD)

Añadimos todos los tipos de consultas que podemos hacer con la API:






http://127.0.0.1:8080/cliente/
GET
EX1
Send
Content
Authorization
Headers
Raw (2)
JSON (application/json)
{"key": "value"}
Status: 200 (OK) Time: 36 ms Size: 0.71 kb
Content (13)
Headers (6)
Raw (8)
JSON
{
"cliente": [
[2, "Brad Pitt", "Miami, N\u00famero 2", 623456789, "bradPitt@email.com"],
[3, "Tom Hanks", "Beverly Hills, Calle Principal", 655511122, "tomHanks@email.com"],
[4, "Leonardo DiCaprio", "Malibu, Frente al Mar", 677788899, "leoDiCaprio@email.com"],
[5, "Dwayne Johnson", "Hollywood Blvd, Casa 5", 633344455, "dwayneJohnson@email.com"],
[6, "Chris Hemsworth", "Sydney, Australia", 699988877, "chrisHemsworth@email.com"],
[7, "Robert Downey Jr.", "Los Angeles, Rodeo Drive", 666777888, "robertDowney@email.com"],
[8, "Keanu Reeves", "New York City, Times Square", 611122233, "keanuReeves@email.com"],
[1, "Carlos Tevez", "Sal\u00f3n de la Fama 1", 611134111, "carlosTevez@email.com"],
[12, "Fernando Acosta", "Mallorca", 626123456, "fernandoacosta@email.com"]
]
http://127.0.0.1:8080/cliente/12
DELE
EX1
Send
Content
Authorization
Headers
Raw (2)
JSON (application/json)
{"key": "value"}
Status: 200 (OK) Time: 52 ms Size: 0.05 kb
Content (3)
Headers (6)
Raw (8)
JSON
{
"message": "Cliente eliminado correctamente."
}
http://127.0.0.1:8080/cliente/
GET
EX1
Send
Content
Authorization
Headers
Raw (2)
JSON (application/json)
{"key": "value"}
Status: 200 (OK) Time: 41 ms Size: 0.64 kb
Content (12)
Headers (6)
Raw (8)
JSON
{
"cliente": [
[2, "Brad Pitt", "Miami, N\u00famero 2", 623456789, "bradPitt@email.com"],
[3, "Tom Hanks", "Beverly Hills, Calle Principal", 655511122, "tomHanks@email.com"],
[4, "Leonardo DiCaprio", "Malibu, Frente al Mar", 677788899, "leoDiCaprio@email.com"],
[5, "Dwayne Johnson", "Hollywood Blvd, Casa 5", 633344455, "dwayneJohnson@email.com"],
[6, "Chris Hemsworth", "Sydney, Australia", 699988877, "chrisHemsworth@email.com"],
[7, "Robert Downey Jr.", "Los Angeles, Rodeo Drive", 666777888, "robertDowney@email.com"],
[8, "Keanu Reeves", "New York City, Times Square", 611122233, "keanuReeves@email.com"],
[1, "Carlos Tevez", "Sal\u00f3n de la Fama 1", 611134111, "carlosTevez@email.com"]
]
}


http://127.0.0.1:8080/cliente/
GET
EX1
Send
Content
Authorization
Headers
Raw (2)
JSON (application/json)
{"key": "value"}
Status: 200 (OK) Time: 42 ms Size: 0.63 kb
Content (12)
Headers (6)
Raw (8)
JSON
{
"cliente": [
[3, "Tom Hanks", "Beverly Hills, Calle Principal", 655511122, "tomHanks@email.com"],
[4, "Leonardo DiCaprio", "Malibu, Frente al Mar", 677788899, "leoDiCaprio@email.com"],
[5, "Dwayne Johnson", "Hollywood Blvd, Casa 5", 633344455, "dwayneJohnson@email.com"],
[6, "Chris Hemsworth", "Sydney, Australia", 699988877, "chrisHemsworth@email.com"],
[7, "Robert Downey Jr.", "Los Angeles, Rodeo Drive", 666777888, "robertDowney@email.com"],
[8, "Keanu Reeves", "New York City, Times Square", 611122233, "keanuReeves@email.com"],
[1, "Carlos Tevez", "Sal\u00f3n de la Fama 1", 611134111, "carlosTevez@email.com"],
[2, "Brad Pitt", "California 3", 656123456, "bradpitt@email.com"]
]
}