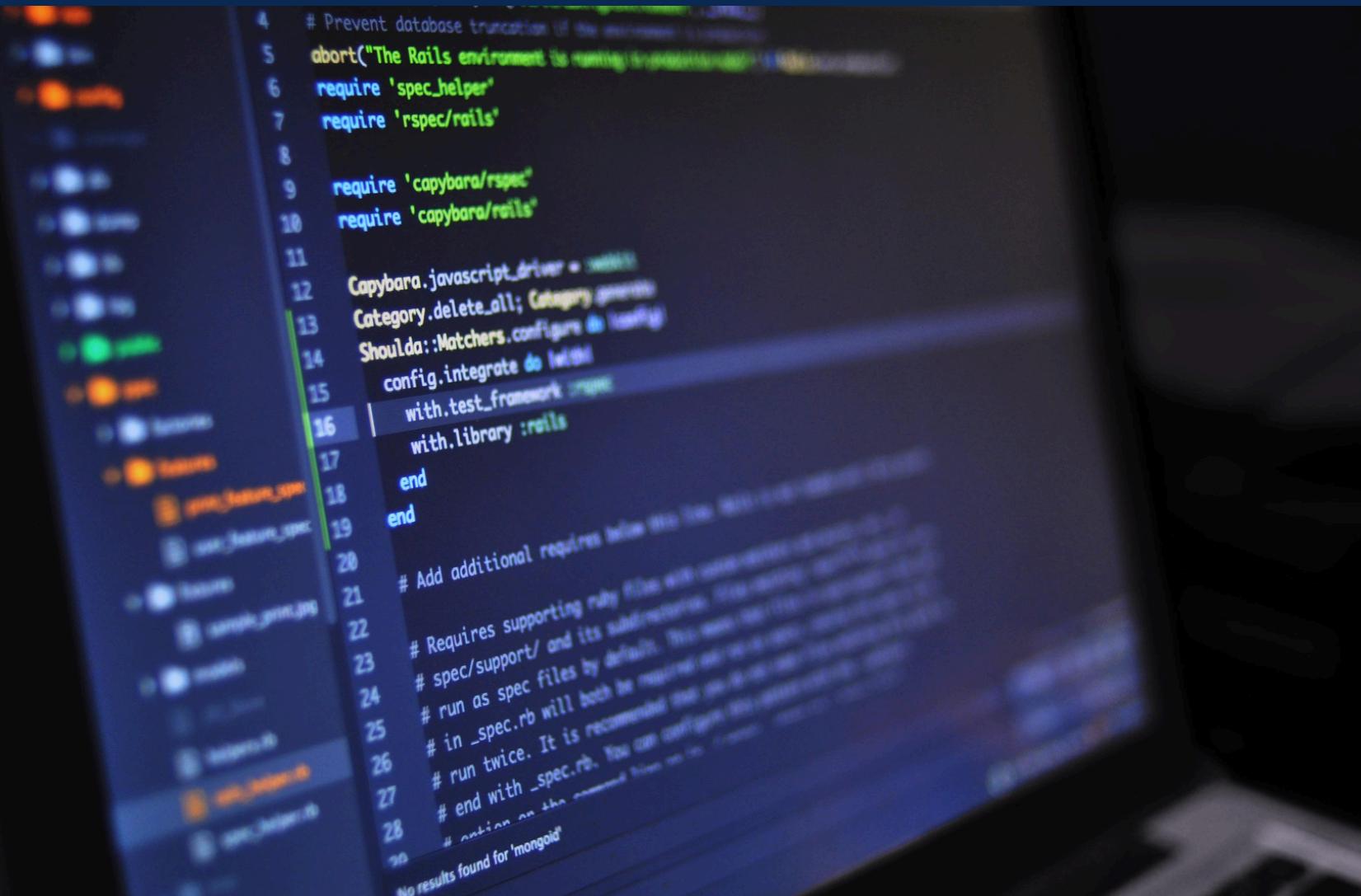


TEAM CHALLENGE SQL MURDER DISEÑO DE BASE DE DATOS

SHEILA RODRÍGUEZ
SANDRA LÓPEZ
DAVID TOROSYAN

INDICE DE CONTENIDOS

1. PARTE 1: SQL MURDER MYSTERY	3
2. PARTE 2:DISEÑO DE BASE DE DATOS AVANZADAS	17



```
4 # Prevent database truncation if the test fails
5 abort("The Rails environment is running in production mode")
6 require 'spec_helper'
7 require 'rspec/rails'

8 require 'capybara/rspec'
9 require 'capybara/rails'

10 Capybara.javascript_driver = :webkit
11 Category.delete_all; Category.create!(name: "Electronics")
12 Shoulda::Matchers.configure do |config|
13   config.integrate do |sp|
14     sp.with.test_framework :rspec
15     sp.with.library :rails
16   end
17 end
18
19 # Add additional requires below this line if you need them
20
21 # Requires supporting ruby files with custom matchers
22 # in spec/support/ and its subdirectories. This directory
23 # also contains supporting files for this generator.
24 # run as spec files by default. You can change this
25 # to run specs with `rake spec` or `rake db:setup` with
26 # `--tag=feature` or `--tag=team-challenge` respectively
27 # run twice. It is recommended that you do not name
28 # your files with the _spec.rb suffix when you
# mention in the command above on the command line
29
30 # no results found for 'mongoid'
```

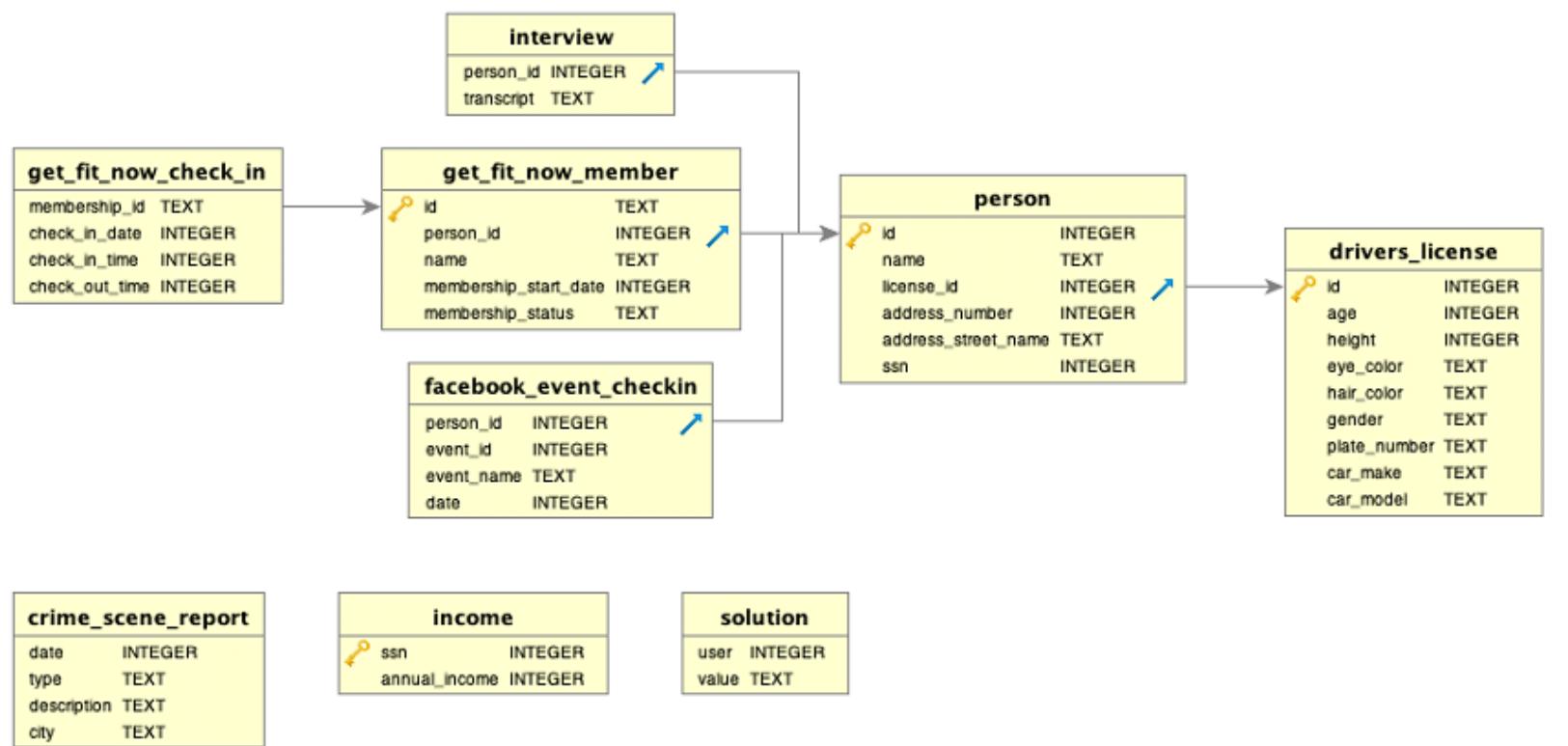
TEAM CHALLENGE - 02

1/ SQL MURDER MYSTERY

TEAM CHALLENGE - 03

1/ MURDER MYSTERY

SQL Murder Mystery es un juego interactivo que combina lógica detectivesca con SQL.
Nuestro objetivo: resolver un asesinato utilizando consultas a una base de datos real.



1 / MURDER MYSTERY

Se busca un asesinato ocurrido el 15 de enero de 2018 en SQL City. Consulta SQL usada:

```
SELECT *
FROM crime_scene_report
WHERE date = 20180115 and city = "SQL City"
```

date	type	description	city
20180115	assault	Hamilton: Lee, do you yield? Burr: You shot him in the side! Yes he yields!	SQL City
20180115	assault	Report Not Found	SQL City
20180115	murder	Security footage shows that there were 2 witnesses. The first witness lives at the last house on "Northwestern Dr". The second witness, named Annabel, lives somewhere on "Franklin Ave".	SQL City

1 / MURDER MYSTERY

Se busca un asesinato ocurrido el 15 de enero de 2018 en SQL City. Consulta SQL usada:

```
SELECT *
FROM crime_scene_report
WHERE date = 20180115 and city = "SQL City"
```

date	type	description	city
20180115	assault	Hamilton: Lee, do you yield? Burr: You shot him in the side! Yes he yields!	SQL City
20180115	assault	Report Not Found	SQL City
20180115	murder	Security footage shows that there were 2 witnesses. The first witness lives at the last house on "Northwestern Dr". The second witness, named Annabel, lives somewhere on "Franklin Ave".	SQL City

En la tabla anterior vemos que ha habido, efectivamente, un asesinato.

“Las imágenes de seguridad muestran que había dos testigos. El primero vive en la última casa de Northwestern Dr. El segundo, llamado Annabel, vive en algún lugar de Franklin Ave.”

Sabemos pues que hay un testigo que vive en la última casa de la calle Northwestern Dr. y otro que se llama Annabel que vive en algún lugar de Franklin Ave.

1/ MURDER MYSTERY

Como de la segunda tenemos más datos vamos a empezar por ese testigo testeando en la tabla “person”.

Código:

```
SELECT *  
FROM person  
WHERE name LIKE "Annabel%"
```

id	name	license_id	address_number	address_street_name	ssn
16371	Annabel Miller	490173	103	Franklin Ave	318771143
78354	Annabell Siona	158932	978	Whitewater Dr	800278294
78799	Annabell Droneburg	984316	1944	W Natalie Dr	478793500
86541	Annabell Zwilling	709133	1859	Patti Rd	332961158

Salen 4 Annabel pero sólo una que vive en Franklin Ave. Verifiquemos ahora los eventos de facebook, con el id 16371.

Código:

```
SELECT *  
FROM facebook_event_checkin  
WHERE person_id = 16371
```

person_id	event_id	event_name	date
16371	4719	The Funky Grooves Tour	20180115

1 / MURDER MYSTERY

Annabel estuvo en un evento en la misma fecha que ocurrió el asesinato. Es poco probable que sea ella la asesina. Pero veamos su declaración a la policía:

Código:

```
SELECT *  
FROM interview  
WHERE person_id = 16371
```

person_id	transcript
16371	I saw the murder happen, and I recognized the killer from my gym when I was working out last week on January the 9th.

Tenemos una testigo directa del asesinato. Reconoce que va a su gimnasio y estaba entrenando el día 9 de Enero.

Veamos la ficha de gimnasio de Annabel:

Código:

```
SELECT *  
FROM get_fit_now_member  
WHERE person_id = 16371
```

id	person_id	name	membership_start_date	membership_status
90081	16371	Annabel Miller	20160208	gold

Annabel es miembro oro del gimnasio y lleva acudiendo un par de años antes del asesinato.

1 / MURDER MYSTERY

Veamos sus check in para el día en que vio al asesino:

Código:

```
SELECT *  
FROM get_fit_now_check_in  
WHERE check_in_date = 20180109
```

membership_id	check_in_date	check_in_time	check_out_time
X0643	20180109	957	1164
UK1F2	20180109	344	518
XTE42	20180109	486	1124
1AE2H	20180109	461	944
6LSTG	20180109	399	515
7MWHJ	20180109	273	885
GE5Q8	20180109	367	959
48Z7A	20180109	1600	1730
48Z55	20180109	1530	1700
90081	20180109	1600	1700

Annabel es la miembro 90081 con lo cual, si lo vio en el gimnasio o bien tuvo que entrar a la misma hora que ella (o quizá poco antes o poco después).

1 / MURDER MYSTERY

A la misma hora que Annabel entró en miembro 48Z55. Empezamos por ese y cotejamos su ficha:

Código:

```
SELECT *  
FROM get_fit_now_member  
WHERE id = "48Z55"
```

id	person_id	name	membership_start_date	membership_status
48Z55	67318	Jeremy Bowers	20160101	gold

Tenemos pues, el nombre del primer sospechoso: Jeremy Bowers.

Cotejamos al segundo sospechoso con identificador de gimnasio 48Z7A:

Código:

```
SELECT *  
FROM get_fit_now_member  
WHERE id = "48Z7A"
```

id	person_id	name	membership_start_date	membership_status
48Z7A	28819	Joe Germuska	20160305	gold

1 / MURDER MYSTERY

Con todos estos datos recabados vamos a volver al principio y ver qué nos puede aportar el segundo testigo:

El que vive en la última casa de la calle Northwestern Dr.

Código:

```
SELECT *  
FROM person  
WHERE address_street_name LIKE "Northwestern Dr%"
```

id	name	license_id	address_number	address_street_name	ssn
10010	Muoi Cary	385336	741	Northwestern Dr	828638512
12711	Norman Apolito	667757	599	Northwestern Dr	778264744
14887	Morty Schapiro	118009	4919	Northwestern Dr	111564949
15171	Weldon Penso	336999	311	Northwestern Dr	131379495

La tabla es más extensa pero el número más alto en la dirección es el de Morty Schapiro.

Buscamos su ficha:

Código:

```
SELECT *  
FROM person  
WHERE name = "Morty Schapiro"
```

id	name	license_id	address_number	address_street_name	ssn
14887	Morty Schapiro	118009	4919	Northwestern Dr	111564949

Respuesta:

1 / MURDER MYSTERY

Y también su declaración:

Código:

```
SELECT *  
FROM interview  
WHERE person_id = 14887
```

person_id	transcript
14887	I heard a gunshot and then saw a man run out. He had a "Get Fit Now Gym" bag. The membership number on the bag started with "48Z". Only gold members have those bags. The man got into a car with a plate that included "H42W".

El asesino, según Morty, es miembro gold del gimnasio. El número de socio empieza por 48Z y conduce un coche matrícula H42W.

Empecemos por la matrícula:

Código:

```
SELECT *  
FROM drivers_license  
WHERE plate_number LIKE "%H42W%"
```

id	age	height	eye_color	hair_color	gender	plate_number	car_make	car_model
183779	21	65	blue	blonde	female	H42W0X	Toyota	Prius
423327	30	70	brown	brown	male	0H42W2	Chevrolet	Spark LS
664760	21	71	black	black	male	4H42WR	Nissan	Altima

1 / MURDER MYSTERY

Morty dijo que era un hombre así que o bien es el segundo o el tercero de la tabla.

Bien, tenía dos sospechosos: Jeremy Bowers (person_id = 67318) y Joe Germuska (person_id = 28819)

Veamos sus entrevistas:

Código:

```
SELECT *  
FROM interview  
WHERE person_id = 67318
```

person_id	transcript
67318	I was hired by a woman with a lot of money. I don't know her name but I know she's around 5'5" (65") or 5'7" (67"). She has red hair and she drives a Tesla Model S. I know that she attended the SQL Symphony Concert 3 times in December 2017.

Jeremy es el asesino, pero a sueldo.

Compruebo:

Código:

value
Congrats, you found the murderer! But wait, there's more... If you think you're up for a challenge, try querying the interview transcript of the murderer to find the real villain behind this crime. If you feel especially confident in your SQL skills, try to complete this final step with no more than 2 queries. Use this same INSERT statement with your new suspect to check your answer.

1 / MURDER MYSTERY

Correcto, es el asesino pero tenemos que buscar a la mujer pelirroja con mucho dinero que conduce un Tesla Model S y que fue al concierto sinfónico SQL tres veces en diciembre de 2017.

Buscamos en los datos de licencia de conducir que es en donde podemos afinar más:

Código:

```
SELECT *
```

```
FROM drivers_license
```

```
WHERE gender = "female" and car_make = "Tesla" and car_model =  
"Model S" and hair_color = "red"
```

id	age	height	eye_color	hair_color	gender	plate_number	car_make	car_model
202298	68	66	green	red	female	500123	Tesla	Model S
291182	65	66	blue	red	female	08CM64	Tesla	Model S
918773	48	65	black	red	female	917UU3	Tesla	Model S

Hay 3 mujeres que cuadran con la descripción.

Veamos quién son:

Código:

```
SELECT *
```

```
FROM person
```

```
WHERE license_id IN (202298,291182,918773)
```

id	name	license_id	address_number	address_street_name	ssn
78881	Red Korb	918773	107	Camerata Dr	961388910
90700	Regina George	291182	332	Maple Ave	337169072
99716	Miranda Priestly	202298	1883	Golden Ave	987756388

1/ MURDER MYSTERY

La que mejor está económicamente es Miranda Priestly.

Veamos ahora el evento del concierto a ver si podemos cruzar datos:

Código:

```
SELECT *
FROM facebook_event_checkin
WHERE event_name = "SQL Symphony Concert" and date LIKE
"201712%" and person_id = 99716
```

person_id	event_id	event_name	date
99716	1143	SQL Symphony Concert	20171206
99716	1143	SQL Symphony Concert	20171212
99716	1143	SQL Symphony Concert	20171229

Efectivamente, Miranda Priestly fue 3 veces al SQL Symphony Concert en 2017.

Comprobamos:

value
Congrats, you found the brains behind the murder! Everyone in SQL City hails you as the greatest SQL detective of all time. Time to break out the champagne!

Código:

```
INSERT INTO solution VALUES (1, 'Miranda Priestly');
```

```
SELECT value FROM solution;
```



CONCLUSIONES

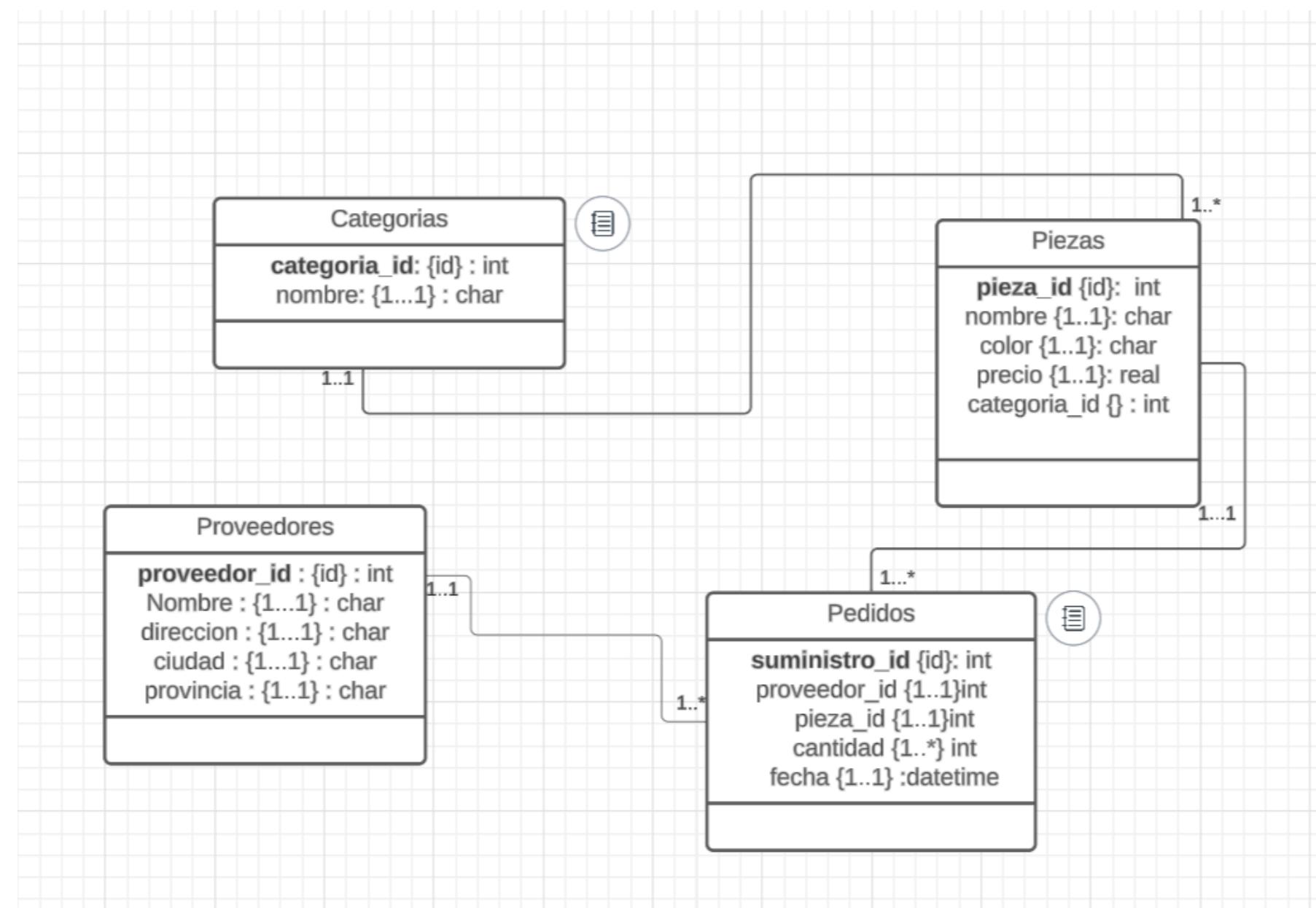
¿Qué aprendimos con SQL Murder Mystery?"

- Aplicamos SELECT, WHERE, LIKE, JOIN y subconsultas
- Analizamos datos relacionando tablas distintas
- Resolvimos el caso paso a paso gracias al modelo relacional
- Entendimos el valor de estructurar bien nuestras consultas

2/ DISEÑO DE BASE DE DATOS AVANZADA SOBRE PIEZAS DE AUTOMOCIÓN

2 / MODELO RELACIONAL DEL SISTEMA DE GESTIÓN

Para organizar la información de proveedores, piezas y pedidos, diseñamos un modelo relacional con 4 tablas principales conectadas por claves foráneas. Este modelo nos permite garantizar que los datos sean coherentes y realizar consultas mediante SQL.



2 / GENERACIÓN DE DATOS CON FAKER

Utilizamos la librería Faker en español para generar datos ficticios y realistas.

Generamos automáticamente:

- Proveedores con nombre, dirección, ciudad y provincia.
- Piezas con nombre, color, precio y categoría.
- Pedidos con fecha aleatoria, cantidad y pieza asignada a un proveedor.

Los datos se guardaron en archivos .csv y .json y luego se insertaron en la base de datos SQLite.

```
1 nombre,direccion,ciudad,provincia,codigo_proveedor
2 Banca Privada IB S.Coop., "Alameda de Salomé Perea 64, Navarra, 47769",Córdoba,Asturias,1978
3 Manufacturas IP S.L.U., "Vial Morena Lladó 69, Cuenca, 29294",Ciudad,Segovia,5868
4 Tecnologías Enriquez S.A.U., "Pasaje Amor Calvo 67, Palencia, 13800",Ciudad,Valencia,1546
5 Carrera & Asociados S.A.U., "Calle Aránzazu Cardona 11 Apt. 45 , Salamanca, 06026",Santa Cruz de Tenerife,Guipúzcoa,4637
6 Hermanos Carrasco S.L., "Rambla de Ricardo Pedrosa 82 Apt. 36 , Baleares, 06469",La Rioja,Cuenca,2431
7 Verónica Gibert Osorio S.Coop., "Vía Alfonso Lorenzo 56 Apt. 10 , La Rioja, 34590",Santa Cruz de Tenerife,Cantabria,8122
8 Rivera y Salazar S.L., "Callejón Jose Ramón Colomer 64, Huesca, 07336",Cuenca,Guadalajara,1663
9 Grupo Ríos S.A.T., "Callejón de Cruz Delgado 309, Madrid, 04394",Huelva,Vizcaya,6083
10 Eli Tomé Montes S.L.L., "Pasaje de Enrique Ocaña 6 Puerta 4 , Jaén, 50400",Cáceres,Sevilla,6457
11 Banca Privada QVA S.C.P., "Urbanización de Amaya Gabaldón 370, Ávila, 38670",Las Palmas,Sevilla,7125
12 Acuña y Perales S.L.L., "Acceso Iris Palacio 129 Piso 7 , Ávila, 50450",Córdoba,Navarra,3628
13 Construcción Catalá S.L.N.E, "Camino Rufino Ferrero 70, Girona, 03537",Vizcaya,Castellón,9152
14 Tecnologías Españolas S.Com., "Paseo Néstor Casanova 99 Apt. 64 , Guadalajara, 45887",Soria,Albacete,7573
15 Peiró & Asociados S.A., "Avenida de Maricela Vicens 3 Piso 2 , Soria, 03711",Ávila,Ourense,5558
16 Atienza y Rey S.A., "C. Arsenio Chacón 16, Ourense, 48879",La Rioja,Cantabria,7829
17 Samper y Llorens S.Com., "Cañada de Ester Quirós 1 Puerta 8 , Albacete, 19026",Baleares,Ourense,1341
18 Fábrica Farré y asociados S.A., "Acceso Joel Puerta 33, Soria, 48888",Asturias,Guadalajara,7935
19 Manufacturas Integrales S.A., "Camino de Ezequiel Amat 237 Piso 2 , Palencia, 21816",Toledo,Segovia,4800
20 Yuste y asociados S.A., "Calle de Enrique Verdugo 67, Albacete, 50038",Zaragoza,Albacete,4452
21 Alimentación Lobo y asociados S.Coop., "Rambla Amelia López 96, Castellón, 06986",Ávila,Girona,1337
22 Casas y Puente S.C.P., "Paseo de Moa Varela 70 Apt. 69 , Valencia, 38419",Huelva,Ávila,3683
23 Severo Correa Viñas S.L., "Rambla de Sancho Blasco 891 Puerta 6 , Girona, 48277",Cuenca,Toledo,7927
24 Hnos Gangallo S.Coop., "Vial de Encarnacion Viana 997, Badajoz, 14229",Murcia,Ourense,6058
25 Consultoría Españolas S.Com., "Callejón de Felicia Tena 95 Apt. 16 , Melilla, 44829",Barcelona,Jaén,7798
26 Comercializadora del Noroeste S.L.N.E, "Avenida de Bernardo Iborra 68 Apt. 19 , Lleida, 34896",Soria,Badajoz,1723
27 Despacho del Mediterráneo S.A., "Ronda Araceli Cepeda 73 Puerta 5 , Lleida, 47458",Segovia,Lugo,5809
```

2 / ¿QUÉ INFORMACIÓN RECOGE CADA TABLA?

📁 Categorías

- Contiene los diferentes tipos de piezas (Ej: Mecánica, Electrónica...).
- Cada pieza pertenece a una única categoría.
- Campos: ‘categoria_id’, ‘nombre’, ‘codigo_categoria’.

```
# Generar categorias
df_categorias = pd.DataFrame({
    "categoria_id": [2, 3, 4, 5, 6],
    "nombre": ["Mecánica", "Interiores", "Carrocería", "Neumaticos", "Accesorios"]
})

# Mostrar la tabla
print(df_categorias)

df_categorias.to_csv("categorias.csv", index=False)
```

	categoria_id	nombre
0	2	Mecánica
1	3	Interiores
2	4	Carrocería
3	5	Neumaticos
4	6	Accesorios

2 / ¿QUÉ INFORMACIÓN RECOGE CADA TABLA?

Proveedores

- Guarda la información de cada proveedor: ‘nombre’, ‘dirección’, ‘ciudad’ y ‘provincia’.
- Cada proveedor tiene un id único o (‘proveedor_id’).

```
df_proveedores.head()
```

	nombre	direccion	ciudad	provincia	codigo_proveedor
0	Banca Privada IB S.Coop.	Alameda de Salomé Pérez 64, Navarra, 47769	Córdoba	Asturias	1978
1	Manufacturas IP S.L.U.	Vial Morena Lladó 60, Cuenca, 29294	Ciudad	Segovia	6868
2	Tecnologías Enríquez S.A.U	Paseo Amor Calvo 67, Palencia, 13000	Ciudad	Valencia	1546
3	Carrera & Asociados S.A.U	Calle Aránzazu Cardona 11 Apt. 45 , Salamanca,...	Santa Cruz de Tenerife	Guipúzcoa	4637
4	Hermanos Carrasco S.L.	Rambla de Ricardo Pedrosa 82 Apt. 36 , Baleare...	La Rioja	Cuenca	2431

```
# Número de proveedores a generar
num_proveedores = 100

# Crear una lista de diccionarios con los datos de los proveedores
proveedores = []
for _ in range(num_proveedores):
    proveedor = {
        "nombre": fake.company(),
        "direccion": str(fake.address()).replace("\n", ", "),
        "ciudad": fake.city().replace("\n", ", "),
        "provincia": fake.state(),
        "codigo_proveedor": fake.random_int(min=1000, max=9999)
    }
    proveedores.append(proveedor)

# Convertir la lista en un DataFrame de Pandas
df_proveedores = pd.DataFrame(proveedores)

# Mostrar las primeras filas del DataFrame
print(df_proveedores.head())

# Guardar la base de datos en un archivo CSV
df_proveedores.to_csv("./archivos_csv/proveedores.csv", index=False)
```

2 / ¿QUÉ INFORMACIÓN RECOGE CADA TABLA?

Piezas

- Registra todas las piezas disponibles en el sistema.
- Contiene ‘nombre’, ‘color’, ‘precio’ y ‘categoría’.
- Se relaciona con la tabla ‘categorías’ a través de ‘categoria_id’

```
# Generar piezas
colores = ["rojo", "azul", "verde", "negro", "blanco"]
categorias = [1, 2, 3, 4, 5, 6] # ID de categorías ya insertadas
nombres_piezas = [
    "Filtro de aceite", "Amortiguador", "Radiador", "Bujía", "Pastilla de freno",
    "Alternador", "Motor de arranque", "Batería", "Disco de freno", "Inyector",
    "Caja de cambios", "Escape", "Compresor", "Espejo retrovisor", "Ventilación",
    "Faros delanteros", "Asientos", "Embrague", "Volante", "Neumáticos de coche",
    "Neumático de 4 estaciones", "Alfombrillas", "Espejos", "Parabrisas", "Llantas"]

# Generar piezas
piezas = []
for i in range(100):
    piezas.append({
        "codigo_pieza": fake.random_int(min=100, max=1000),
        "nombre": random.choice(nombres_piezas),
        "color": random.choice(colores),
        "precio": round(random.uniform(10, 500), 2),
        "categoria_id": random.choice(categorias)
    })
df_piezas = pd.DataFrame(piezas)

print(df_piezas.head())

df_piezas.to_csv("./archivos_csv/piezas.csv", index=False)
```

codigo_pieza	nombre	color	precio	categoria_id
757	Escape	negro	154.83	2
100	Asientos	azul	212.82	5

2 / GENERADOR AUTOMÁTICO DE PEDIDOS

```
from datetime import timedelta
fake = Faker("es_ES")

# Generar pedidos
num_pedidos = 200
pedidos = []
pieza_ya_reservada = {}

for _ in range(num_pedidos):

    proveedor = fake.random_int(min=1000, max=9999)
    pieza = fake.random_int(min=100, max=1000)
    cantidad = random.randint(1, 10)
    fecha = fake.date_between(start_date='-1y', end_date='today')

    proveedor_pieza = (proveedor, pieza)

    if proveedor_pieza in pieza_ya_reservada:
        while fecha in pieza_ya_reservada[proveedor_pieza]:
            fecha += timedelta(days=1)
        pieza_ya_reservada[proveedor_pieza].add(fecha)
        print(f'Tu pieza {pieza} se ha reservado del proveedor {proveedor} para el dia {fecha}')
    else:
        pieza_ya_reservada[proveedor_pieza] = {fecha}

    pedidos.append({
        "proveedor_id": proveedor,
        "pieza_id": pieza,
        "cantidad": cantidad,
        "fecha": fecha.isoformat()
    })

df_pedidos = pd.DataFrame(pedidos)

print(df_pedidos.head())
df_pedidos.to_csv("./archivos_csv/pedidos.csv", index=False)
```

Generamos pedidos de manera automática combinando piezas y proveedores aleatorios.

Para cada pedido, se definió:

- Un proveedor válido
- Una pieza disponible
- Una cantidad entre 1 y 10
- Una fecha aleatoria en el último año

Los datos generados se almacenaron en un DataFrame y luego se insertaron en la base de datos SQLite.

Este sistema nos permitió llenar la base de datos con un histórico realista de suministros para poder realizar consultas SQL.

2 / BASE DE DATOS Y CONSULTAS SQL

```
import sqlite3

connection = sqlite3.connect("base_de_datos.db")
cursor = connection.cursor()

cursor.execute("SELECT name FROM sqlite_master WHERE type='table'")
cursor.fetchall()

[('Proveedores',),
 ('Categorias',),
 ('Pedidos',),
 ('sqlite_sequence',),
 ('Piezas',)]
```



```
cursor.execute("""SELECT
    p.fecha,
    pr.nombre AS proveedor,
    pi.nombre AS pieza,
    pi.precio,
    p.cantidad
FROM pedidos p
JOIN proveedores pr ON p.proveedor_id = pr.proveedor_id
JOIN piezas pi ON p.pieza_id = pi.pieza_id
ORDER BY p.fecha DESC""")
cursor.fetchall()
```



```
[('2024-09-30', 'Olivera y Soria S.A.T.', 'Pastilla de freno', 192.81, 6)]
```

Creamos una base de datos relacional en SQLite con las tablas: categorias, proveedores, piezas y pedidos.

Insertamos datos generados automáticamente desde pandas y Faker con .to_sql(...).

Realizamos consultas SQL para:

- Visualizar relaciones entre piezas, proveedores y pedidos
- Comprobar estructura del modelo relacional

```
cursor.execute("SELECT * FROM Proveedores")
cursor.fetchall()
```

```
[(1, 'Proveedor A', 'Calle Falsa 123', 'CDMX', 'CDMX'),
(1116,
 'Instalaciones Gallo & Asociados S.Coop.',
 'Via Xiomara Quintero 425, Guadalajara, 64000',
 'Tarragona',
 'Ourense'),
(1286,
 'Egea y Herrero S.A.',
 'Pasadizo de Mariano Planas 862, Navarra, 03490',
 'Almeria',
 'Jaén'),
(1296,
 'Consultoría Globales S.L.N.E',
 'Vial de Salvador Cervantes 131, Soria, 11633',
 'Ceuta',
 'Almería'),
```

CONCLUSIONES

- Creamos un sistema funcional y escalable para gestionar piezas, proveedores y pedidos.
- La base de datos quedó estructurada y llena de datos realistas gracias a Faker.
- El sistema es consultable y ampliable con nuevas funcionalidades.
- Este proyecto nos ha permitido aplicar SQL, Python, lógica relacional y trabajo en equipo.

MUCHAS GRACIAS

SHEILA RODRÍGUEZ
SANDRA LÓPEZ
DAVID TOROSYAN

```
render() {
  return (
    <React.Fragment>
      <div className="py-5">
        <div className="container">
          <Title name="our" title="product">
            <div className="row">
              <ProductConsumer>
                {(value) => {
                  |   |   |   console.log(value)
                }}
              </ProductConsumer>
            </div>
          </div>
        </div>
      </React.Fragment>
```