

Gestión de bases de datos

Nombre: César Augusto Carchi Ludeña

Tarea 1: Despliegue y gestión de bases de datos relacionales

1. Instala MySQL server en un entorno local. Se recomienda el uso de un cliente de virtualización como VMware o VirtualBox y de un sistema operativo basado en Linux como Ubuntu.

```
sudo apt update
sudo apt get install
sudo apt install mysql-server
sudo systemctl status mysql
sudo mysql_secure_installation
```

Nota: En caso de que no permita cambiar root, entramos a mysql (**sudo mysql**) y ejecutamos el siguiente comando:

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password by
'mynewpassword';
```

2. Descarga e instala **Sakila**. Adjunta una captura de pantalla del resultado de ejecutar las siguientes consultas SQL:

Abrimos una terminal, nos posicionamos en donde tenemos los archivos de sakila y ejecutamos mysql:

```
mysql -u root -p      --ingresamos contraseña de root
source sakila-schema.sql; --esto para ejecutar el esquema y crear las
tablas.
source sakila-data.sql; -- esto para cargar los datos.
```

a. Accedemos a Sakila:

```
mysql> USE sakila;
```

```
sheshar@dvlpr: ~/Documentos/sakila-db
sheshar@dvlpr:~/Documentos/sakila-db$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 17
Server version: 8.0.29-0ubuntu0.22.04.2 (Ubuntu)

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use sakila;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> s
```

b. Mostrar tablas de Sakila:

```
mysql> SHOW FULL TABLES;
```

```
sheshar@dvlpr: ~/Documentos/sakila-db
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use sakila;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SHOW FULL TABLES;
+-----+-----+
| Tables_in_sakila | Table_type |
+-----+-----+
| actor             | BASE TABLE |
| actor_info        | VIEW        |
| address           | BASE TABLE |
| category          | BASE TABLE |
| city              | BASE TABLE |
| country           | BASE TABLE |
| customer           | BASE TABLE |
| customer_list     | VIEW        |
| film              | BASE TABLE |
| film_actor        | BASE TABLE |
| film_category     | BASE TABLE |
| film_list         | VIEW        |
| film_text         | BASE TABLE |
| inventory          | BASE TABLE |
| language          | BASE TABLE |
| nicer_but_slower_film_list | VIEW        |
| payment           | BASE TABLE |
| rental            | BASE TABLE |
| sales_by_film_category | VIEW        |
| sales_by_store    | VIEW        |
| staff             | BASE TABLE |
| staff_list        | VIEW        |
| store             | BASE TABLE |
+-----+-----+
23 rows in set (0,00 sec)

mysql>
```

c. Mostrar número de registros de la tabla "film":

```
mysql> SELECT COUNT(*) FROM film;
```

```
sheshar@dvlpr: ~/Documentos/sakila-db

mysql> SHOW FULL TABLES;
+-----+-----+
| Tables_in_sakila | Table_type |
+-----+-----+
| actor             | BASE TABLE |
| actor_info        | VIEW        |
| address           | BASE TABLE |
| category          | BASE TABLE |
| city              | BASE TABLE |
| country           | BASE TABLE |
| customer          | BASE TABLE |
| customer_list     | VIEW        |
| film              | BASE TABLE |
| film_actor        | BASE TABLE |
| film_category     | BASE TABLE |
| film_list         | VIEW        |
| film_text         | BASE TABLE |
| inventory         | BASE TABLE |
| language          | BASE TABLE |
| nicer_but_slower_film_list | VIEW        |
| payment           | BASE TABLE |
| rental            | BASE TABLE |
| sales_by_film_category | VIEW        |
| sales_by_store    | VIEW        |
| staff             | BASE TABLE |
| staff_list        | VIEW        |
| store             | BASE TABLE |
+-----+-----+
23 rows in set (0,00 sec)

mysql> SELECT COUNT(*) FROM film;
+-----+
| COUNT(*) |
+-----+
|      1000 |
+-----+
1 row in set (0,02 sec)

mysql>
```

d. Mostrar número de registros de la tabla "film_text":

```
mysql> SELECT COUNT(*) FROM film_text;
```

```
sheshar@dvlpr: ~/Documentos/sakila-db

| category          | BASE TABLE |
| city              | BASE TABLE |
| country           | BASE TABLE |
| customer          | BASE TABLE |
| customer_list     | VIEW        |
| film              | BASE TABLE |
| film_actor        | BASE TABLE |
| film_category     | BASE TABLE |
| film_list         | VIEW        |
| film_text         | BASE TABLE |
| inventory         | BASE TABLE |
| language          | BASE TABLE |
| nicer_but_slower_film_list | VIEW        |
| payment           | BASE TABLE |
| rental            | BASE TABLE |
| sales_by_film_category | VIEW        |
| sales_by_store    | VIEW        |
| staff             | BASE TABLE |
| staff_list        | VIEW        |
| store             | BASE TABLE |
+-----+-----+
23 rows in set (0,00 sec)

mysql> SELECT COUNT(*) FROM film;
+-----+
| COUNT(*) |
+-----+
|      1000 |
+-----+
1 row in set (0,02 sec)

mysql> SELECT COUNT(*) FROM film_text;
+-----+
| COUNT(*) |
+-----+
|      1000 |
+-----+
1 row in set (0,00 sec)

mysql>
```

3. MySQL Workbench es una herramienta visual para arquitectos de bases de datos, desarrolladores y DBAs. MySQL Workbench proporciona modelado de datos, desarrollo de SQL y herramientas de administración completas para la configuración del servidor de base de datos, la administración de usuarios, las copias de seguridad y mucho más. Descarga e instala MySQL Workbench y visualiza la base de datos Sakila. Adjunta una captura de pantalla de las siguientes consultas realizadas en MySQL Workbench:

- **Descargar workbench:**

```
sudo snap install mysql-workbench-community
```

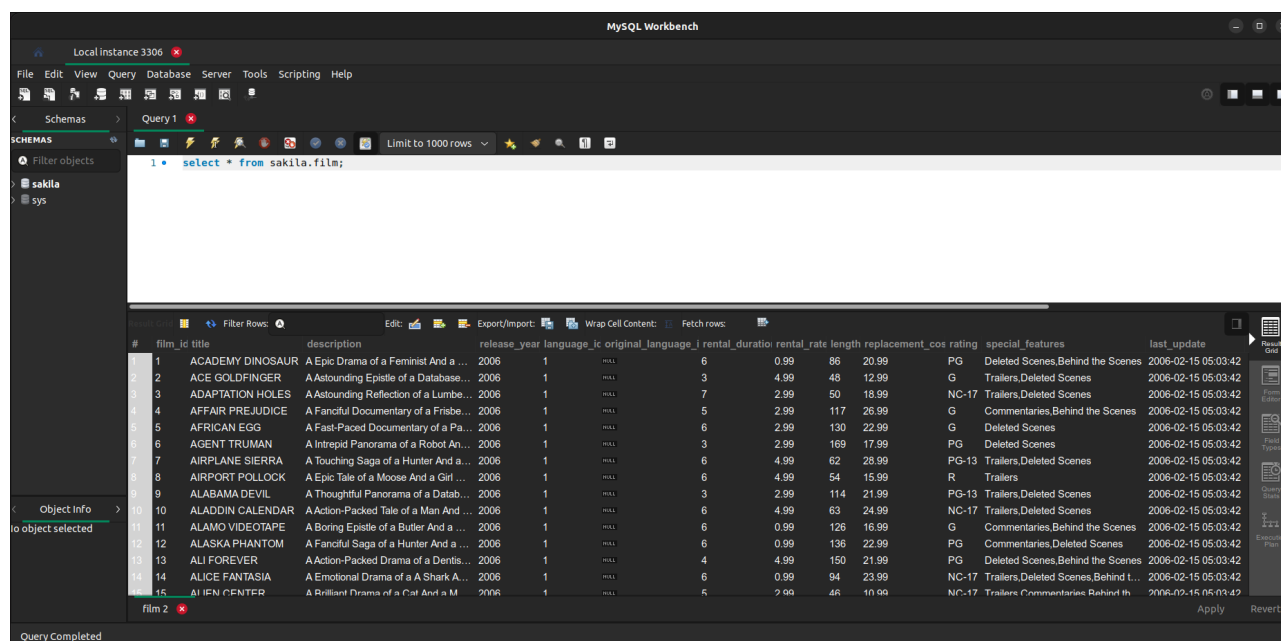
- **Configurar conexión ssh y password manager para workbench:**

```
snap connect mysql-workbench-community:password-manager-service
```

```
snap connect mysql-workbench-community:ssh-keys
```

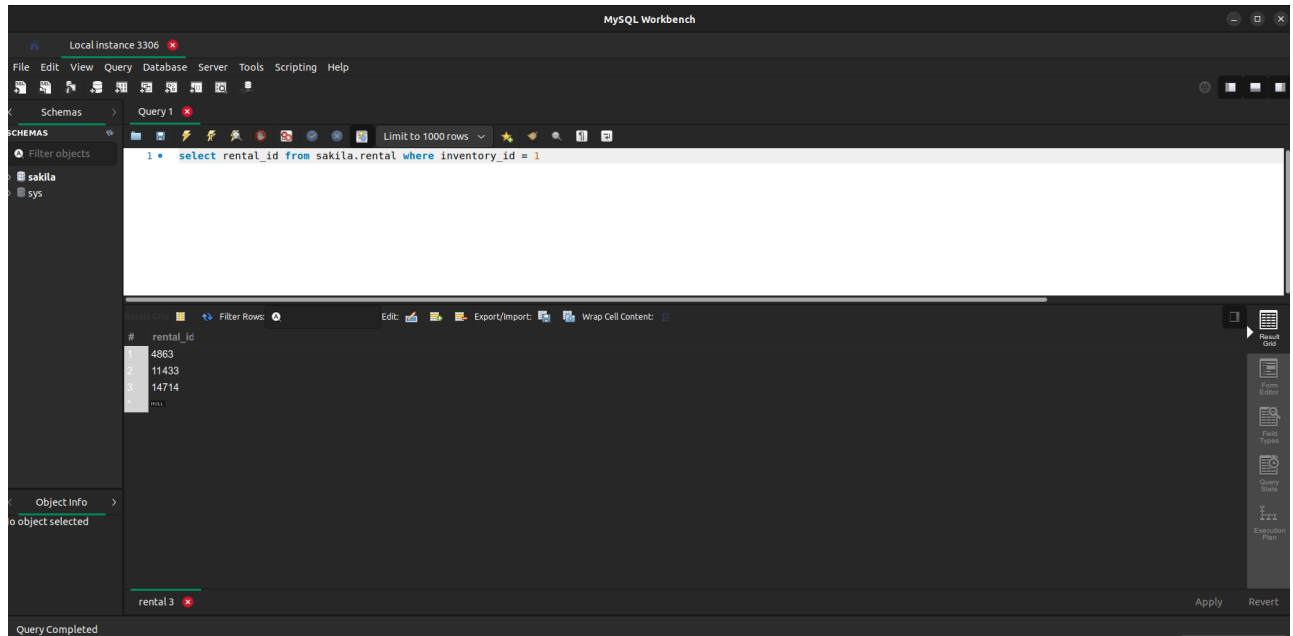
- **Ingresar a workbench e ingresar consultas:**

a. `select * from sakila.film;`



#	film_id	title	description	release_year	language_id	original_language_id	rental_duration	rental_rate	length	replacement_cost	rating	special_features	last_update
1	1	ACADEMY DINOSAUR	A Epic Drama of a Feminist And a ...	2006	1	en_US	6	0.99	86	20.99	PG	Deleted Scenes, Behind the Scenes	2006-02-15 05:03:42
2	2	ACE GOLDFINGER	A Astounding Epistle of a Database...	2006	1	en_US	3	4.99	48	12.99	G	Trailers, Deleted Scenes	2006-02-15 05:03:42
3	3	ADAPTATION HOLES	A Astounding Reflection of a Lumbe...	2006	1	en_US	7	2.99	50	18.99	NC-17	Trailers, Deleted Scenes	2006-02-15 05:03:42
4	4	AFFAIR PREJUDICE	A Fanciful Documentary of a Frisbe...	2006	1	en_US	5	2.99	117	26.99	G	Commentaries, Behind the Scenes	2006-02-15 05:03:42
5	5	AFRICAN EGG	A Fast-Paced Documentary of a Pa...	2006	1	en_US	6	2.99	130	22.99	G	Deleted Scenes	2006-02-15 05:03:42
6	6	AGENT TRUMAN	A Intrepid Panorama of a Robot An...	2006	1	en_US	3	2.99	169	17.99	PG	Deleted Scenes	2006-02-15 05:03:42
7	7	AIRPLANE SIERRA	A Touching Saga of a Hunter And a ...	2006	1	en_US	6	4.99	62	28.99	PG-13	Trailers, Deleted Scenes	2006-02-15 05:03:42
8	8	AIRPORT POLLOCK	A Epic Tale of a Moose And a Girl...	2006	1	en_US	6	4.99	54	15.99	R	Trailers	2006-02-15 05:03:42
9	9	ALABAMA DEVIL	A Thoughtful Panorama of a Datab...	2006	1	en_US	3	2.99	114	21.99	PG-13	Trailers, Deleted Scenes	2006-02-15 05:03:42
10	10	ALADDIN CALENDAR	A Action-Packed Tale of a Man And...	2006	1	en_US	6	4.99	63	24.99	NC-17	Trailers, Deleted Scenes	2006-02-15 05:03:42
11	11	ALAMO VIDEOTAPE	A Boring Epistle of a Butler And a ...	2006	1	en_US	6	0.99	126	16.99	G	Commentaries, Behind the Scenes	2006-02-15 05:03:42
12	12	ALASKA PHANTOM	A Fanciful Saga of a Hunter And a ...	2006	1	en_US	6	0.99	136	22.99	PG	Commentaries, Deleted Scenes	2006-02-15 05:03:42
13	13	ALI FOREVER	A Action-Packed Drama of a Dentis...	2006	1	en_US	4	4.99	150	21.99	PG	Deleted Scenes, Behind the Scenes	2006-02-15 05:03:42
14	14	ALICE FANTASIA	A Emotional Drama of a A Shark A...	2006	1	en_US	6	0.99	94	23.99	NC-17	Trailers, Deleted Scenes, Behind L...	2006-02-15 05:03:42
15	15	ALIEN CENTER	A Brilliant Drama of a Cat And a M...	2006	1	en_US	5	2.99	56	10.99	NC-17	Trailers, Commentaries, Behind th...	2006-02-15 05:03:42

b. `select rental_id from sakila.rental where inventory_id = 1;`



4. Hasta ahora hemos visto algunas formas de acceder a la base de datos de manera manual. En esta sección vamos a explorar el concepto de [ORM](#). **Implementa un breve programa que acceda a la base de datos Sakila y realice la siguiente consulta utilizando una librería que aplique el concepto de ORM. Adjunta una captura de pantalla con el resultado de la ejecución del programa. a. `SELECT country_id, country from sakila.country;`**

ORM a utilizar (Prisma)

Prisma es un ORM de código abierto y consta de las siguientes partes:

1. Prisma Client: es un generador de consultas automáticas para Node y Typescript.
2. Prisma Migrate: es el sistema de migración.
3. Prisma Studio: es la interfaz gráfica del usuario para editar gráficamente los datos

• Instalar prisma.

- Creamos una carpeta para iniciar un proyecto con prisma:

```
> mkdir prismaORM
```

- Iniciamos node:

```
> npm init -y
```

- Instalamos prisma:

```
> npm install prisma --save -dev
```

- Para utilizar el cliente de prisma ejecutamos:

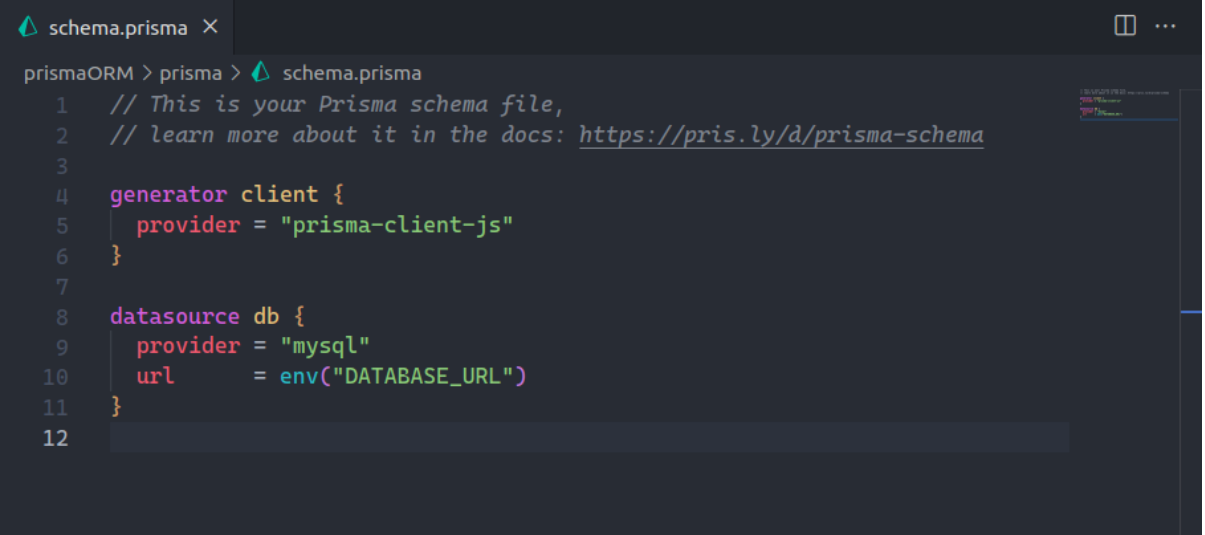
```
> npx prisma
```

```
> npx prisma init
```

Nota: El comando "NPX PRISMA INIT" lo que hace es crear las variables de entorno (.env), adicionalmente crea la carpeta prisma y el archivo schema.prisma

- **Conectar Prisma con Sakila.**

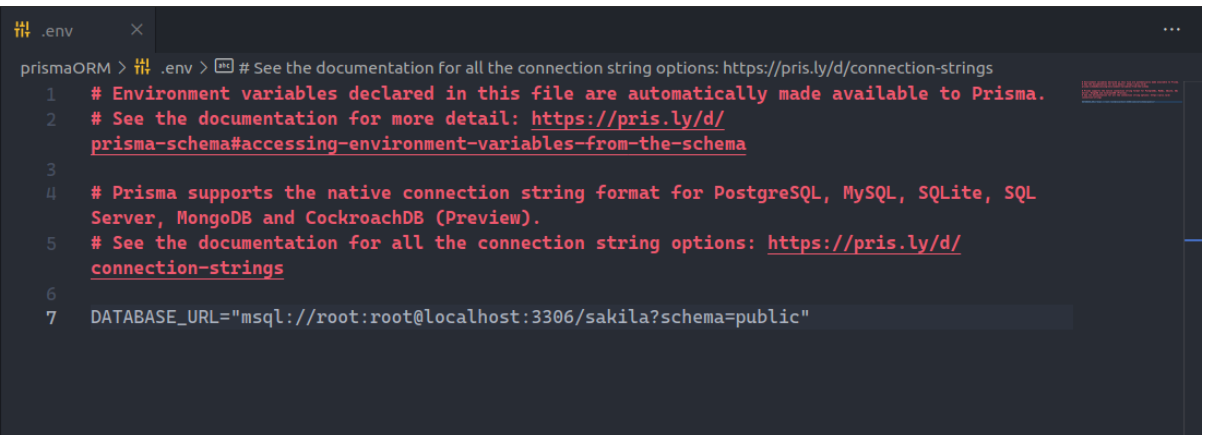
- Para conectar con la base de datos "Sakila". Accedemos al archivo `./prisma/schema.prisma`. Por defecto viene provider "postgres", se lo cambia por "mysql".



```
schema.prisma x
prismaORM > prisma > schema.prisma
1 // This is your Prisma schema file,
2 // learn more about it in the docs: https://pris.ly/d/prisma-schema
3
4 generator client {
5   provider = "prisma-client-js"
6 }
7
8 datasource db {
9   provider = "mysql"
10  url      = env("DATABASE_URL")
11 }
12
```

- Seguido, se configura las variables de entorno (el archivo ".env") y ahí ponemos los datos de nuestra base de datos:

```
DATABASE_URL="mysql://root:root@localhost:3306/sakila?
schema=public"
```



```
.env x
prismaORM > .env # See the documentation for all the connection string options: https://pris.ly/d/connection-strings
1 # Environment variables declared in this file are automatically made available to Prisma.
2 # See the documentation for more detail: https://pris.ly/d/prisma-schema#accessing-environment-variables-from-the-schema
3
4 # Prisma supports the native connection string format for PostgreSQL, MySQL, SQLite, SQL
5 # See the documentation for all the connection string options: https://pris.ly/d/connection-strings
6
7 DATABASE_URL="mysql://root:root@localhost:3306/sakila?schema=public"
```

- Ahora se procede a crear los modelos en el archivo "schema.prisma". Para este ejercicio describimos el modelo "country" el cual representa la tabla "country" de la base de datos Sakila:

```
model country { // nombre del modelo
  country_id Int @id @default(autoincrement()) // id de la tabla
  country String // atributo nombre de tipo string
  last_update DateTime // atributo fecha de actualización de tipo
```

```
DateTime
}
```

- Para que prisma migre todos los modelos creados, se ejecuta el siguiente comando:

```
prisma migrate
```

- Una vez configuradas las variables de entorno y creado los modelos, se procede a instalar prisma client.

```
npm install @prisma/client
```

- **Crear app para consumir la base de datos con el ORM Prisma.**

- En la raíz del proyecto creamos "app.js". En este archivo ingresaremos las consultas a la base de datos sakila utilizando el ORM Prisma.

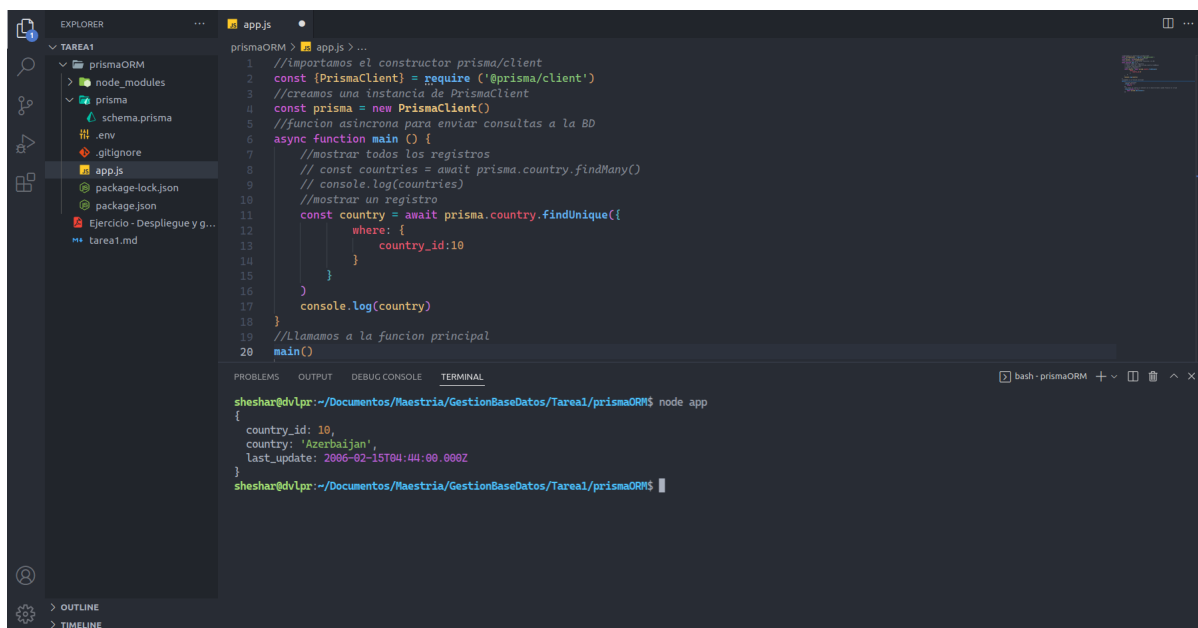
```
touch app.js
```

- Dentro del app.js ingresamos el siguiente código:

```
//importamos el constructor prisma/client
const {PrismaClient} = require ('@prisma/client')
//creamos una instancia de PrismaClient
const prisma = new PrismaClient()
//funcion asincrona para enviar consultas a la BD
async function main () {
  //mostrar todos los registros
  //const countries = await prisma.country.findMany()
  //console.log(countries)
  //mostrar un registro
  const country = await prisma.country.findUnique ({
    where: {
      country_id:10
    }
  })
  console.log(country)
}
//Llamamos a la funcion principal
main()
//errores posibles
.catch( (e) => {
  throw e
})
//al final se cierra la conexiòn con la base de datos cuando
finalice el script
.finally(async () => {
  await prisma.$disconnect()
})
```

- Finalmente para ejecutar el programa ponemos en la consola el siguiente comando:

node app



```
1 //importamos el constructor prisma/client
2 const {PrismaClient} = require ('@prisma/client')
3 //creamos una instancia de PrismaClient
4 const prisma = new PrismaClient()
5 //funcion asincrona para enviar consultas a la BD
6 async function main () {
7   //mostrar todos los registros
8   // const countries = await prisma.country.findMany()
9   // console.log(countries)
10  //mostrar un registro
11  const country = await prisma.country.findUnique({
12    where: {
13      country_id:10
14    }
15  })
16  console.log(country)
17 }
18 //Llamamos a la funcion principal
19 main()
20
```

```
sheshar@dvlp:~/Documentos/Maestria/GestionBaseDatos/Tareal/prismaORM$ node app
{
  country_id: 10,
  country: 'Azerbaijan',
  last_update: 2006-02-15T04:44:00.000Z
}
sheshar@dvlp:~/Documentos/Maestria/GestionBaseDatos/Tareal/prismaORM$
```

Se puede observar como se obtiene la información del país con id "10".