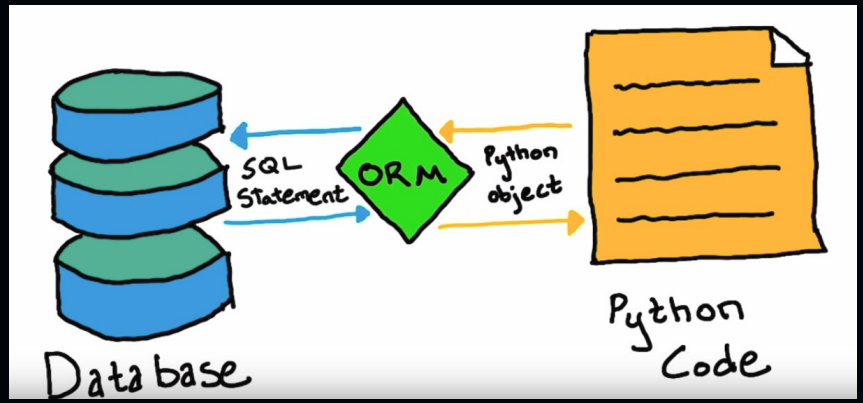
Clase N° 13: NextJS - Prisma

Crombie

Qué es Prisma?

- ➔ Prisma es un ORM (Object-Relational mapper).
- ➔ Simplifica la interacción con las bases de datos, trabajando con ellas a partir de esquemas del código.
- ➔ Compatible con múltiples bases de datos.



Crombie

Qué es un ORM?

- ➔ Son herramientas que nos permiten interactuar con bases de datos utilizando objetos en vez de SQL.
- ➔ Traduce consultas de SQL complejas a métodos y propiedades de objetos en el lenguaje que estemos usando.
- ➔ Ventajas:
  - ◆ Reducción de errores humanos al escribir SQL.
  - ◆ Código más legible y reutilizable.
  - ◆ Abstracción del motor de bases de datos subyacente.

```
SQL Query:
SELECT * FROM users WHERE id = 1;

(Prisma):
const user = await prisma.user.findUnique({ where: { id: 1 } });
```

Por qué Prisma?

- ➔ Tipado automático: Prisma genera un cliente con tipos estrictos basado en tu esquema.
- ➔ Fácil de usar: Sintaxis intuitiva y soporte para TypeScript.
- ➔ Abstracción poderosa:
  - ◆ Migraciones de esquema gestionadas automáticamente.
  - ◆ Soporte para relaciones complejas entre tablas.
- ➔ Compatible con Next.js:
  - ◆ Integra perfectamente con funciones de servidor (/api) y getServerSideProps.

Prisma Schema

- Prisma utiliza un esquema para definir el modelo de la base de datos.
- Incluye:
  - Definición de los modelos.
  - Relaciones entre tablas.
  - Configuración de la base de datos.

```
model User {
  id        Int      @id @default(autoincrement())
  name      String
  email     String   @unique
  posts     Post[]
}

model Post {
  id        Int      @id @default(autoincrement())
  title     String
  content   String
  authorId  Int
  author    User     @relation(fields: [authorId], references: [id])
}
```

Crombie

Prisma Client

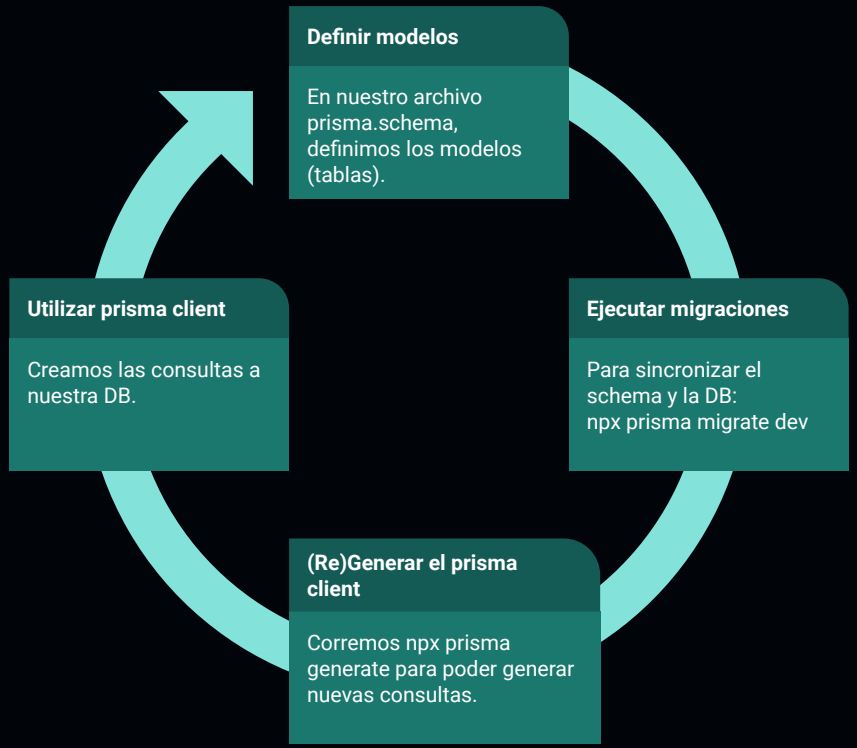
- ➔ Prisma Client es una API auto-generada para interactuar con tu base de datos.
- ➔ Funciona con promesas y soporta métodos CRUD.
- ➔ Nos permite hacer consultas a nuestra base de datos de una manera mucho más simple.

```
// Crear un usuario
const newUser = await prisma.user.create({
  data: {
    name: "Alice",
    email: "alice@example.com",
  },
});

// Leer usuarios
const users = await prisma.user.findMany();

// Actualizar un usuario
const updatedUser = await prisma.user.update({
  where: { id: 1 },
  data: { name: "Alice Updated" },
});

// Eliminar un usuario
await prisma.user.delete({ where: { id: 1 } });
```

Comandos que utilizaremos

- ➔ npm install prisma --save-dev
- ➔ npx prisma init
- ➔ npx prisma migrate dev --name init

Flujo de trabajo con Prisma



Crombie

Momento de práctica:

- ➔ Desarrollar un modelo básico de una base de datos para un e-commerce utilizando Prisma.
- ➔ Implementar consultas CRUD.
- ➔ Definir dos tablas:
  - ◆ Product:
    - id
    - name
    - description
    - price
    - stock
    - categoryId
    - category
    - createdAt
    - updatedAt
  - ◆ Category:
    - id
    - name
    - products

Crombie