```
tags:
- javascript
- typescript
- node
- express
- orm
```

Prisma

Prisma est un ORM *nouvelle génération* permettant de communiquer avec une base de données.

Installation

```
npm install -D prisma
```

Initialisation

```
npx prisma init
```

Cette commande crée un répertoire `prisma` avec un fichier `schema.prisma` ainsi que le fichier `.env` s'il n'existe pas.

Connexion à la base de données (MySQL)

Les données relatives à la connexion à la base de données sont placées dans le fichier `.env` sous la forme d'une URL :

```
DATABASE_URL="mysql://root:randompassword@localhost:3306/mydb"
```

Cette URL apparait dans le fichier `prisma/schema.prisma` :

```
generator client {
        provider = "prisma-client-js"
        output = "./output"
}

datasource db {
        provider = "mysql"
        url = env("DATABASE_URL")
}
```

Dans le cas d'une base de données SQLite, ce fichier se présente de la forme :

```
datasource db {
        provider = "sqlite"
        url = "file:./dev.db"
}
```

Attention aux types des données stockées, elles ne sont pas les mêmes entre une base de données traditionnelles comme MySQL et SQLite. Il est plus intéressant de configurer une connexion avec une vraie base de données à partir d'un moment où l'application s'inscrit dans un véritable processus de développement.

Introspection

Si la base de données existe déjà, il est possible d'en faire une introspection avec

```
npx prisma db pull
```

Cela peuplera le fichier `prisma/schema.prisma`.

Créer un modèle

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

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

Après avoir créé/modifié un modèle, il faut appliquer ces modifications à la base de données avec la commande :

```
npx prisma migrate dev --name initial-migration
```

One-to-One Relation

```
model User {
        id      Int      @id     @default(autoincrement())
        email   String   @unique
        profile Profile?
}

model Profile {
        name   String
        gender String
        age    String

        userId Int  @unique
        user   User @relation(fields: [userId], references: [id])
}
```

```
...
prisma.user.findMany({
        include: {
                profile: true
        }
});
```

One-to-Many Relation

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

model Post {
        title String

        userId Int  @unique
        user   User @relation(fields: [userId], references: [id])
}
```

```
...
prisma.user.findMany({
        include: {
                posts: true
        }
});
```

Implicit Many-to-Many Relation

```
model Post {
        id    Int    @id     @default(autoincrement())
        title String
        tags Tag[]
}

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

```
...
prisma.post.findMany({
        include: {
                tags: true
        }
});
```

Explicit Many-to-Many Relation

```
model Post {
        id    Int    @id     @default(autoincrement())
        title String
        tags PostTag[]
}

model Tag {
        id    Int    @id     @default(autoincrement())
        name  String
        posts PostTag[]
}

model PostTag {
        postId Int
        post Post @relation(fields: [postId], references: [id])

        tagId Int
        tag Tag @relation(fields: [tagId], references: [id])

        @@id([postId, tagId])
}
```

```
...
prisma.post.findMany({
        include: {
                tags: true
        }
});
```

Manipuler les données avec Node.js

```
import { PrismaClient, Prisma } from "@prisma/client";

...
```

Interagir avec la base de données : requêtes

A partir d'une application Express, on accède à la base de données avec Prisma et on effectue différentes actions en fonction du chemin d'accès.

```
// server.ts
import express, { Express, Request, Response } from "express";
import { PrismaClient, Prisma } from "@prisma/client";

const app: Express = express();
const prisma: Prisma = new PrismaClient();

app.get('/', async (req: Request, res: Response) => {
        const users: Prisma.UserSelect = await prisma.user.findMany({
                where: { admin: true },
                include: { posts: true }
        });
        res.json(users);
});

app.post('/post', async (req: Request, res: Response) => {
        const { title, content, authorEmail } = req.body;
        const post = await prisma.post.create({
                data: {
                        title,
                        content,
                        author: { connect: { email: authorEmail } }
                }
        });
        res.json(post);
});

app.put('/publish/:id', async (req: Request, res: Response) => {
        const { id } = req.params;
        const post = await prisma.post.update({
                where: { id },
                data: { published: true }
        });
        res.json(post);
});

app.delete('/user/:id', async (req: Request, res: Response) => {
```

```
        const { id } = req.params;
        const user = await prisma.user.delete({
                where: {
                        id
                }
        });
        res.json(user);
});

app.listen(3000);
```