4 ОКТЯБРЯ 2022 • • ТЕХНОЛОГИЮ, ,

Как настроить Prisma с Next.js и Postgres

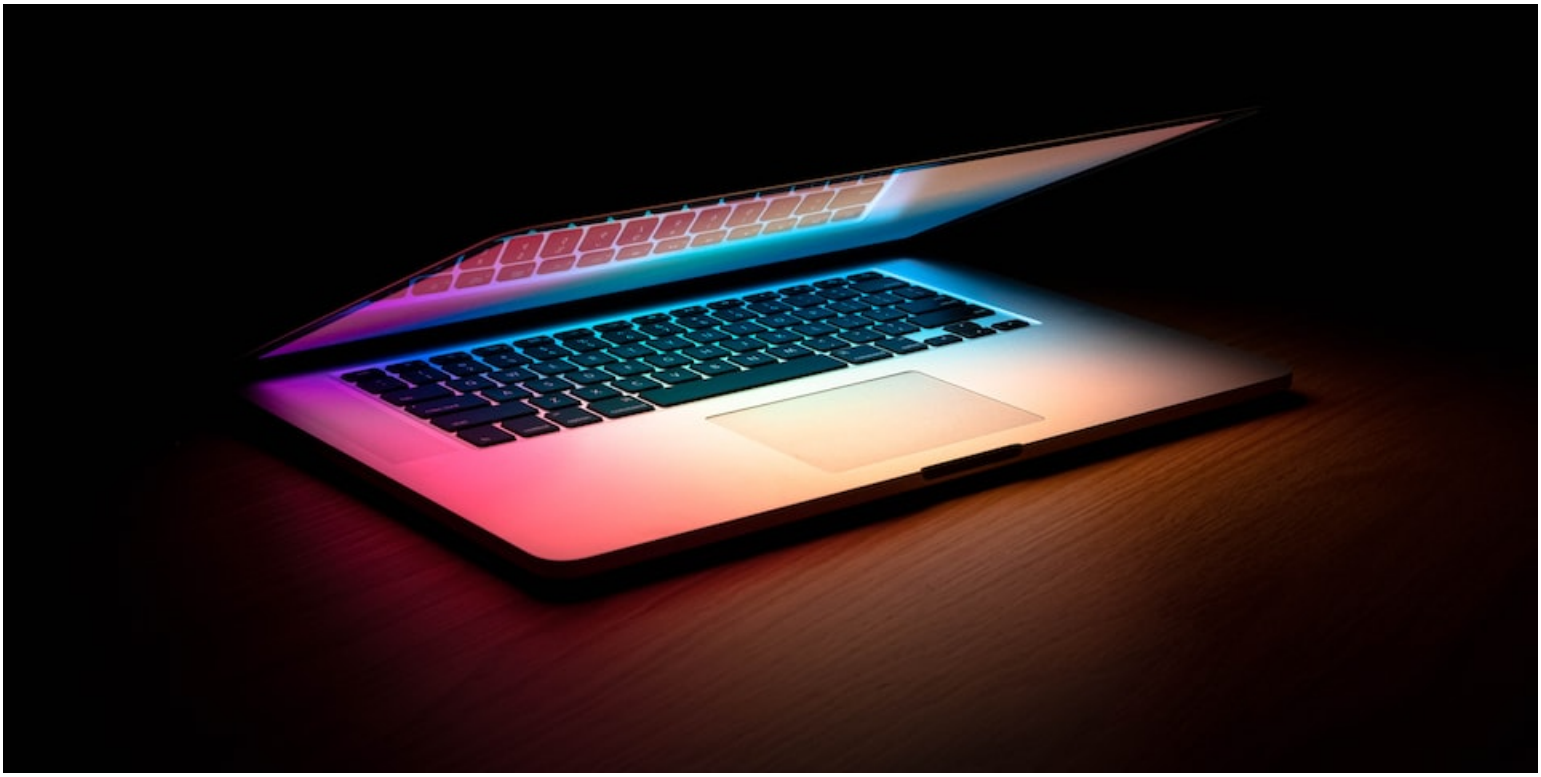

Фото Джошуа Реддекоппа / Unsplash

How to set up Prisma with Next.js and Postgres!

Быстрый переход

- Официальный Next.js + Пример Prisma
- Официальная настройка Prisma
- Docker Postgres
- Полный пример кода на GitHub

При создании приложения SaaS вам нужно будет быстро взаимодействовать с базой данных. Вы можете либо использовать уровень абстракции ORM, либо писать SQL-запросы и выполнять их. Лучшие ORM позволят вам быстро писать SQL там, где вам нужно, удаляя шаблонный код.

ORM - это объектно-реляционный уровень отображения. Он берет данные из базы данных и преобразует их в легко управляемые объекты для программиста. Я нашел

тонкий, простой слой абстракции, который лучше всего подходит для производительности и доставки кода.

В настоящее время одной из лучших ORM для TypeScript является Prisma. TypeScript - лучший язык для написания веб-приложений, но ORM исторически были сложными. В прошлом ORM на JavaScript использовали для работы много динамического программирования. И типами с такой черной магией трудно управлять.

Prisma использует другой подход, когда она определяет схему на своем независимом языке. С одной стороны, это еще кое-что, чему стоит научиться. С другой стороны, он идеально генерирует типы для вашей конкретной модели. TypeScript работает так же хорошо, как и types, так что это компромисс, который я нахожу приемлемым.

Запустите Postgres Container

Postgres - моя предпочтительная база данных, и ее очень легко запустить локально с помощью docker. Просто запустите:

```
432 -e POSTGRES_HOST_AUTH_METHOD=trust -e POSTGRES_DB=databasename postgres
```

Запустите контейнер Postgres

Когда изображение запущено, вы можете подключиться через порт, `5432` используя имя пользователя `postgres`, без пароля. Убедитесь, что вы подключаетесь к базе данных, имя которой вы использовали выше, а не к базе данных по умолчанию.

Настройка Prisma

Я начну с Next.js + TypeScript starter. Если у вас уже есть проект, вы можете пропустить это и просто использовать свой проект.

```
pnpm create next-app

# Or

yarn create next-app
```

Затем установите новые зависимости. Устанавливайте клиент только для производства.

```
pnpm i -D prisma
pnpm i @prisma/client

# Or

yarn add -SED prisma
yarn add -SE @prisma/client
```

После установки Prisma выполните следующее:

```
npx prisma init
```

Команда `prisma init` выполнит начальную настройку для клиента Prisma. Затем, чтобы подключиться к базе данных и начать использовать Prisma, вам нужно добавить `DATABASE_URL` в свой `.env` файл:

```
ABASE_URL="postgresql://postgres@localhost:5432/databasename?schema=public"
```

.env

Настройка клиента добавляет строку в этот файл по умолчанию. Убедитесь, что вы изменили его, чтобы он соответствовал указанным выше учетным данным или любым другим учетным данным, которые вы используете для своей базы данных. Это также необходимо будет обновить для вашей производственной среды.

Откройте новый `prisma/schema.prisma` файл, и вы сможете увидеть информацию о подключении. Далее давайте добавим базовую модель пользователя, чтобы убедиться, что все это работает:

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

prisma/schema.prisma

Вы можете найти ссылку на схему здесь.

Теперь вы можете создать свою первую миграцию, в которой будет записан SQL для базы данных. Он также запускает генерацию клиента.

```
npx prisma migrate dev --name init
```

Теперь вы готовы запросить свою базу данных!

Примечание по развертыванию

Клиент Prisma хранит свои определения типов в `node_modules` папке. Это убирает их с дороги и не загромождает вашу кодовую базу. Однако, когда вы выполняете развертывание, например, в Vercel, `node_modules` папка переустанавливается в процессе сборки.

Определения типов для вашего клиента там не будет.

Это приведет к сбою сборки позже, когда определения типов для ваших моделей отсутствуют. Чтобы исправить это, вам следует обновить `build` команду в вашем `package.json` файле.

```
"build": "prisma generate && next build",
```

package.json

При обновлении этой команды при выполнении производственной сборки первое, что генерируется, - это типы клиентов. Тогда остальная часть сборки будет работать так, как ожидалось.

Заполнение базы данных

Если вы хотите заполнить базу данных, вам необходимо установить, `ts-node` который запустит для вас начальный скрипт.

```
pnpm i -D ts-node

# Or

yarn add -SED ts-node
```

Затем обновите свой `package.json`, чтобы включить новый ключ Prisma. Это будет на верхнем уровне с начальной командой в нем:

```
"prisma": {
  "seed": "ts-node --compiler-options {\"module\":\"CommonJS\"} prisma/seed
},
```

Add this to your `package.json` file

Lastly, you can create the seeding script. Create a new file located at `prisma/seed.ts`. Here's an example creating a user:

```
import { PrismaClient } from '@prisma/client'

const prisma = new PrismaClient()

async function main() {
  const user = await prisma.user.upsert({
    where: { email: 'test@test.com' },
    update: {},
    create: {
      email: 'test@test.com',
      name: 'Test User',
        password: `$2y$12$GBfcgD6XwaMferSOdYGiduw3Awuo95QAPhxFE0oNJ.Ds8qj3p:
    },
  })
  console.log({ user })
}
main()
  .then(() => prisma.$disconnect())
  .catch(async (e) => {
    console.error(e)
    await prisma.$disconnect()
    process.exit(1)
  })
```

To execute the seed, you can run:

```
npx prisma db seed
```

Seeding also runs when you run `prisma migrate dev` or `npx prisma migrate reset`.

Now that you have some data, we can use it!

Using the Client

Create a new file at `lib/prisma.ts`. If you don't have a lib folder, create one. This file should not be exported in a barrel rollup file (`index.ts`) since the Prisma client can't be used in the

browser. Instead, import it as:

```
import { prisma } from '@/lib/prisma'
```

В файл добавьте следующее:

```
import { PrismaClient } from '@prisma/client'

// PrismaClient is attached to the `global` object in development to prevent
// exhausting your database connection limit.
//
// Learn more:
// https://pris.ly/d/help/next-js-best-practices

const globalForPrisma = global as unknown as { prisma: PrismaClient }

export const prisma =
  globalForPrisma.prisma ||
  new PrismaClient({
    log: ['query'],
  })

if (process.env.NODE_ENV !== 'production') globalForPrisma.prisma = prisma
```

библиотека/prisma.ts

вышесказанное добавляет глобальную переменную, чтобы не исчерпывать соединения.

Использование клиента Prisma

Вы можете использовать клиент Prisma в любом месте сервера. Вы часто будете использовать его в маршрутах API для изменения данных и в своих компонентах для извлечения данных.

Каталог приложений

Когда вы хотите получить данные в серверном компоненте, вы можете прочитать их непосредственно в компоненте:

```
import { prisma } from '@/lib/prisma'

export default async function Home() {
  const user = await prisma.user.findFirst({
    where: {
      email: 'test@test.com'
    }
  })

  return (
    <main>
      <div>Hello, {user?.name}</div>
    </main>
  )
}
```

A React Server Component `app/page.tsx`

Pages Directory

When you want to query with the client (such as in `getServerSideProps`), do:

```
import { prisma } from '@/lib/prisma'
import { User } from '@prisma/client'
import { GetServerSideProps } from 'next'

type Props = {
  user: User
}

export default function Page(props: Props) {
  return <main>Hello, {props.user.name}</main>
}

export const getServerSideProps: GetServerSideProps = async (context) => {
  const user = await prisma.user.findFirst({
    where: {
      email: 'test@test.com'
    }
  })

  return {
    props: {
      user
    }
  }
}
```

And there you go! You can now set up, manage, and query your database with Prisma.

Я создаю набор интернет-продуктов, чтобы найти лучшую альтернативу следующей технической работе. Первый продукт - это приложение SaaS, которое помогает управлять локализациями и копированием для вашего приложения.

Если вы хотите подписаться, пожалуйста, найдите меня в Twitter. Это очень много значит!

Опубликовано:

Вам также может понравиться...

АПРЕЛЬ 25 Использование среды выполнения Edge от Supabase для повышения п... 8 минут чтения

МАРТ 31 Как загрузить файл в Next.js Более 13 каталогов приложений без библ... 3 минуты чтения

МАРТ 30 Создайте REST API с помощью Next.js 13 ★ 1 минута чтения

МАРТ 24 Кембрийский взрыв искусственного интеллекта 1 минута чтения

МАРТ 24 Программное обеспечение - это инструмент 1 минута чтения

Регистрация