VARA API CODE TEST

This shows a solution to the to the coding exercise given by VARA EDTECH and how to use endpoints to perform CRUD operation. **REST API with TypeScript** using Express and Prisma Client. The solution uses a MySQL database with some initial dummy data which you can find at `./prisma/seed.ts`.

Getting started

1. Clone the repo and install dependencies

Clone this repository:

```
git clone https://github.com/Marrwan/vara_edtech.githttps://github.com/Marrwan/vara_edtech.git
```

Install npm dependencies:

```
cd vara
npm install
```

2. Enviroment variables

Create a `.env` file in the root of the project and add the following variables:

```
DATABASE_URL="mysql://root:password@localhost:3306/vara"
NODE_ENV="development"
PORT=4000
```

Or you can copy it from the `.env.example` file.

3. Create and seed the database

Run the following command to create your SQLite database file. This also creates the `Customer` and `Address` tables that are defined in `prisma/schema.prisma`:

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

Run the following command to seed the database with some initial data:

```
npx ts-node prisma/seed.ts
```

The seed file in `prisma/seed.ts` will be executed and your database will be populated with the sample data.

3. Start the REST API server

```
npm run dev
```

The server is now running on `http://localhost:4000`. You can now run the API requests, e.g. `http://localhost:4000/api/customers`.

Using the REST API

You can access the REST API of the server using the following endpoints:

GET

CUSTOMERS

- `/api/customers` - Fetch all customers
- `/api/customers/:id` - fetch a customer by its id

ADDRESSES

- `/api/address` - Fetch all addresses
- `/api/address/:id` - fetch an address by its id

POST

CUSTOMERS

- `/api/customers` - Create a new customer
  - Body:
    - `name - string` (required) : The name of the customer
    - `email - string` (required) : The email address of the customer
    - `phone - string` (required) : The phone number of the customer

ADDRESSES

- `/api/address` - Create a new address
  - Body:
    - `street - string` (required) : The street of the address
    - `city - string` (required) : The city of the address
    - `state - string` (required) : The state of the address
    - `zip - string` (required) : The zip code of the address
    - `customerId - string` (required) : The id of the customer that the address belongs to

PATCH

CUSTOMERS

- `/api/customers/:id` - Update a customer
  - Body:
    - `name - string` (optional) : The name of the customer
    - `email - string` (optional) : The email address of the customer
    - `phone - string` (optional) : The phone number of the customer

ADDRESSES

- `/api/address/:id` - Update an address
  - Body:
    - `street - string` (optional) : The street of the address
    - `city - string` (optional) : The city of the address
    - `state - string` (optional) : The state of the address
    - `zip - string` (optional) : The zip code of the address
    - `customerId - string` (optional) : The id of the customer that the address belongs to

DELETE

CUSTOMERS

- `/api/customers/:id` - Delete a customer

ADDRESSES

- `/api/address/:id` - Delete an address

Switch to another database (e.g. PostgreSQL, MySQL, SQL Server, MongoDB)

If you want to try this example with another database than MySQL, you can adjust the the database connection in `prisma/schema.prisma` by reconfiguring the `datasource` block.

Learn more about the different connection configurations in the docs.

Expand for an overview of example configurations with different databases

PostgreSQL

For PostgreSQL, the connection URL has the following structure:

```
datasource db {
  provider = "postgresql"
  url      = "postgresql://USER:PASSWORD@HOST:PORT/DATABASE?schema=SCHEMA"
}
```

Here is an example connection string with a local PostgreSQL database:

```
datasource db {
  provider = "postgresql"
  url      = "postgresql://janedoe:mypassword@localhost:5432/notesapi?schema=public"
}
```

MySQL

For MySQL, the connection URL has the following structure:

```
datasource db {
  provider = "mysql"
  url      = "mysql://USER:PASSWORD@HOST:PORT/DATABASE"
}
```

Here is an example connection string with a local MySQL database:

```
datasource db {
  provider = "mysql"
  url      = "mysql://janedoe:mypassword@localhost:3306/notesapi"
}
```

Microsoft SQL Server

Here is an example connection string with a local Microsoft SQL Server database:

```
datasource db {
  provider = "sqlserver"
  url      = "sqlserver://localhost:1433;initial catalog=sample;user=sa;password=mypassword;"
}
```

MongoDB

Here is an example connection string with a local MongoDB database:

```
datasource db {
  provider = "mongodb"
  url      = "mongodb://USERNAME:PASSWORD@HOST/DATABASE?authSource=admin&retryWrites=true&w=majority"
}
```

Because MongoDB is currently in Preview, you need to specify the `previewFeatures` on your `generator` block:

```
generator client {
  provider        = "prisma-client-js"
  previewFeatures = ["mongodb"]
}
```