CS 631 Database Management Systems Design

Employee Pay Rolls

Project Report

BY:
Shravani Aedla(sa2588)

Harini Reddy Gade(hg294)

<u>Table of Contents</u>

1 Introduction

CS 631 Employee payrolls provide various functions for employee details. The project is basically used by an employee to first enter their login details and each employee has certain records which contain information related to the employee, which is saved in the database. Employees can do certain operations such as insert, update and delete in the database and calculate the salary.

2 Summary of System Requirements(Technical Stack)

The following tools were used to create the database for the Payroll and the applications used by the HRs to input and extract information of the employees so the payroll can be calculated.

- **NodeJS**
  Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser.

- **MySQL**
  This is an open-source relational database management system. This was used to create the schema that will store the customer and employee information.

- **Prisma**
  Prisma is an open-source ORM for Node.js and TypeScript. It is used as an alternative to writing plain SQL or using another database access tool such as SQL query builders (like knex.js) or ORMs (like TypeORM and Sequelize). Prisma currently supports PostgreSQL, MySQL, SQL Server, SQLite, and MongoDB (preview).
- **Docker**
  Docker is an open-source containerization platform. It enables developers to package applications into containers—standardized executable components combining application source code with the operating system (OS) libraries and dependencies required to run that code in any environment.

- **NextJS**
  Next.js is an open-source web development framework built on top of Node.js enabling React-based web applications functionalities such as server-side rendering and generating static websites. React documentation mentions Next.js among "Recommended Toolchains" advising it to developers as a solution when "Building a server-rendered website with Node.js".[4] Where traditional React apps can only render their content in the client-side browser, Next.js extends this functionality to include applications rendered on the server side.

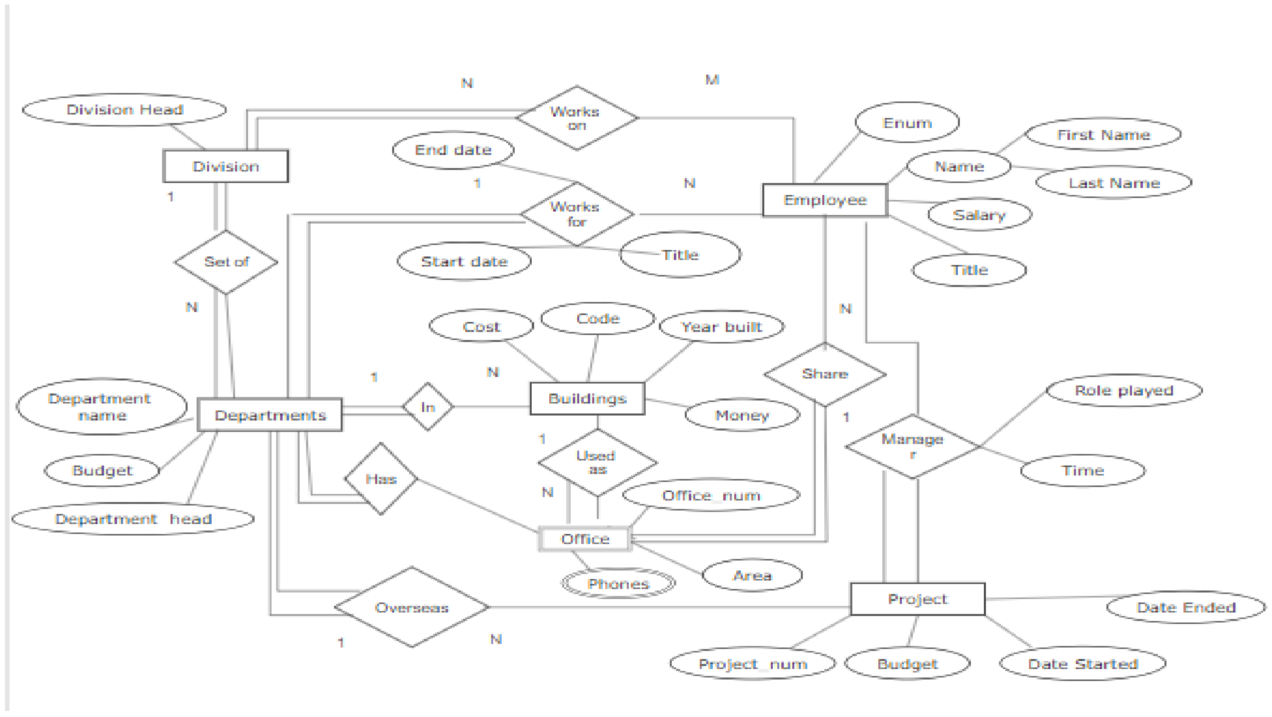
- **Material-UI**
  Material-UI is simply a library that allows us to import and use different components to create a user interface in our React applications. This saves a significant amount of time since the developers do not need to write everything from scratch

- **HTML**
  Hypertext Markup Language is the standard markup language for documents designed to be displayed in a web browser. This was used for the user interface.

RAM Specifications: 8GB RAM
Storage Specifications: 50 GB HardDisk

3 Entity-Relationship Diagram

This ER Diagram shows the entities used and the relationships between them that were identified in the database design requirements.



This is a full fledged ERD diagram for the full project.

4 Logical Database Design

Entities:

Departments
Division
Employee
Project
Buildings

One to One mapping

Step 1: Handling Entities

- **Department:** Department_name, Budget, Dept_head.
- **Division:** Division_head.
- **Employee:** Enum,first_name,lastname,salary,title.
- **Project:** Projectno,date_started,date_ended.
- **Buildings:** Code,Cost,Year built,Money.
- **Office:** office num,phones,area.

Step 2: Weak Entities

- **Office:** office num,phones,area,Enum,Code,Department_name.

Step 3: 1 to N relations

a 1:N relationship is mapped by included the key of the table resulting from mapping the entity on the 1 side of the relationship into the table of the one at the N side.

Step 4: 1 to N relations

- **Department:** Dept_name,Budget,Dept_head,(F-K)->Division head
- **Project:** Projectnum,budget,datestarted,dateended,(FK)->Dept_name
- **Buildings:** Code,Cost,yearbuilt,money,(FK)->Dept_name
- **Office:** office_num,phones,area,(FK)->Code
- **Employee:** Enum,F-N,L-N,Salary,title,(FK)->Office_Num
- **Division:** Division_head

Step 5: N to M relations

- **Manage:** Enum,Projectno,Roleplayed,time
- **Workson:** Division,Enum

5 Application Program Design

HR Application

Creating a new employee: Give Name, Email, Salary, a new employee is created.

Deleting existing client: From the employee table, using the delete button, an employee can be deleted

Salary Calculations

Federal Tax: We take 10% as the federal tax on the income, So therefore we have a column showing the exact deductions due to this federal tax applied on the total net salary.

State Tax: We take 5% as the state tax on the income, So therefore we have a column showing the exact deductions due to this state tax applied on the total net salary.

Other Tax: We take 3% as the other tax on the income, So therefore we have a column showing the exact deductions due to this other tax applied on the total net salary.

In-hand salary: The final salary after all the deductions are displayed here.

6 Design Decisions

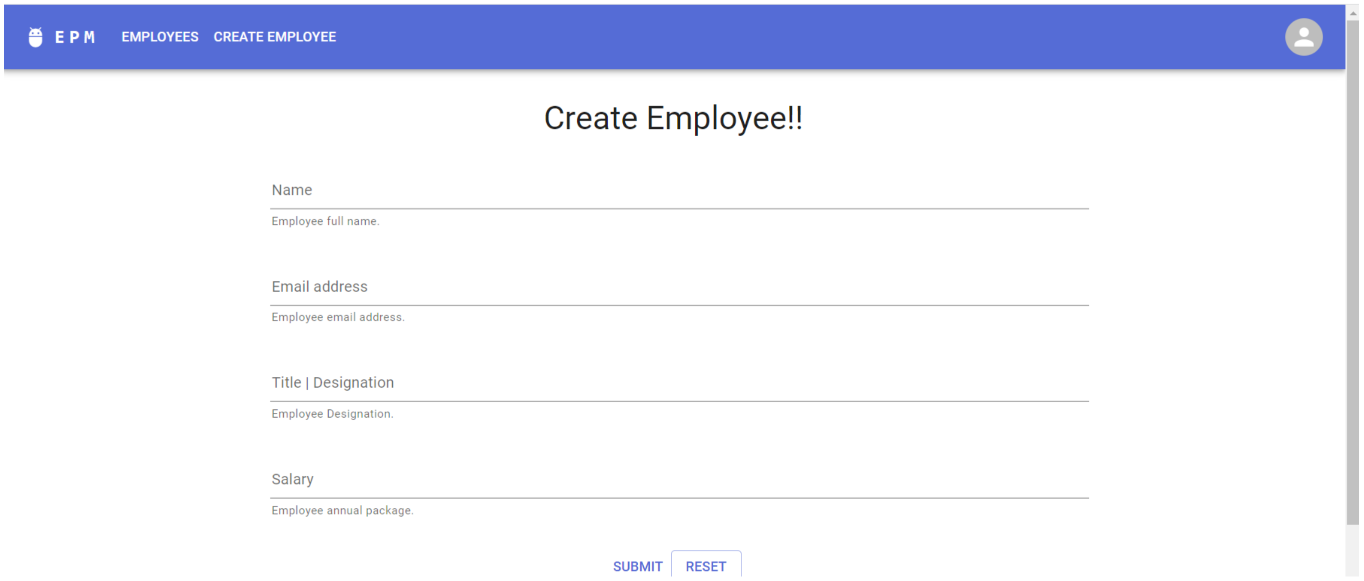
For our Database, we opted for MySQL Server as it is used by a lot of companies and tech experts and is very feasible to use. We were contemplating between PHP and NodeJS but decided to go with NodeJS as our choice of code. NodeJS ecosystem has a lot of new modern tools which enabled programmers to write code and debug very efficiently. We also have an orm called Prisma, Which is very special as its client is automatically generated with the help of sch.

The reason behind choosing NextJS is the most obvious reason of it being a decendant of ReactJS. ReactJS has been around for a long time and it has a lot of prebuilt libraries which help in speeding up the development time. For example, we have used Material-UI wich let us eliminate writing of any css on out code, even doing any styling for our project. NextJS can be also be said as full stack project as we can implement apis as well in the nextjs project.

In the design model, we made sure we covered the requirements stated for the Term Project. The tables or the entities that are to be created for Employees are in form and intact. Furthermore, our ER Diagram explains everything.

7. Relational Instances Tables

Create Employee



Create Employee!!

Name

Employee full name.

Email address

Employee email address.

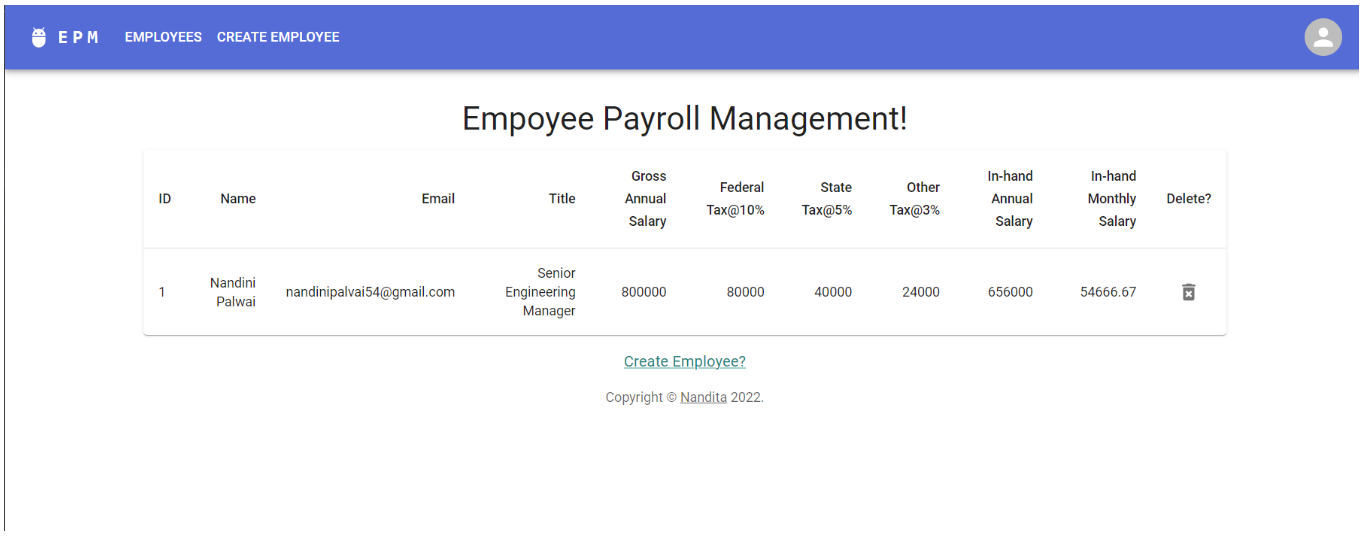
Title | Designation

Employee Designation.

Salary

Employee annual package.

Employee Payroll Management



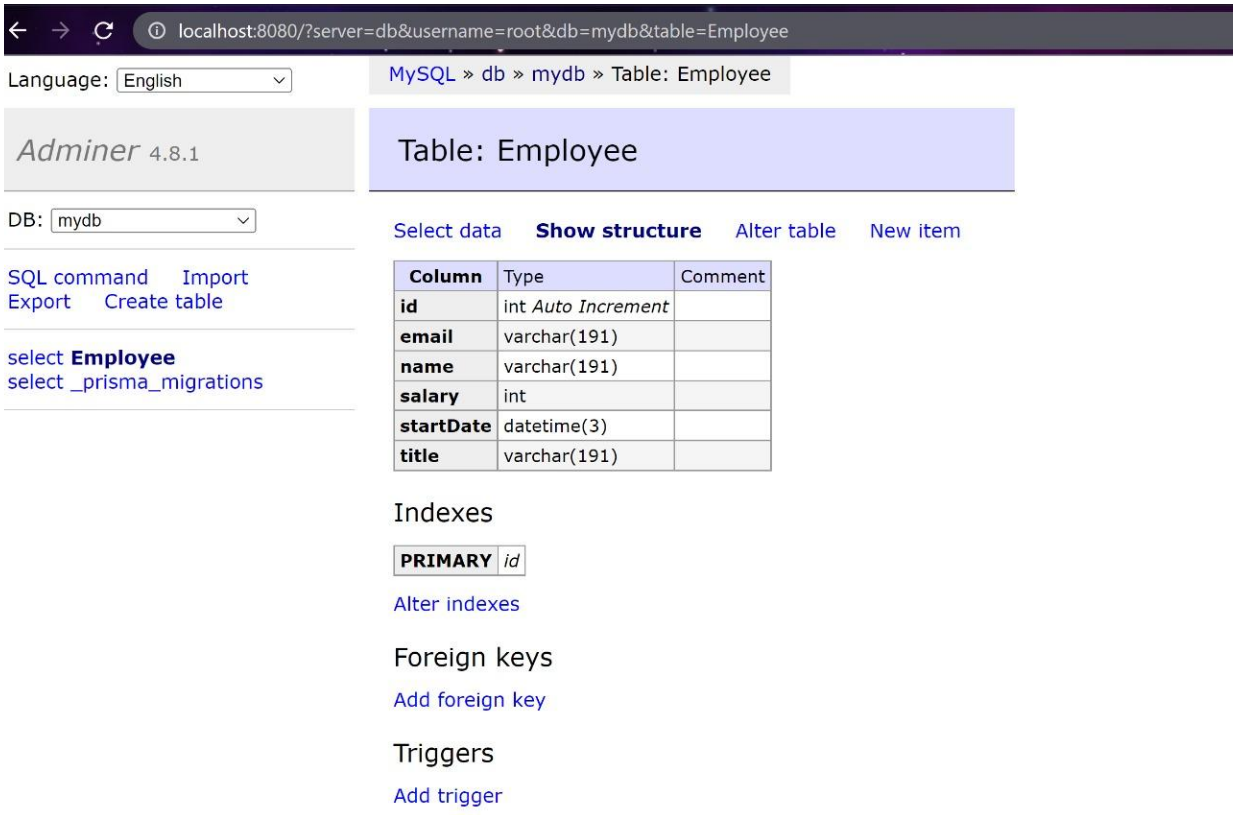
Empoyee Payroll Management!

ID	Name	Email	Title	Gross Annual Salary	Federal Tax@10%	State Tax@5%	Other Tax@3%	In-hand Annual Salary	In-hand Monthly Salary	Delete?
1	Nandini Palwai	nandinipalvai54@gmail.com	Senior Engineering Manager	800000	80000	40000	24000	656000	54666.67	🗑

Create Employee?

Copyright © Nandita 2022.

Employee Table



localhost:8080/?server=db&username=root&db=mydb&table=Employee

Language: English

MySQL » db » mydb » Table: Employee

Adminer 4.8.1

DB: mydb

SQL command Import Export Create table

select **Employee**
select _prisma_migrations

Table: Employee

Select data **Show structure** Alter table New item

Column	Type	Comment
id	int *Auto Increment*	
email	varchar(191)	
name	varchar(191)	
salary	int	
startDate	datetime(3)	
title	varchar(191)	

Indexes

PRIMARY	*id*

Alter indexes

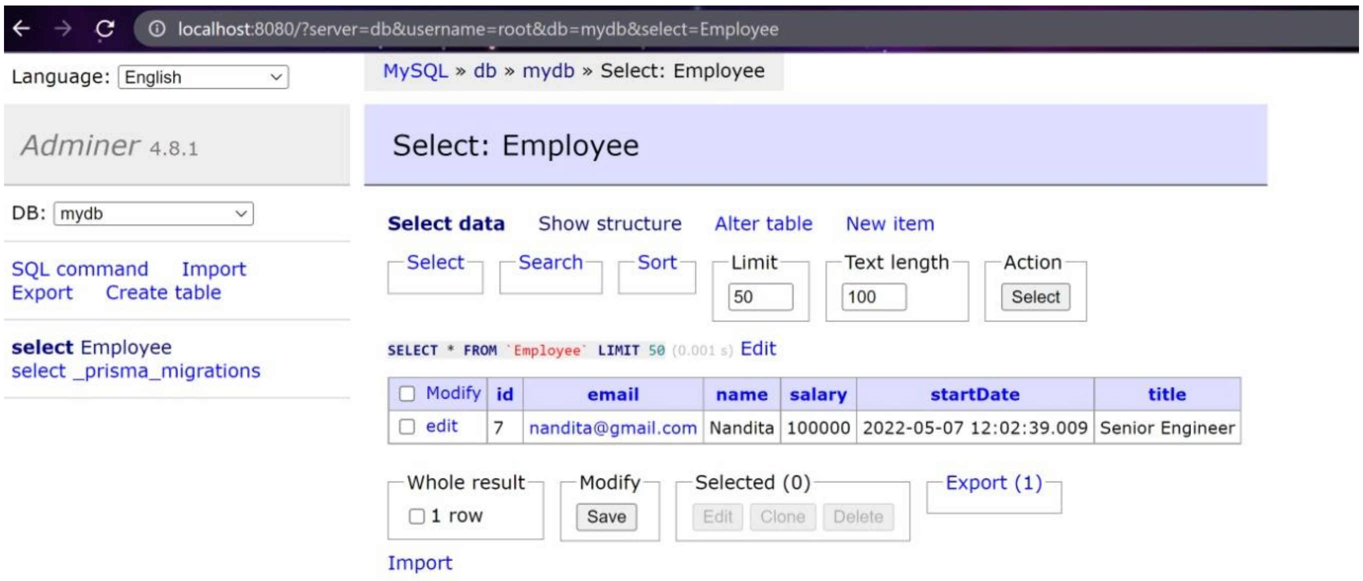
Foreign keys

Add foreign key

Triggers

Add trigger

Select Employee



localhost:8080/?server=db&username=root&db=mydb&select=Employee

Language: English

MySQL » db » mydb » Select: Employee

Adminer 4.8.1

DB: mydb

SQL command Import Export Create table

select Employee
select _prisma_migrations

Select: Employee

Select data Show structure Alter table New item

Select Search Sort Limit 50 Text length 100 Action Select

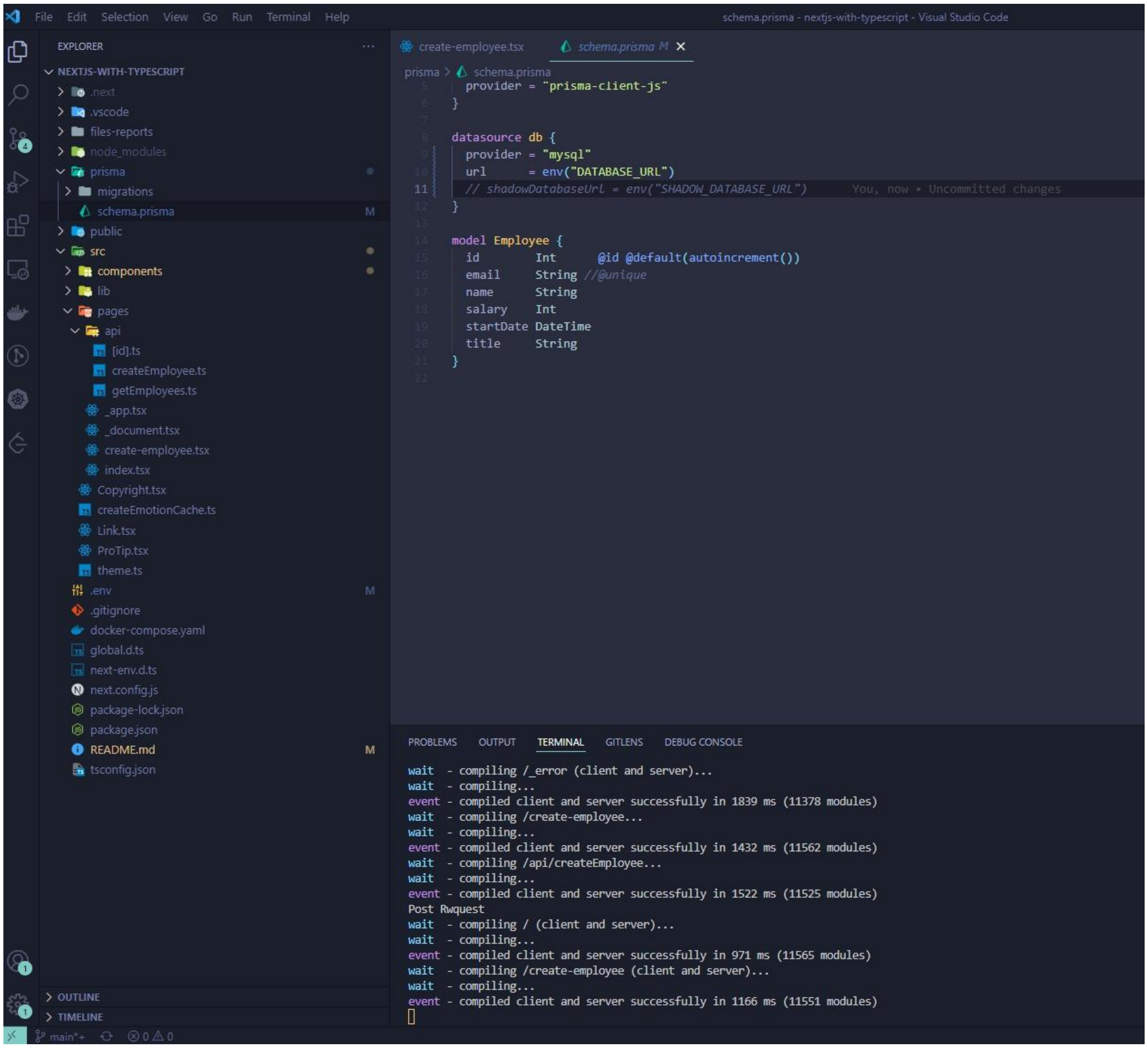
```
SELECT * FROM `Employee` LIMIT 50 (0.001 s) Edit
```

Modify	id	email	name	salary	startDate	title
edit	7	nandita@gmail.com	Nandita	100000	2022-05-07 12:02:39.009	Senior Engineer

Whole result: 1 row
Modify: Save
Selected (0): Edit Clone Delete
Export (1)

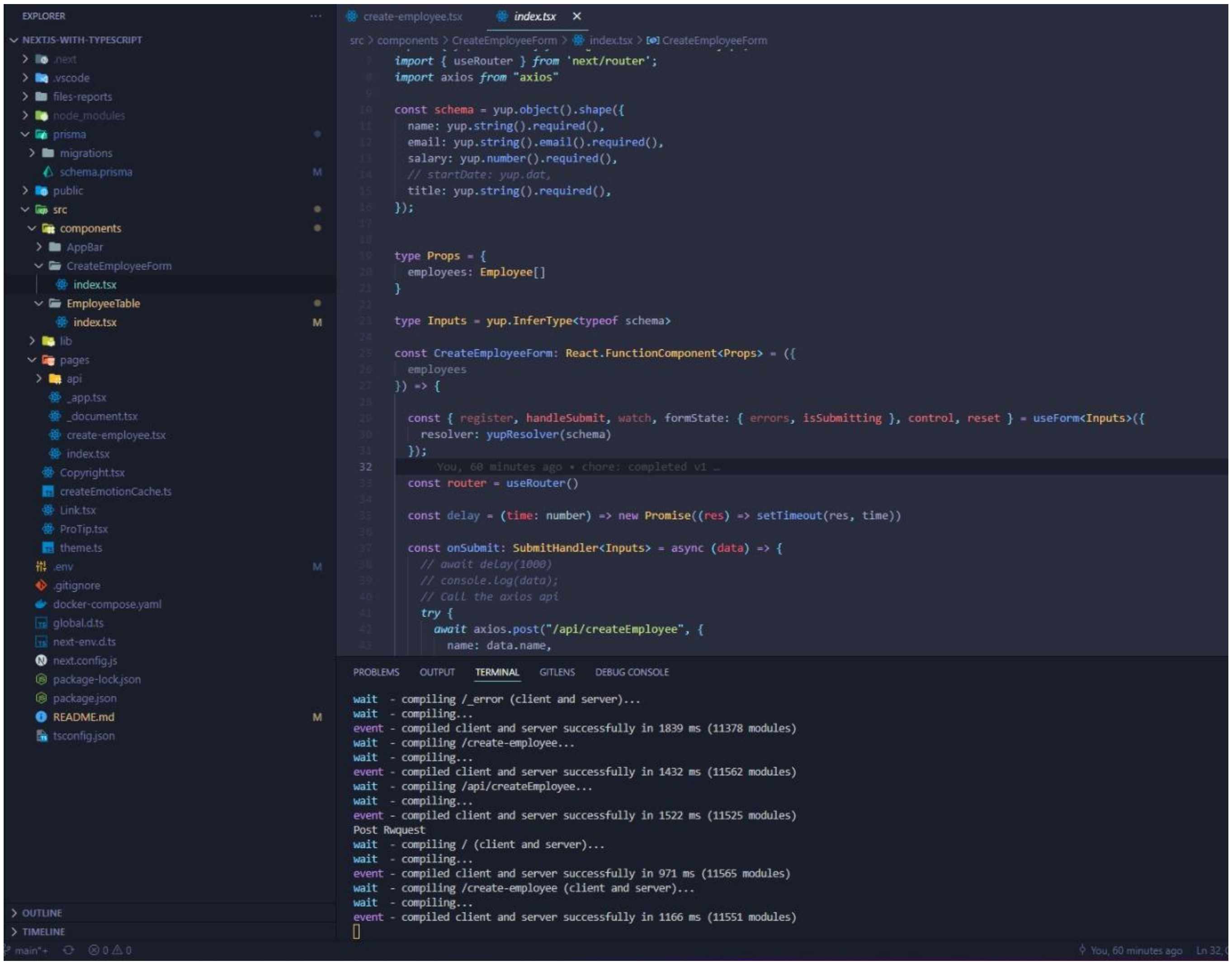
Import

Code Snippets : Create Employee Snippet



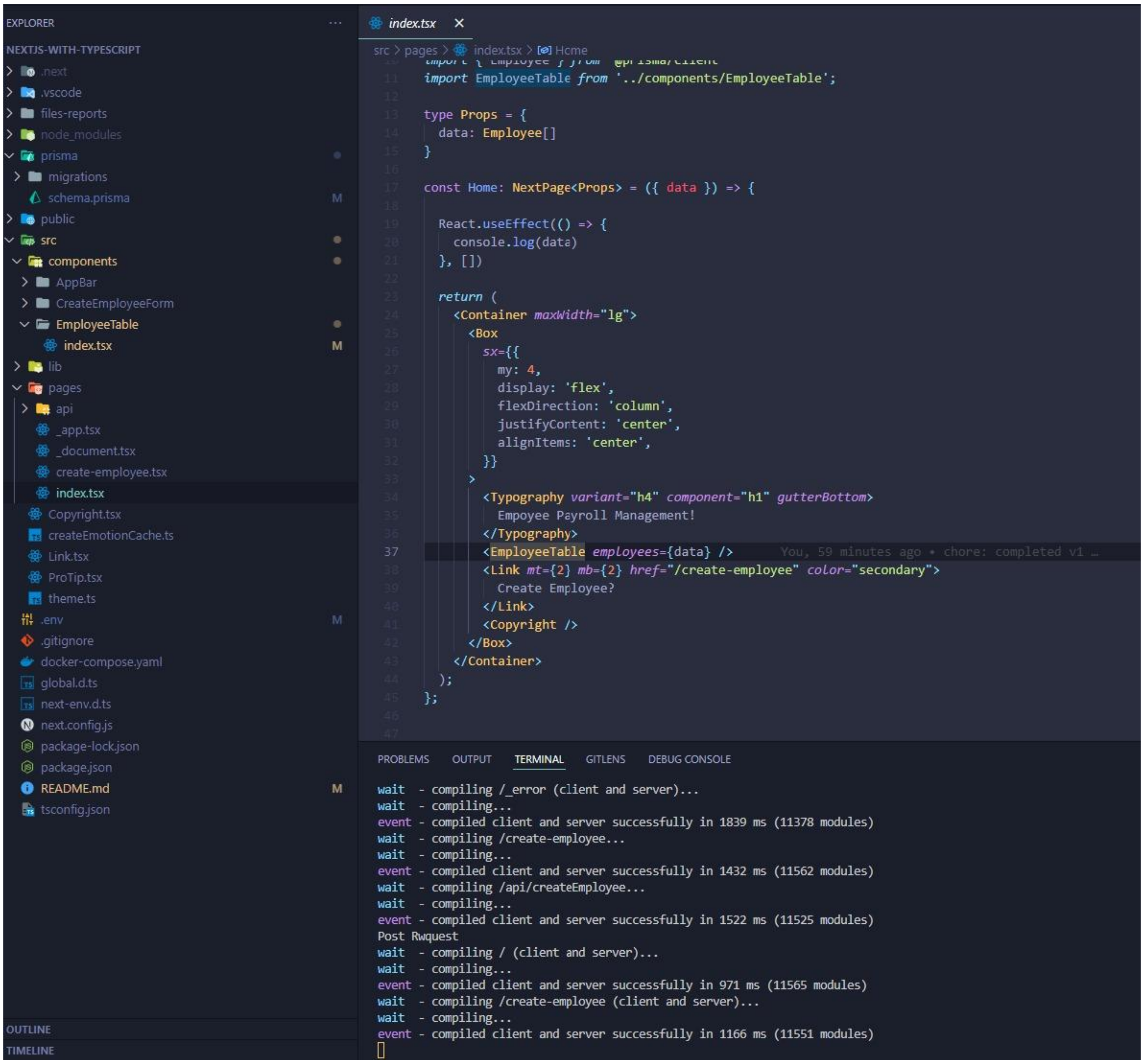
```
  provider = "prisma-client-js"
}

datasource db {
  provider = "mysql"
  url      = env("DATABASE_URL")
  // shadowDatabaseUrl = env("SHADOW_DATABASE_URL")      You, now • Uncommitted changes
}

model Employee {
  id        Int      @id @default(autoincrement())
  email     String //@unique
  name      String
  salary    Int
  startDate DateTime
  title     String
}
```

```
wait  - compiling /_error (client and server)...
wait  - compiling...
event - compiled client and server successfully in 1839 ms (11378 modules)
wait  - compiling /create-employee...
wait  - compiling...
event - compiled client and server successfully in 1432 ms (11562 modules)
wait  - compiling /api/createEmployee...
wait  - compiling...
event - compiled client and server successfully in 1522 ms (11525 modules)
Post Rwquest
wait  - compiling / (client and server)...
wait  - compiling...
event - compiled client and server successfully in 971 ms (11565 modules)
wait  - compiling /create-employee (client and server)...
wait  - compiling...
event - compiled client and server successfully in 1166 ms (11551 modules)
```



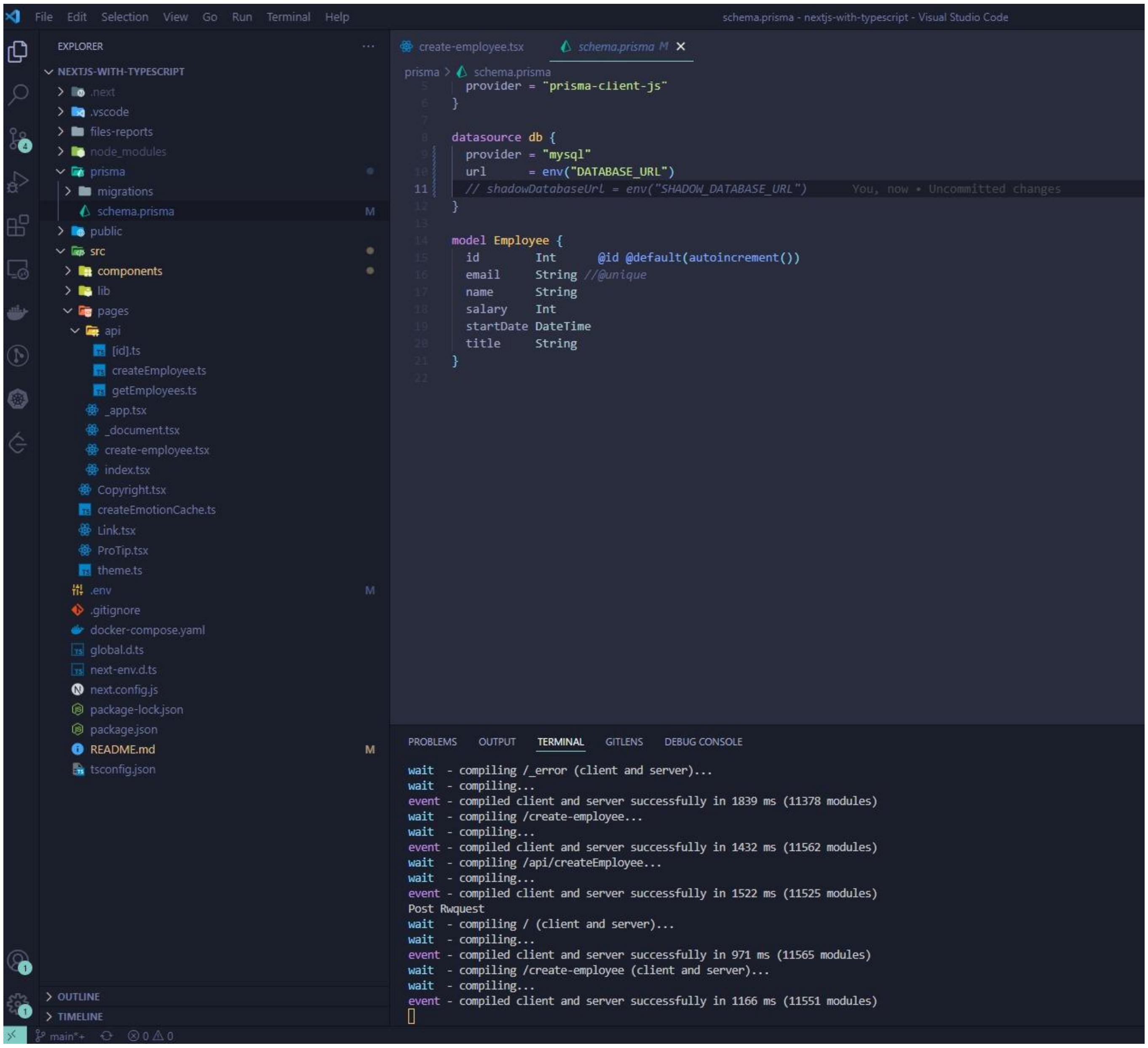
```tsx
import { useRouter } from 'next/router';
import axios from "axios"

const schema = yup.object().shape({
  name: yup.string().required(),
  email: yup.string().email().required(),
  salary: yup.number().required(),
  // startDate: yup.dat,
  title: yup.string().required(),
});


type Props = {
  employees: Employee[]
}

type Inputs = yup.InferType<typeof schema>

const CreateEmployeeForm: React.FunctionComponent<Props> = ({
  employees
}) => {

  const { register, handleSubmit, watch, formState: { errors, isSubmitting }, control, reset } = useForm<Inputs>({
    resolver: yupResolver(schema)
  });
      You, 60 minutes ago • chore: completed v1 …
  const router = useRouter()

  const delay = (time: number) => new Promise((res) => setTimeout(res, time))

  const onSubmit: SubmitHandler<Inputs> = async (data) => {
    // await delay(1000)
    // console.log(data);
    // Call the axios api
    try {
      await axios.post("/api/createEmployee", {
        name: data.name,
```

```
wait  - compiling /_error (client and server)...
wait  - compiling...
event - compiled client and server successfully in 1839 ms (11378 modules)
wait  - compiling /create-employee...
wait  - compiling...
event - compiled client and server successfully in 1432 ms (11562 modules)
wait  - compiling /api/createEmployee...
wait  - compiling...
event - compiled client and server successfully in 1522 ms (11525 modules)
Post Rwquest
wait  - compiling / (client and server)...
wait  - compiling...
event - compiled client and server successfully in 971 ms (11565 modules)
wait  - compiling /create-employee (client and server)...
wait  - compiling...
event - compiled client and server successfully in 1166 ms (11551 modules)
```



```tsx
import { Employee } from '@prisma/client'
import EmployeeTable from '../components/EmployeeTable';

type Props = {
  data: Employee[]
}

const Home: NextPage<Props> = ({ data }) => {

  React.useEffect(() => {
    console.log(data)
  }, [])

  return (
    <Container maxWidth="lg">
      <Box
        sx={{
          my: 4,
          display: 'flex',
          flexDirection: 'column',
          justifyContent: 'center',
          alignItems: 'center',
        }}
      >
        <Typography variant="h4" component="h1" gutterBottom>
          Empoyee Payroll Management!
        </Typography>
        <EmployeeTable employees={data} />       You, 59 minutes ago • chore: completed v1 …
        <Link mt={2} mb={2} href="/create-employee" color="secondary">
          Create Employee?
        </Link>
        <Copyright />
      </Box>
    </Container>
  );
};
```

```
wait  - compiling /_error (client and server)...
wait  - compiling...
event - compiled client and server successfully in 1839 ms (11378 modules)
wait  - compiling /create-employee...
wait  - compiling...
event - compiled client and server successfully in 1432 ms (11562 modules)
wait  - compiling /api/createEmployee...
wait  - compiling...
event - compiled client and server successfully in 1522 ms (11525 modules)
Post Rwquest
wait  - compiling / (client and server)...
wait  - compiling...
event - compiled client and server successfully in 971 ms (11565 modules)
wait  - compiling /create-employee (client and server)...
wait  - compiling...
event - compiled client and server successfully in 1166 ms (11551 modules)
```



```
  provider = "prisma-client-js"
}

datasource db {
  provider = "mysql"
  url      = env("DATABASE_URL")
  // shadowDatabaseUrl = env("SHADOW_DATABASE_URL")      You, now • Uncommitted changes
}

model Employee {
  id        Int      @id @default(autoincrement())
  email     String //@unique
  name      String
  salary    Int
  startDate DateTime
  title     String
}
```

```
wait  - compiling /_error (client and server)...
wait  - compiling...
event - compiled client and server successfully in 1839 ms (11378 modules)
wait  - compiling /create-employee...
wait  - compiling...
event - compiled client and server successfully in 1432 ms (11562 modules)
wait  - compiling /api/createEmployee...
wait  - compiling...
event - compiled client and server successfully in 1522 ms (11525 modules)
Post Rwquest
wait  - compiling / (client and server)...
wait  - compiling...
event - compiled client and server successfully in 971 ms (11565 modules)
wait  - compiling /create-employee (client and server)...
wait  - compiling...
event - compiled client and server successfully in 1166 ms (11551 modules)
```