МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИИ

федеральное государственное автономное образовательное учреждение высшего образования

Санкт-Петербургский национальный исследовательский университет информационных технологий механики и оптики

Мегафакультет трансляционных информационных технологий

Факультет информационных технологий и программирования

Лабораторная работа №3

По дисциплине «Web-программирование»

Создание доменной модели

Выполнил студент группы
***M33122*:**
Федотенко Николай Владимирович

Проверил:
Приискалов Роман Андреевич

САНКТ-ПЕТЕРБУРГ

2022

Цель работы:

Спроектировать модель данных, описывающую выбранную прикладную область.

Краткие теоретические сведения:

- **Heroku Postgres** – это сервис для работы с PostgreSQL через Heroku, основные преимущества: надёжность, безопасность и масштабируемость.
- **Prisma** – это ORM нового поколения с открытым исходным кодом для Node.js и TypeScript; состоит из инструментов: *Prisma Client*, *Prisma Migrate* и *Prisma Studio*. Слоган Prisma: "*Разработчики приложений должны думать о данных, а не о SQL*".
- **DDD** (*Domain Driven Design*, рус. *Предметно-ориентированное проектирование*) – это набор принципов и схем, направленных на создание систем объектов.
  Основные определения:
  - **Область** (*Domain*) – предметная область, к которой применяется ПО;
  - **Модель** (*Model*) – описывает отдельные аспекты области;
  - **Язык описания** – используется для единого стиля описания домена/модели.

Ход выполнения работы:

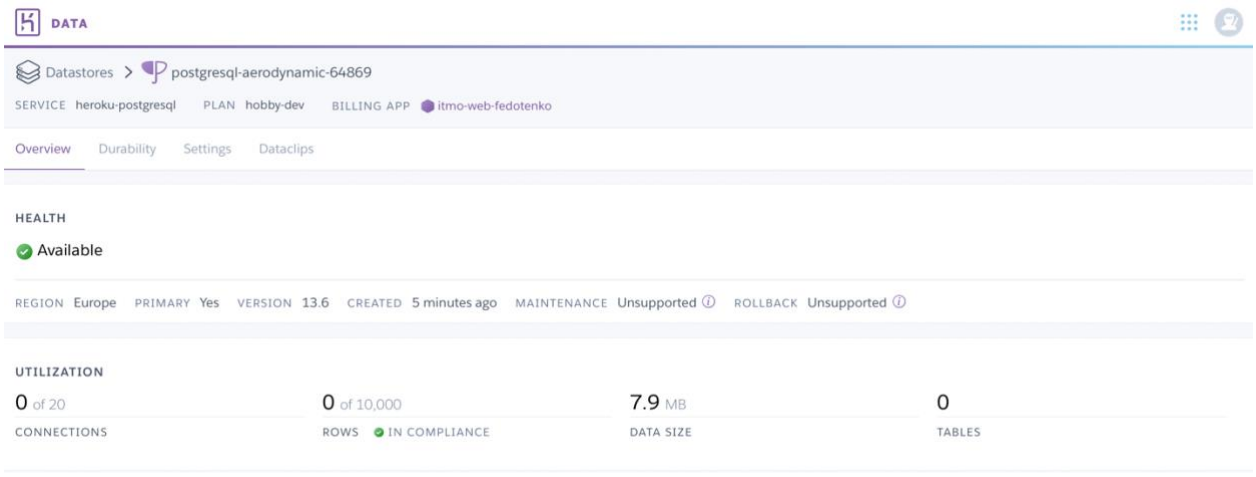
Данная лабораторная работа выполнена в операционной системе *macOS*.

Используемая IDE: *WebStorm 2021.3* (by JetBrains)

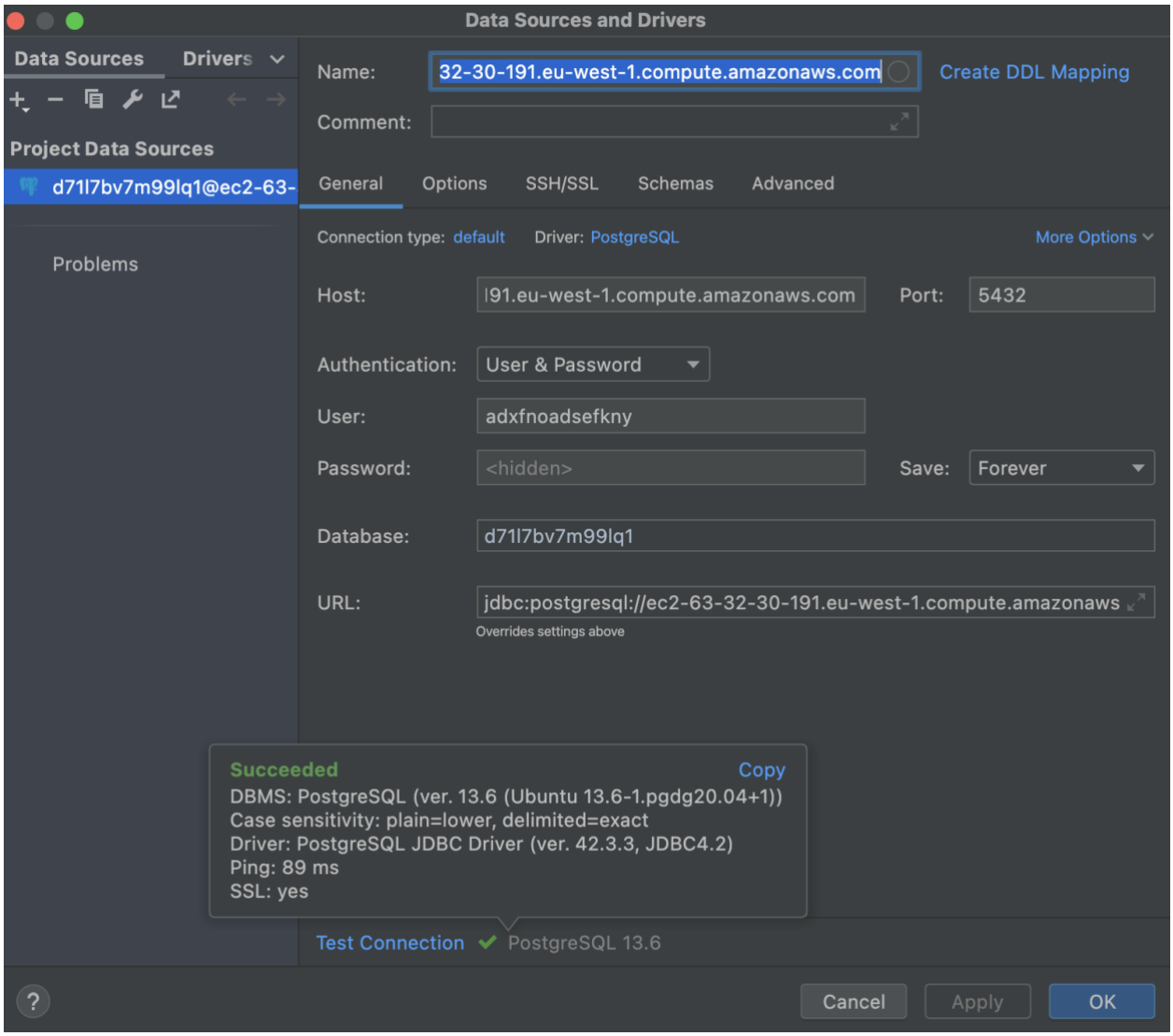
1. Подключение Heroku Postgres в качестве СУБД:

```
(base) kalkolay@MacBook-Pro-Nikolay itmo-web-2 % heroku addons:create heroku-postgresql:hobby-dev
Creating heroku-postgresql:hobby-dev on ⬢ itmo-web-fedotenko... !
 ▸    Invalid credentials provided.
heroku: Press any key to open up the browser to login or q to exit:
Opening browser to https://cli-auth.heroku.com/auth/cli/browser/4725710f-6d17-4f4e-a821-6d3eb0ccc2f1?requestor=SFMyNTY.g2gDbQAAAA4xNjQuOTIuMTUyLjIyNW4GAMjgOZJ_AWIAAVGA.fDE7vbO83KbhG_0CVMaBnxX3rxWsHJvbEyBw2W5UDRA
Logging in... done
Logged in as vafedotenko@gmail.com
Creating heroku-postgresql:hobby-dev on ⬢ itmo-web-fedotenko... free
Database has been created and is available
 ! This database is empty. If upgrading, you can transfer
 ! data from another database with pg:copy
Created postgresql-aerodynamic-64869 as DATABASE_URL
Use heroku addons:docs heroku-postgresql to view documentation
(base) kalkolay@MacBook-Pro-Nikolay itmo-web-2 %
```

Убедимся, что на Heroku действительно появилась новая БД:



2. Проверка подключения к БД с помощью DataGrip:



3. Подключение коннектора к выбранной СУБД к проекту:

```
(base) kalkolay@MacBook-Pro-Nikolay itmo-web-2 % npm install pg --save

added 14 packages, and audited 798 packages in 2s

83 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
(base) kalkolay@MacBook-Pro-Nikolay itmo-web-2 %
```

4. Установка ORM для работы со слоем данных (**Prisma**):

```
$ npm install prisma --save-dev
```

```
(base) kalkolay@MacBook-Pro-Nikolay itmo-web-2 % npx prisma init

✔ Your Prisma schema was created at prisma/schema.prisma
  You can now open it in your favorite editor.

warn You already have a .gitignore. Don't forget to exclude .env to not commit any secret.

Next steps:
1. Set the DATABASE_URL in the .env file to point to your existing database. If your database has no tables yet, read https://pris.ly/d/getting-started
2. Set the provider of the datasource block in schema.prisma to match your database: postgresql, mysql, sqlite, sqlserver or mongodb (Preview).
3. Run prisma db pull to turn your database schema into a Prisma schema.
4. Run prisma generate to generate the Prisma Client. You can then start querying your database.

More information in our documentation:
https://pris.ly/d/getting-started
```

Содержание модифицированных файлов Prisma:

schema.prisma

```
// This is your Prisma schema file,
// learn more about it in the docs: https://pris.ly/d/prisma-schema

generator client {
  provider = "prisma-client-js"
}

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

.env

```
# Environment variables declared in this file are automatically made available to Prisma.
# See the documentation for more detail: https://pris.ly/d/prisma-schema#accessing-environment-variables-from-the-schema

# Prisma supports the native connection string format for PostgreSQL, MySQL, SQLite, SQL Server, MongoDB (Preview) and C
# See the documentation for all the connection string options: https://pris.ly/d/connection-strings

DATABASE_URL="postgres://adxfnoadsefkny:683ffdb757d9e29dcedc9d5ca42f2701af5d7c15db0b7efabd6dc4d675139c93@ec2-63-32-30-19
```

Установка Prisma клиента:

```
$ npm install @prisma/client
```

5. Описание моделей данных в проекте:

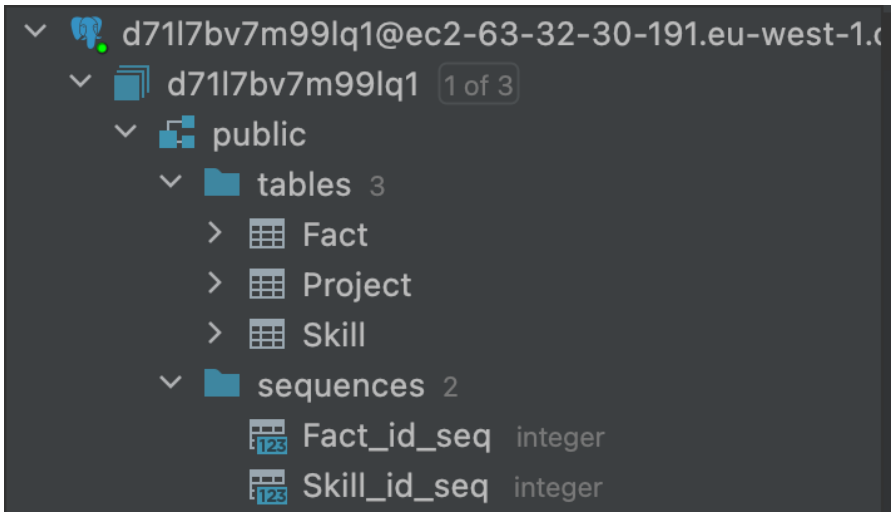
Было решено выделить в отдельные модели содержимое блоков с **фактами**, **навыками** и **проектами** (в файле *schema.prisma*), так как данные сущности на текущий момент реализации передаются в контроллер, хотя впоследствии могут быть модифицированы или дополнены:

```
model Fact {
  id   Int     @default(autoincrement()) @id
  fact String  @unique
}

model Skill {
  id   Int     @default(autoincrement()) @id
  name String?
  link String
}

model Project {
  id   String  @id
  name String?
  link String
}
```

Генерация SQL файлов из модели Prisma:

```
(base) kalkolay@MacBook-Pro-Nikolay itmo-web-2 % npx prisma db push --preview-feature
prisma:warn Prisma "db push" was in Preview and is now Generally Available.
You can now remove the --preview-feature flag.
Environment variables loaded from .env
Prisma schema loaded from prisma/schema.prisma
Datasource "db": PostgreSQL database "d71l7bv7m99lq1", schema "public" at "ec2-63-32-30-191.eu-west-1.compute.amazonaws.com:5432"

🚀  Your database is now in sync with your schema. Done in 2.50s

✔ Generated Prisma Client (3.11.0 | library) to ./node_modules/@prisma/client in 57ms
```

Проверка добавления в Heroku Postgres:



Реализация обёрток (сервисов) для взаимодействия через Prisma Client API:

prisma.service.ts

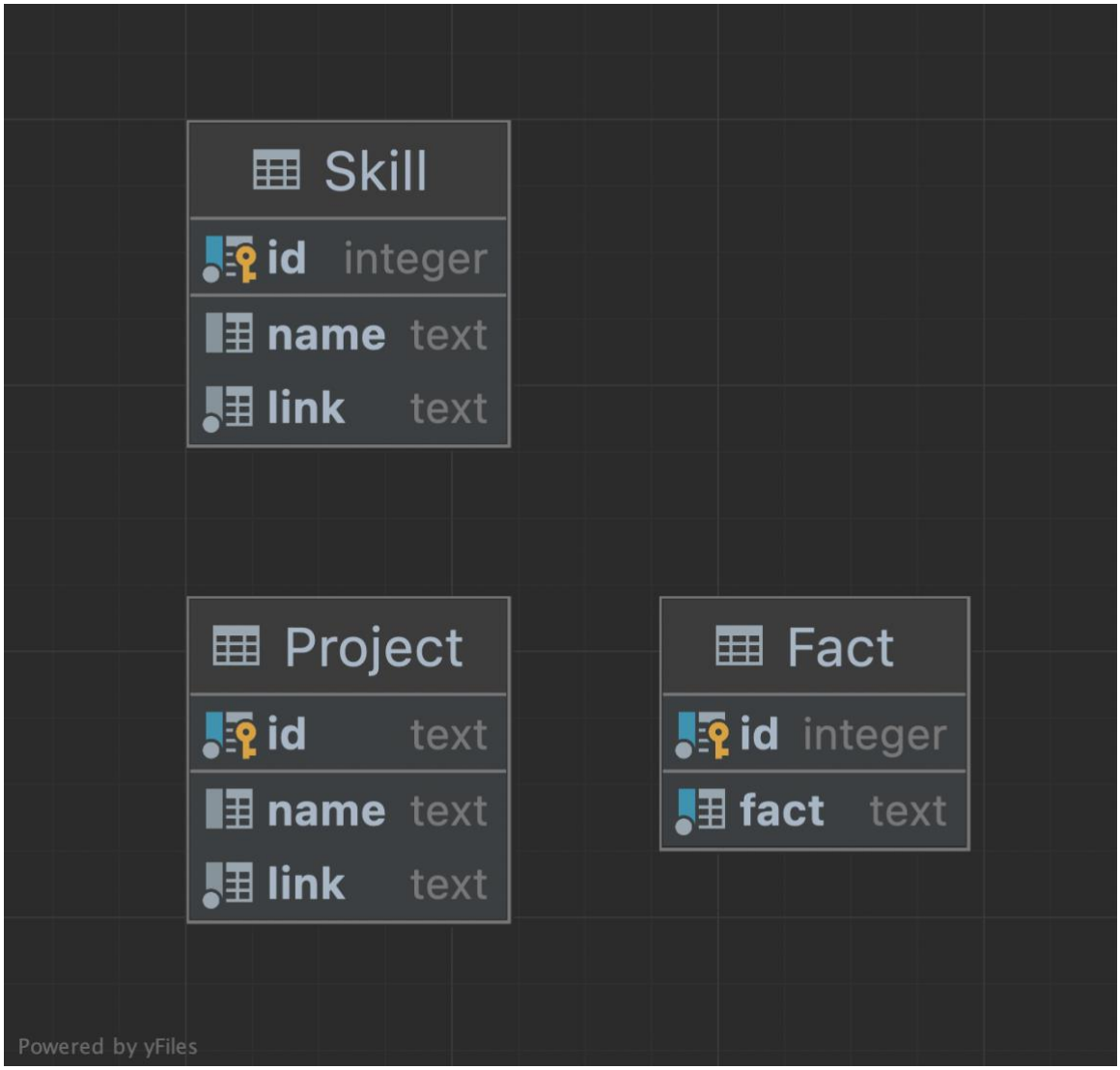
```ts
import { INestApplication, Injectable, OnModuleInit } from '@nestjs/common';
import { PrismaClient } from '@prisma/client';

@Injectable()
export class PrismaService extends PrismaClient implements OnModuleInit {
  async onModuleInit() {
    await this.$connect();
  }

  async enableShutdownHooks(app: INestApplication) {
    this.$on('beforeExit', async () => {
      await app.close();
    });
  }
}
```

fact.service.ts (аналогично выглядят сервисы *skill* и *project*):

```ts
import { Injectable } from '@nestjs/common';
import { PrismaService } from './prisma.service';
import { Fact, Prisma } from '@prisma/client';

@Injectable()
export class FactService {
  constructor(private prisma: PrismaService) {}

  async fact(
    factWhereUniqueInput: Prisma.FactWhereUniqueInput,
  ): Promise<Fact | null> {
    return this.prisma.fact.findUnique({
      where: factWhereUniqueInput,
    });
  }

  async facts(params: {
    skip?: number;
    take?: number;
    cursor?: Prisma.FactWhereUniqueInput;
    where?: Prisma.FactWhereInput;
    orderBy?: Prisma.FactOrderByWithRelationInput;
  }): Promise<Fact[]> {
    const { skip, take, cursor, where, orderBy } = params;
    return this.prisma.fact.findMany({
      skip,
      take,
      cursor,
      where,
      orderBy,
    });
  }

  async createFact(data: Prisma.FactCreateInput): Promise<Fact> {
    return this.prisma.fact.create({
      data,
    });
  }

  async updateFact(params: {
```

```
    where: Prisma.FactWhereUniqueInput;
    data: Prisma.FactUpdateInput;
  }): Promise<Fact> {
    const { data, where } = params;
    return this.prisma.fact.update({
      data,
      where,
    });
  }

  async deleteFact(where: Prisma.FactWhereUniqueInput): Promise<Fact> {
    return this.prisma.fact.delete({
      where,
    });
  }
}
```

6. Визуальное представление схемы в виде ERD диаграммы:



Вывод:

Я спроектировал модель данных для имеющегося проекта с использованием Prisma ORM и Heroku Postgres, придерживаясь принципа DDD.