САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа №3

Выполнил:

Дорофеева Арина

Группа к33401

Проверил:
Добряков Д. И.

Санкт-Петербург

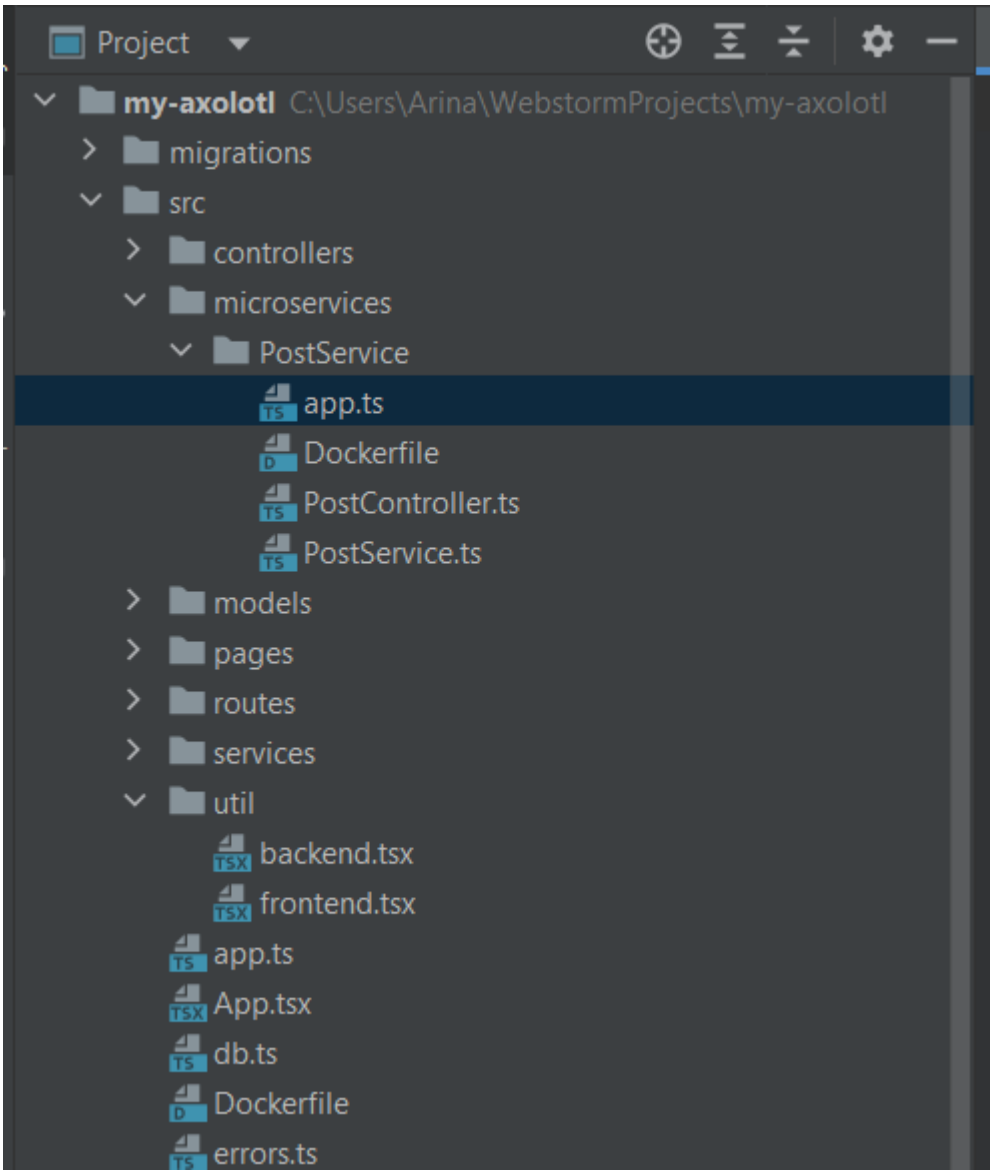
2022 г.

Задача

Необходимо реализовать отдельный микросервис, выполняющий какую-либо содержательную функцию из всего арсенала функций вашего приложения.

Ход работы

Структура проекта



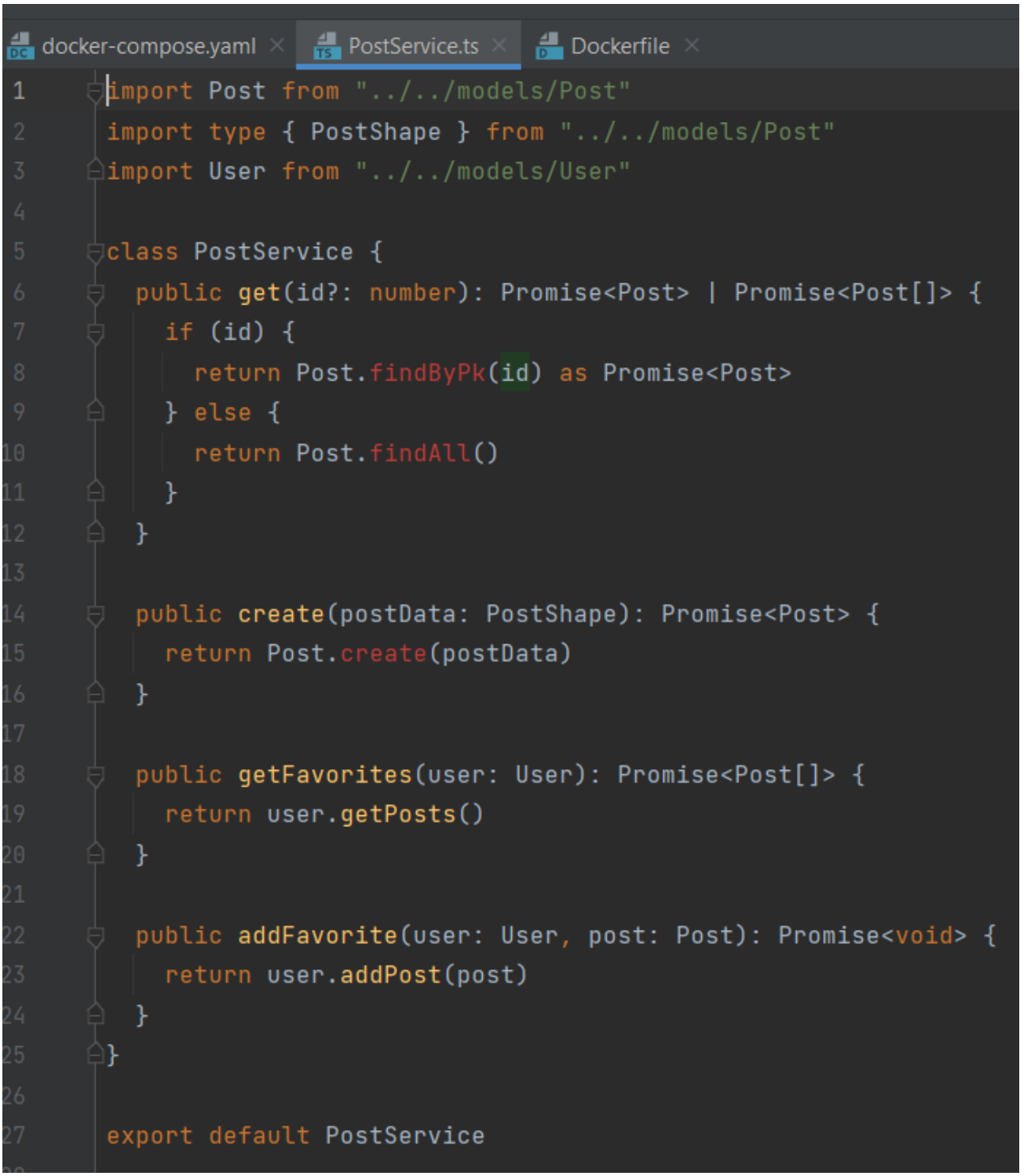
app.ts

app.ts

```ts
import dotenv from "dotenv"
dotenv.config()
import ...

const { DB_NAME, POST_PORT } = process.env

sequelize
  .authenticate()
  .then(() => {
    console.log(`Connected to database ${ DB_NAME }.`)
    sequelize.associate?.()
    sequelize
      .sync()
      .then(() => console.log(`Models have been synced to database ${ DB_NAME }.`))
  })

const app = Express()
```

```ts
app.use(cookieParser)
app.use(cookieAuther)
app.use(BodyParser.urlencoded({ extended: false }))
app.use(BodyParser.json())

const controller = new PostController()

app.get("/:id", controller.get)
app.get("/", controller.get)
app.post("/", controller.post)
app.post("/favorites", controller.addFavorite)

const server = createServer(app)

server.listen(POST_PORT)
console.log(`Listening on port ${ POST_PORT }.`)
```

PostService.ts



PostController.ts

docker-compose.yaml | PostService.ts | PostController.ts | Dockerfile

```
import { handleGenericError } from "../../errors"
import PostService from "./PostService"
import type { PostShape } from "../../models/Post"
import type { Request } from "express"
import Post from "../../models/Post"
import { ResponseOrError } from "../../models/shapes"

class PostController {
  private postService: PostService

  public constructor() {
    this.postService = new PostService()
  }
```

```
  public get = (req: Request, res: ResponseOrError<PostShape | PostShape[]>) => {
    const { id } = req.params
    const { search, favorites } = req.query
    const { user } = res.locals
    if (id) {
      return this.postService.get(Number(id))
        .then(post => res.status(200).send(post))
    } else {
      Promise.all( values: [
        this.postService.get(),
        this.postService.getFavorites(user)
      ]) Promise<(Awaited<Post | Post[]>)[]>
        .then(([all : Post | Post[] , fav : Post | Post[] ]) => {
          let result = favorites ? fav as Post[] : all as Post[];
          if (search) {
            result = result.filter(post =>
              (post.title + post.text).toLowerCase().includes((search as string).toLowerCase()))
          }
          result = result
            .map(post => {
              (post as any).dataValues.favorite = fav.some(favPost => post.id === favPost.id)
              return post
            })
          res.status(200).send(result)
        }) Promise<void>
        .catch(e => handleGenericError(res, e))
    }
  }
```

```
  public post = (req: Request, res: ResponseOrError<PostShape>) => {
    const { post }: { post: PostShape } = req.body
    this.postService
      .create(post)
      .then(post => res.status(200).send(post))
      .catch(e => handleGenericError(res, e))
  }

  public addFavorite = (req: Request, res: ResponseOrError<any>) => {
    const { user } = res.locals
    const { id: postId } = req.body
    this.postService
      .get(Number(postId))
      .then(post =>
        this.postService
          .addFavorite(user, post as Post)
          .then(() => res.status(200).send({}))
      )
  }
}

export default PostController
```

Вывод

В ходе работы я вынесла в отдельный микросервис часть работы приложения, которая отвечает за получение аксолотлей, получение одного аксолотля, добавление и тоггл избранного.