

**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

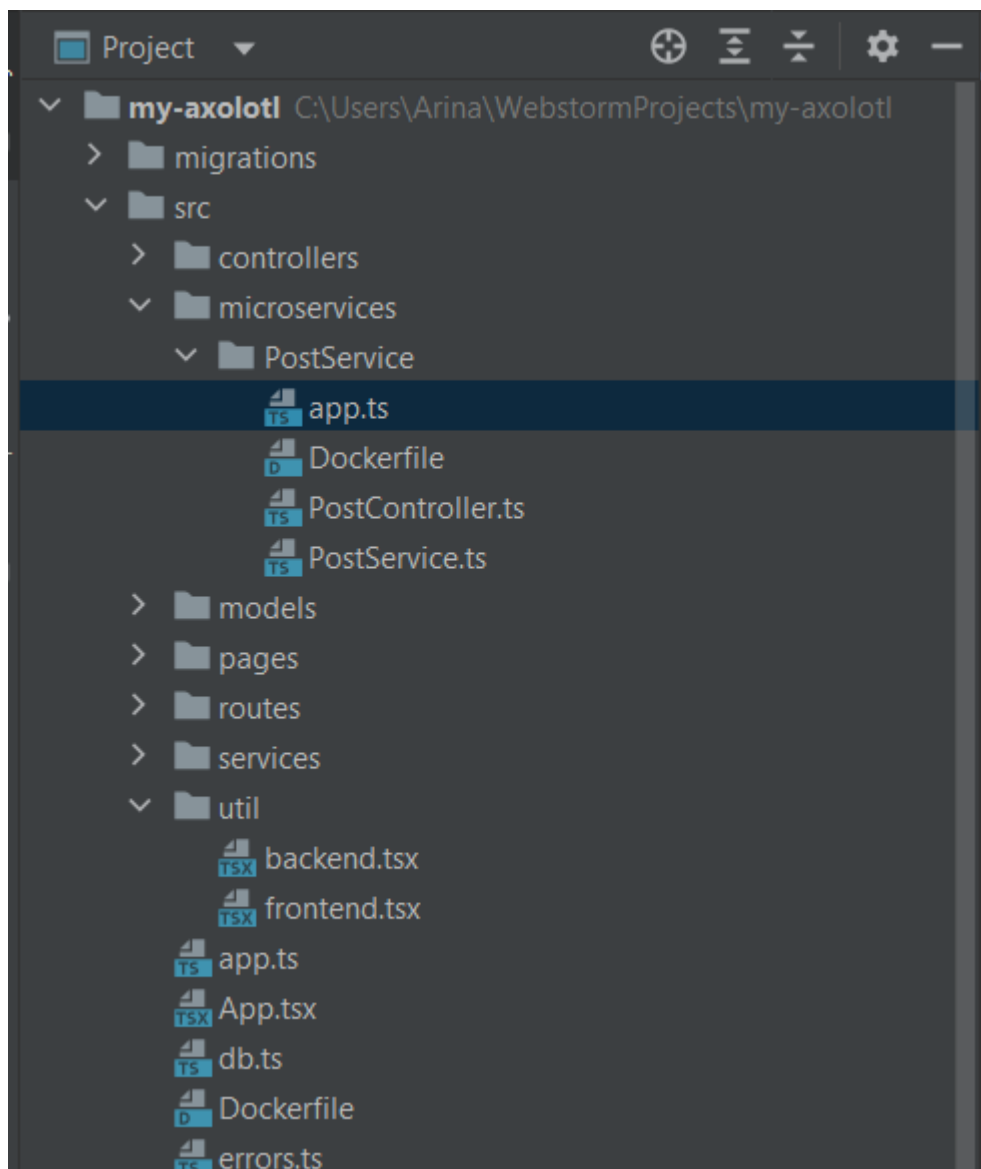
2022 г.

## Задача

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

## Ход работы

### Структура проекта



## app.ts

```
1  import dotenv from "dotenv"
2  dotenv.config()
3  import ...
9
10 const { DB_NAME, POST_PORT } = process.env
11
12 sequelize
13   .authenticate()
14   .then(() => {
15     console.log(`Connected to database ${ DB_NAME }.`)
16     sequelize.associate?().()
17     sequelize
18       .sync()
19       .then(() => console.log(`Models have been synced to database ${ DB_NAME }.`))
20   })
21
22 const app = Express()
23
24 app.use(cookieParser)
25 app.use(cookieAuther)
26 app.use(bodyParser.urlencoded({ extended: false }))
27 app.use(bodyParser.json())
28
29 const controller = new PostController()
30
31 app.get("/:id", controller.get)
32 app.get("/", controller.get)
33 app.post("/", controller.post)
34 app.post("/favorites", controller.addFavorite)
35
36 const server = createServer(app)
37
38 server.listen(POST_PORT)
39 console.log(`Listening on port ${ POST_PORT }.`)
40
```

## PostService.ts

```
docker-compose.yaml × PostService.ts × Dockerfile ×
1  import Post from "../../models/Post"
2      import type { PostShape } from "../../models/Post"
3  import User from "../../models/User"
4
5  class PostService {
6      public get(id?: number): Promise<Post> | Promise<Post[]> {
7          if (id) {
8              return Post.findByPk(id) as Promise<Post>
9          } else {
10             return Post.findAll()
11          }
12      }
13
14      public create(postData: PostShape): Promise<Post> {
15          return Post.create(postData)
16      }
17
18      public getFavorites(user: User): Promise<Post[]> {
19          return user.getPosts()
20      }
21
22      public addFavorite(user: User, post: Post): Promise<void> {
23          return user.addPost(post)
24      }
25  }
26
27  export default PostService
```

## PostController.ts

```
1 import { handleGenericError } from "../../errors"
2 import PostService from "../PostService"
3 import type { PostShape } from "../../models/Post"
4 import type { Request } from "express"
5 import Post from "../../models/Post"
6 import { ResponseOrError } from "../../models/shapes"
7
8 class PostController {
9     private postService: PostService
10
11     public constructor() {
12         this.postService = new PostService()
13     }
14
15     public get = (req: Request, res: ResponseOrError<PostShape | PostShape[]>) => {
16         const { id } = req.params
17         const { search, favorites } = req.query
18         const { user } = res.locals
19         if (id) {
20             return this.postService.get(Number(id))
21                 .then(post => res.status(200).send(post))
22         } else {
23             Promise.all( values: [
24                 this.postService.get(),
25                 this.postService.getFavorites(user)
26             ]) Promise<(Awaited<Post | Post[]>)>
27                 .then(([all : Post | Post[] , fav : Post | Post[] ]) => {
28                     let result = favorites ? fav as Post[] : all as Post[];
29                     if (search) {
30                         result = result.filter(post =>
31                             (post.title + post.text).toLowerCase().includes((search as string).toLowerCase()))
32                     }
33                     result = result
34                         .map(post => {
35                             (post as any).dataValues.favorite = fav.some(favPost => post.id === favPost.id)
36                             return post
37                         })
38                     res.status(200).send(result)
39                 }) Promise<void>
40             .catch(e => handleGenericError(res, e))
41         }
42     }
```

```

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.