

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Фронт-энд разработка

Отчет

Лабораторная работа №2

Выполнил:

Жаров Александр Павлович

Группа:

K33402

Проверил:

Добряков Д. И.

Санкт-Петербург

2023 г.

Задача

Расширить функционал на сайте к первой лабораторной. Связать все данные с бд и добавить запросы к API.

Ход работы

Для начала был создан сервер на Node js на котором написан функционал для авторизации и собственное api, с помощью которого сервер общается с MongoDB.

```
1  const express = require("express");
2  const cors = require("cors");
3  const mongoose = require("mongoose");
4  const apiRouter = require("./routers/apiRouter");
5  const authRouter = require("./routers/authRouter");
6
7  const app = express();
8
9  const PORT = 3030;
10 const db = "mongodb://127.0.0.1:27017/portfolio";
11
12 mongoose
13   .connect(db)
14   .then((res) => console.log("Connect to DB"))
15   .catch((error) => console.log(error));
16
17 app.use(cors());
18
19 app.listen(PORT, (error) => {
20   error ? console.log(error) : console.log(`listening port ${PORT}`);
21 });
22
23 app.use(express.urlencoded({ extended: false }));
24 app.use(express.json());
25
26 app.use("/", authRouter);
27 app.use("/api", apiRouter);
28
```

```

You, 9 hours ago | 1 author (You)
1  const Router = require("express");
2  const apiRouter = new Router();
3  const apiController = require("../controllers/apiController");
4  const authMiddleware = require("../middleware/authMiddleware");
5  ⚡
6  apiRouter.post("/get-users", authMiddleware, apiController.getUsers);
7
8  apiRouter.get("/get-user", authMiddleware, apiController.getUserById);
9
10 apiRouter.post("/update-user", authMiddleware, apiController.updateUser);
11
12 module.exports = apiRouter;
13

```

```

1  const Router = require("express");
2  const authRouter = new Router();
3  const authController = require("../controllers/authController");
4
5  authRouter.post("/registration", authController.registration);
6
7  authRouter.post("/login", authController.login);
8
9  module.exports = authRouter;
10

```

Процесс авторизации выглядит следующим образом: сначала пользователь заполняет форму

Создание аккаунта

Имя

Почта

Пароль

Уже есть аккаунт

Создать аккаунт

При нажатии на кнопку создать аккаунт – отправляется post запрос на сервер с введенными данными.

```
const onRegistration = async () => {
  const res = await fetch("http://localhost:3030/registration", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify(formState),
  })
  .then((response) => response.json())
  .then((data) => {
    if (data.token) {
      setCookie("token", JSON.stringify(data.token));
      fetchUser(data.token).then((user) => {
        setUser(user);
        setAuth(true);
        navigate("/");
      });
    } else {
      console.log(data.message);
    }
  })
  .catch((error) => console.log(error));

  return res;
};
```

На сервере происходит обработка запроса, в ходе которой в бд создается пользователь, и генерируется jwt token, который отправляется в ответе на клиент.

```

6  const generateAccessToken = (id, email) => {
7      const payload = {
8          id,
9          email,
10     };
11     ⚡
12     return jwt.sign(payload, secret, { expiresIn: "12h" });
13 };
14 class authController {
15     async registration(req, res) {
16         try {
17             const { email, name, password } = req.body;
18             const candidate = await User.findOne({ email });
19
20             if (candidate) {
21                 return res.status(400).json({
22                     message: "Пользователь с этой почтой уже существует",
23                 });
24             }
25
26             const hashPassword = bcrypt.hashSync(password, 7);
27             const user = new User({ email, name, password: hashPassword });
28
29             await user.save();
30
31             const token = generateAccessToken(user._id, user.email);
32
33             return res.json({ token });
34         } catch (error) {
35             console.error(error);
36             res.status(500).send("Internal Server Error");
37         }
38     }

```

На клиенте полученный токен сохраняется в куки и отправляется следующий запрос, уже к Api.

```

useEffect(() => {
  const token = getCookie("token");
  console.log(token);

  if (token && !isAuthenticated) {
    fetchUserCallback(token).then((user) => {
      if (user.message) {
        setAuth(false);
        setUser(null);
        deleteCookie("token");
      } else {
        setAuth(true);
        setUser(user);
        navigate("/");
        console.log(user);
      }
    });
  }
}, []);
// eslint-disable-next-line react-hooks/exhaustive-deps
}, []);

```

Токен отправляется в хедере запроса, т.к без него api не вернет результат.

```

async getUserById(req, res) {
  try {
    const user = await User.findById(req.user.id);
    res.send(user);
  } catch (error) {
    console.error(error);
    res.status(500).send("Internal Server Error");
  }
}

```

```

1 const jwt = require("jsonwebtoken"); 60.8k (gzipped: 18k)
2 const { secret } = require("../config");
3
4 module.exports = function (req, res, next) {
5   if (req.method === "OPTIONS") {
6     next();
7   }
8
9   try {
10    const token = req.headers.authorization.split(" ")[1];
11
12    if (!token) {
13      return res.status(403).json({ message: "Пользователь не авторизован" });
14    }
15    const decodeData = jwt.verify(token, secret);
16
17    req.user = decodeData;
18    next();
19  } catch (error) {
20    console.log(error);
21    return res
22      .status(403)
23      .json({ message: "Пользователь не авторизован ошибка" });
24  }
25 };

```

Информация о пользователе сохранена в БД и выведена на клиенте

portfolio.users 5 1
DOCUMENTS INDEXES

Documents Aggregations Schema Indexes Validation

Filter Type a query: { field: 'value' } Explain Reset Find </> Options

ADD DATA EXPORT DATA 1 - 5 of 5 ↺ ↻ ⋮ {} ⌂

```

name: "Sasha Zharov"
password: "$2a$07$ungE595loDD9SCk0/R.0w0ewvXQ8ck5CFTLLKWCDA1VJqVF8Dy.3i"
__v: 0
about: "Меня зовут Саша"
experience: "Мое первое место работы была команда 12"
projects: "Проекты 1"

```

```


_id: ObjectId('65302305edd1d80a3b6794ea')
email: "email2"
name: "test21"
password: "$2a$07$7h2gGpCv6C9cEFx9IqmYpexhASknU0J4PaDe0DySEapvMzzd.z516"
__v: 0

```

```


_id: ObjectId('653056a92e23e4f3a3396e38')
email: "a.zharov@gmail.com"
name: "Александр Жаров"
password: "$2a$07$VJ5V4FY0xAWwMSRi8Hd9DelwYyeq/pSu5b1D0qEm.2y.sXcZmLamm"
__v: 0

```

 Профиль Личный кабинет

Search

Search



test

О себе

Места работы

Первое место работы

Проекты


-

Контакты

-

При выходе или окончании срока жизни jwt, он удаляется из cookie и приложение запрашивает авторизоваться заново.

Также был реализован поиск по юзерам.

 Профиль Личный кабинет

Search

Search

Sasha Zharov

Меня зовут Саша

Опыт работы: Мое первое место работы была команда 12

Перейти

test21

Опыт работы:

Перейти

Александр Жаров

Опыт работы: Я не работал

Перейти

Александр

Опыт работы:

При поиске на сервер отправляется запрос на получение юзеров и данные из строки поиска.

Сервер возвращает результат поиска


```

async getUsers(req, res) {
  try {
    let users = [];
    console.log(req.body);
    if (req.body.search) {
      const searchRegex = new RegExp(req.body.search, "i");
      users = await User.find({ name: searchRegex });
    } else {
      users = await User.find();
    }
    res.send(users);
  } catch (error) {
    console.error(error);
    res.status(500).send("Internal Server Error");
  }
}

async getUserById(req, res) {
  try {

```

```

export async function fetchUsers(token, search) {
  const res = await fetch("http://localhost:3030/api/get-users", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      Authorization: `Bearer ${token}`,
    },
    body: JSON.stringify({ search }),
  })
  .then((response) => response.json())
  .then((responseData) => {
    if (responseData) {
      return responseData;
    } else {
      console.log(responseData.message);
    }
  })
  .catch((error) => console.log(error));

  return res;
}

```

Вывод:

В ходе выполнения работы я создал свой сервер и связал его с бд, а также связал фронт с беком через получение данных с помощью запросов к API.