Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное общеобразовательное учреждение высшего образования

Уфимский университет науки и технологий

Факультет информатики и робототехники

Кафедра вычислительной математики и кибернетики

Отчет к лабораторной работе №5

по дисциплине «Web-технологии»

«Cookie-файлы, сессии и авторизация в веб-приложении»

Выполнили:

Студенты группы МО-321

Шарипов А.Г., Дементьев Д.В.,

Проверил:

Юдинцев Б.С.

Уфа-2023

**Задание:**

1. Создать таблицу для хранения аккаунтов пользователей.
2. Создать страницу регистрации пользователя. Пароль зарегистрированного пользователя должен храниться в БД в хешированном виде.
3. Создать страницу авторизации пользователя и реализовать механизм доступа к REST API и странице с формой только для авторизированных пользователей, используя сессии и cookie.
4. Вывести на странице с формой для отправки запросов вывод всех запросов, созданных текущим зарегистрированным пользователем.

**Теоретическая часть.**

**HTTP Basic Authentication**

HTTP предоставляет набор инструментов для разграничения доступа к ресурсам и авторизацией. Самой распространённой схемой HTTP авторизации является "Basic" (базовая) авторизация.

Опишем общий механизм HTTP авторизации. Протокол передачи гипертекста RFC 7235 определяет средства HTTP авторизации, которые может использовать сервер для запроса у клиента аутентификационной информации. Сценарий запрос-ответ подразумевает, что вначале сервер отвечает клиенту со статусом 401 (Unauthorized) и предоставляет информацию о порядке авторизации через заголовок WWW-Authenticate (en-US), содержащий хотя бы один метод авторизации. Клиент, который хочет авторизоваться, может сделать это, включив в следующий запрос заголовок Authorization с требуемыми данными. Обычно, клиент отображает запрос пароля пользователю, и после получения ответа отправляет запрос с пользовательскими данными в заголовке Authorization. В случае базовой авторизации, обмен должен вестись через HTTPS (TLS) соединение, чтобы обеспечить защищённость.

**OAuth**

OAuth — открытый протокол (схема) авторизации, который позволяет предоставить третьей стороне ограниченный доступ к защищённым ресурсам пользователя без необходимости передавать ей (третьей стороне) логин и пароль. Для улучшения защиты может быть использована аутентификация в два шага (например, Google Authenticator), когда для подтверждения задействуется другой сервис пользователя (например, когда для аутентификации на веб-сайте требуется ввести кодовое слово, отправляемое на мобильный телефон).

Ключевая особенность применения OAuth заключается в том, что, если пользователь имеет хорошо защищённый аккаунт, то с его помощью и технологии OAuth он может пройти аутентификацию на других сервисах, и при этом ему не требуется раскрывать свой основной пароль.

**JSON Web Token (JWT)**

JSON Web Token (JWT) — это открытый стандарт (RFC 7519) для создания токенов доступа, основанный на формате JSON. Как правило, используется для передачи данных для аутентификации в клиент-серверных приложениях. Токены создаются сервером, подписываются секретным ключом и передаются клиенту, который в дальнейшем использует данный токен для подтверждения своей личности.

**Ход работы:**

 Валидация пользователя по логину и паролю:

exports.login\_user = function(req, res) {

*// Получаем логин и пароль из данных формы*

    var login = req.body.loginField;

    var password = req.body.passField;

*// Ищем пользователя в БД*

    dbcontext.query(

        'SELECT \* FROM logins WHERE "username" = :username',

        {

            replacements: { username: login },

            type: dbcontext.QueryTypes.SELECT

        }

    ).then(data => {

        var user = data[0]

*// если пользователь не найден переадресуем на страницу /login*

        if (!user) {

            res.redirect('/login');

*// если пользователь найден, проверяем пароль*

        } else if (!bcrypt.compareSync(password, user.password)) {

*// если пароль не прошел проверку, переадресуем на страницу /login*

            res.redirect('/login');

        } else {

*// иначе регистрируем сессию пользователя (записываем логин пользователя в параметр user)*

            req.session.user = user.username;

            req.session.userId = user.id;

*// высылаем сессионную cookie AuthToken с логином*

            res.cookie('AuthToken', user.username);

            res.redirect('/contact');

        }

    })

    .catch(err => {

*// в случае исключения возвращаем код 500 + json-ответ с ошибкой*

        res.status(500).json({ message: err.message });

    });

}

Функция для проверки авторизации пользователя:

exports.sessionCheck = (req, res, next) => {

*// Если не установлен параметр сессии user или значение cookie 'AuthToken' не равно логину пользователя*

    if (!req.session.user || req.cookies['AuthToken'] != req.session.user) {

*// переадресуем на страницу /login*

        res.redirect('/login');

    } else {

*// иначе исполняем следующую функцию обработчика*

        next();

    }

};

Регистрация пользователя:

exports.register\_user = function(req, res) {

*// Проверяем полученные данные на наличие обязательных полей*

    if (!req.body.loginField || !req.body.emailField || !req.body.passField) {

        res.status(400).json({ message: "The data entered are not correct!" });    *// если данные не найдены, возвращаем HTTP-код 400*

    return;

    }

*// Создаем хеш пароля с солью*

    const salt = bcrypt.genSaltSync();

    var hashed = bcrypt.hashSync(req.body.passField, salt);

*// Создаем пользователя в БД*

    dbcontext.query(

        'INSERT INTO logins (`username`, `password`, `email`) VALUES (:username, :password, :email)',

        {

            replacements: { username: req.body.loginField, password: hashed, email: req.body.emailField },

            type: dbcontext.QueryTypes.INSERT

        }

    )

    .then(result => {

        console.log(`Registered as ${req.body.loginField}`);

*// в случае успешной записи переадресуем пользователя на страницу авторизации*

        res.redirect('/login');

    })

    .catch(err => {

*// в случае исключения возвращаем код 500 + json-ответ с ошибкой*

        res.status(500).json({ message: err.message });

    });

};

**Скриншоты страниц**

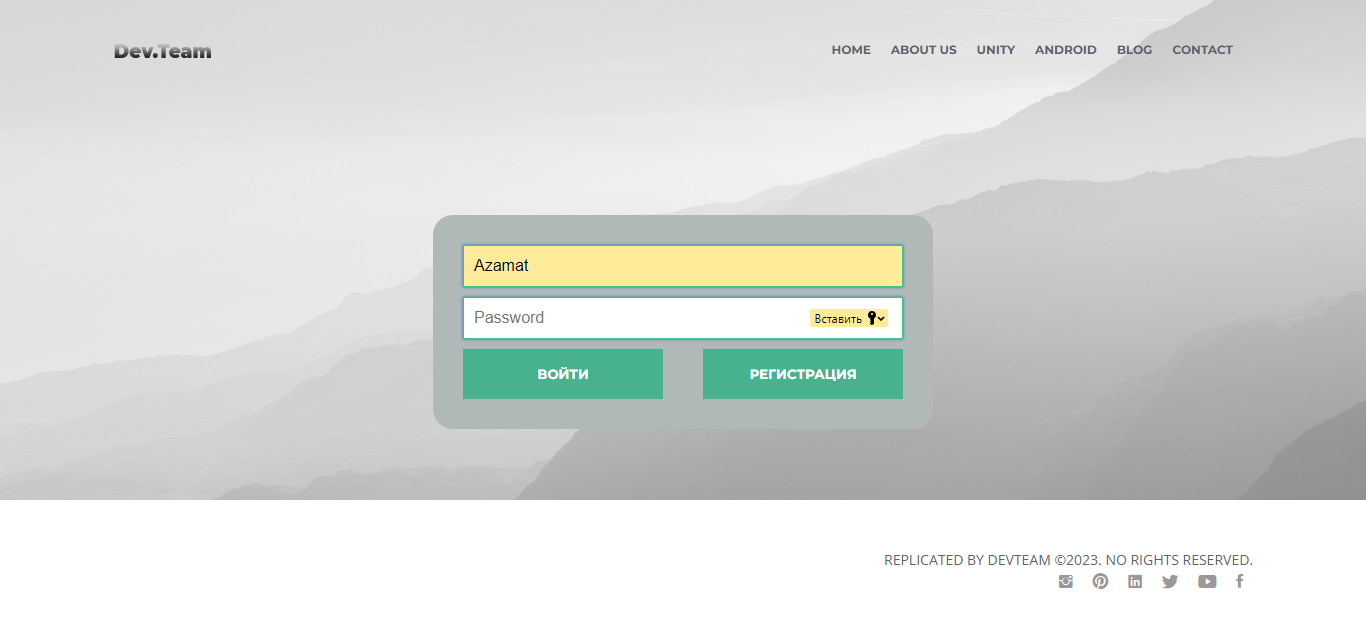
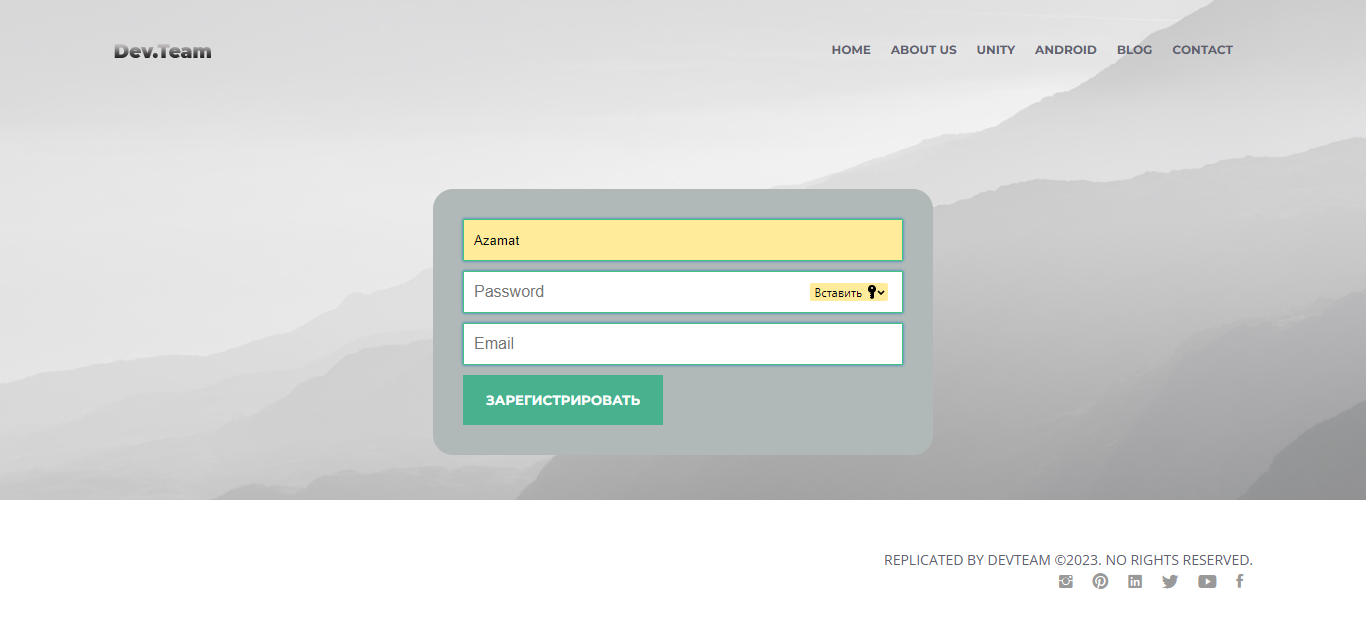


Рисунок 1. Страница авторизации

Рисунок 2. Форма регистрации

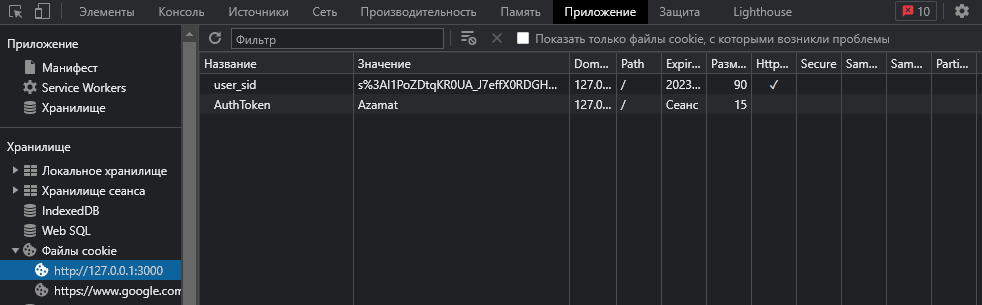


Рисунок 3. Cookies

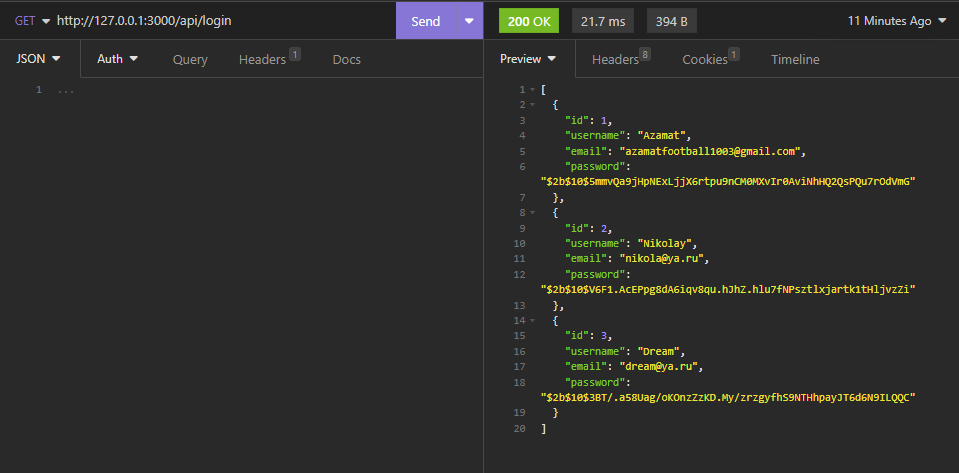


Рисунок 4. Запрос получения всей информации в БД

**Вывод**: в ходе выполнения лабораторной работы изучена работа с сессиями, а также созданы страницы регистрации и авторизации.

Ссылка на гит

[**https://github.com/DemDanV/Web\_lab**](https://github.com/DemDanV/Web_lab)