Лабораторная работа 08

СТРWП

**Задание 01**

1. Разработайте приложение **08-01**, представляющее собой EXPRESS-HTTP-сервер, прослушивающий порт ***3000***.
2. Приложение **08-01** предназначено для предоставления аутентифицированного http-доступа к ресурсу.
3. Приложение **08-01** должно обеспечивать FORMS-аутентификацию на основе сессий с помощью пакета **passport**.
4. Приложение **08-01** должно реагировать на запросы к следующим ресурсам.

|  |  |
| --- | --- |
| GET /login | Вернуть форму для ввода имени пользователя и пароля |
| POST /login | Производить аутентификацию  В случае успешной аутентификации переадресовывать на /profile, иначе – на /login |
| GET /logout | Отключить аутентифицированный доступ к ресурсу |
| GET /profile | Если пользователь аутентифицирован, то отправлять информацию об аутентифицированном пользователе  При попытке неаутентифицированного доступа выполнить переадресацию на /login |
| Остальные URI | Вернуть ответ со статусом 404 |

1. Список разрешенных данных для аутентификации (имя, пароль и т.д.) в приложении должен быть представлен в виде json-объекта, доступ к которому осуществляется с помощью функции **require**.
2. Тестирование выполнить с помощью браузера.

**Задание 02**

1. Разработайте приложение **08-02**, представляющее собой EXPRESS-HTTP-сервер, прослушивающий порт ***3000***. Приложение **08-02** должно использовать FORMS- аутентификацию на основе jwt-токена (только access) с помощью пакета **jsonwebtoken**.
2. Приложение **08-02** должно реагировать на запросы к следующим ресурсам.

|  |  |
| --- | --- |
| POST /login | Производить аутентификацию, принимать данные для аутентификации в формате JSON.  В случае успешной аутентификации возвращать access-токен, иначе ответ со статусом 401 и сообщением «Неправильный логин или пароль» |
| GET /profile | Если токен валидный, то отправлять информацию о текущем пользователе, иначе возвращать ответ со статусом 401 и сообщением «Невалидный токен». |
| Остальные URI | Вернуть ответ со статусом 404 |

1. Список разрешенных данных для аутентификации (имя, пароль и т.д.) хранить в json-файле.
2. Установить время жизни access-токена 10 минут.
3. Access-токен передавать в заголовке запроса Authorization: Bearer <access-token>.
4. Тестирование выполнить с помощью Postman’а.

**Задание 03**

1. Что такое аутентификация?
2. Что такое токен? Какие бывают?
3. Опишите общую схему FORMS-аутентификации на основе сессий.
4. Опишите общую схему FORMS-аутентификации на основе токенов.

1. \*\*Аутентификация\*\* - это процесс проверки подлинности пользователя или системы, чтобы удостовериться в его или ее идентичности. В контексте веб-приложений аутентификация обычно включает в себя проверку учетных данных пользователя, таких как имя пользователя и пароль, перед тем как предоставить доступ к защищенным ресурсам.

2. \*\*Токен\*\* - это уникальная строка данных, которая выдается пользователю или клиенту после успешной аутентификации и предоставляет доступ к ресурсам или услугам. В зависимости от конкретного случая токены могут быть различных типов:

- \*\*JWT (JSON Web Token)\*\*: Компактный и самодостаточный способ представления информации между двумя сторонами в виде JSON-объекта. Обычно содержит идентификатор пользователя, срок действия, а также другую метаинформацию.

- \*\*Bearer Token\*\*: Это тип токена аутентификации, который представляет собой просто строку, которая передается в заголовке HTTP-запроса. Используется вместе с протоколами аутентификации, такими как OAuth 2.0.

- \*\*Session Token\*\*: Это временный идентификатор, который создается сервером и ассоциируется с текущей сессией пользователя. Часто используется в схеме аутентификации на основе сессий.

3. \*\*Схема FORMS-аутентификации на основе сессий\*\*:

- Пользователь вводит учетные данные (имя пользователя и пароль) на странице входа (login page) и отправляет их на сервер.

- Сервер проверяет эти учетные данные. Если они верны, создается сессия для пользователя.

- Во время создания сессии на сервере генерируется уникальный идентификатор сессии (обычно в виде токена), который ассоциируется с пользователем и сохраняется в хранилище сессий (например, в базе данных или кэше).

- Клиенту возвращается ответ с установленным куки, содержащим идентификатор сессии или сам сессионный токен.

- В последующих запросах клиент автоматически отправляет этот сессионный токен на сервер в заголовке запроса, что позволяет серверу идентифицировать пользователя.

4. \*\*Схема FORMS-аутентификации на основе токенов\*\*:

- Пользователь вводит учетные данные на странице входа (login page) и отправляет их на сервер.

- Сервер проверяет эти учетные данные. Если они верны, создается JWT (JSON Web Token) с идентификатором пользователя в качестве полезной нагрузки.

- JWT возвращается клиенту в ответ на успешную аутентификацию.

- Клиент сохраняет JWT, например, в локальном хранилище или куках браузера.

- В последующих запросах клиент отправляет JWT на сервер в заголовке `Authorization`, используя схему аутентификации "Bearer".

- Сервер проверяет JWT на подлинность, а затем извлекает идентификатор пользователя из полезной нагрузки, чтобы определить, какому пользователю принадлежит токен.