Лабораторная работа 12

ПвИ

**Задание 01**

1. Разработайте приложение **12-01**, представляющее собой EXPRESS-HTTP-сервер, прослушивающий порт ***3000***.
2. Приложение **12-01** предназначено для предоставления аутентифицированного http-доступа к ресурсу.
3. Приложение **12-01** должно обеспечивать FORMS-аутентификацию на основе сессий с помощью пакета **passport**.
4. Приложение **12-01** 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 /resource, иначе – на /login |
| GET /logout | Отключить аутентифицированный доступ к ресурсу |
| GET /resource | Если пользователь аутентифицирован, то отправить сообщение RESOURCE и информацию об аутентифицированном пользователе |
| Остальные URI | Вернуть ответ со статусом 404 |

1. При попытке неаутентифицированного доступа возвращать ошибку с 401 статусом и пояснительным сообщением.
2. Список разрешенных для аутентификации данных (имя, пароль и т.д.) в приложении должен быть представлен в виде json, доступ к которому осуществляется с помощью функции **require**.
3. Тестирование выполнить с помощью браузера и Postman’а.

**Задание 02**

1. Разработайте приложение **12-02**, представляющее собой EXPRESS-HTTP-сервер, прослушивающий порт ***3000***. Приложение **12-02** должно использовать FORMS- аутентификацию на основе jwt-токенов (access и refresh) с помощью пакета **jsonwebtoken**.
2. Приложение **12-02** 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 возвращать пару токенов (access и refresh) и переадресовывать на /resource, иначе – на /login |
| GET /refresh-token | Принимать refresh-токен, проверять его.  Если токен валидный, то формировать новую пару access и refresh токена, помещать старый refresh-токен в «черный» список или новый refresh-токен в «белый» список (на выбор).  Если токен не валидный, то возвращать статус 401 |
| GET /logout | Отключить аутентифицированный доступ к ресурсу |
| GET /resource | Если токен валидный, то отправить сообщение RESOURCE и информацию об аутентифицированном пользователе |
| Остальные URI | Вернуть ответ со статусом 404 |

1. При попытке неаутентифицированного доступа возвращать ошибку с 401 статусом и пояснительным сообщением.
2. Список разрешенных данных для аутентификации (имя, пароль и т.д.) хранить в базе данных.Для работы с ней использовать пакет **sequelize** или **prisma**.
3. Установить время жизни access-токена 10 минут, а refresh-токена – 24 часа.
4. Access-токен сохранять в cookie. При установке указать флаги HttpOnly=true (для защиты от XSS-атак), SameSite=Strict (для защиты от CSRF-атак).
5. Refresh-токен хранить тоже в cookie. Дополнительно использовать флаг path.
6. Для «черного»/«белого» списка refresh-токенов использовать Redis.
7. Добавить возможность регистрации.
8. Тестирование выполнить с помощью браузера и Postman’а.

**Задание 03**

1. Что такое токен? Какие бывают?

Токен — это набор символов, который служит средством идентификации или проверки подлинности. Токены широко используются в информационной безопасности для аутентификации пользователей, а также для контроля доступа к ресурсам и сервисам.

Виды токенов:

Access Tokens: Используются для доступа к защищенным ресурсам. Обычно имеют короткое время жизни.

Refresh Tokens: Используются для получения новых Access Tokens, когда текущий истекает. Обычно имеют более долгое время жизни.

ID Tokens: Содержат информацию о пользователе. Часто используются вместе с OAuth 2.0 и OpenID Connect.

CSRF Tokens: Используются для предотвращения атак типа CSRF (Cross-Site Request Forgery), включаясь в запросы, которые изменяют состояние системы.

1. Опишите общую схему FORMS-аутентификации на основе сессий.

Логин: Пользователь вводит свои учетные данные (логин и пароль) на веб-форме.

Проверка учетных данных: Сервер проверяет учетные данные пользователя в базе данных.

Создание сессии: Если учетные данные верны, сервер создает сессию для пользователя и сохраняет ее в своей базе данных или в памяти.

Отправка идентификатора сессии: Сервер отправляет клиенту идентификатор сессии, обычно в виде куки-файла.

Хранение идентификатора сессии: Браузер сохраняет куки с идентификатором сессии.

Использование сессии: При последующих запросах браузер автоматически отправляет куки с идентификатором сессии. Сервер использует этот идентификатор для подтверждения аутентификации пользователя.

1. Опишите общую схему FORMS-аутентификации на основе токенов.

Логин: Пользователь вводит свои учетные данные на веб-форме.

Проверка учетных данных: Сервер проверяет учетные данные пользователя.

Генерация токена: Если учетные данные верны, сервер генерирует токен (например, JWT) и отправляет его пользователю.

Хранение токена: Клиент (например, веб-приложение) сохраняет токен на стороне клиента (например, в localStorage).

Отправка токена с запросами: При последующих запросах к серверу клиент отправляет токен в заголовке запроса для аутентификации.

Валидация токена: Сервер валидирует токен при каждом запросе и, если токен действителен, предоставляет доступ к защищенным ресурсам.