Лабораторная работа 09

СТРWП

**Задание 01**

1. Разработайте приложение **09-01**, представляющее собой EXPRESS-HTTP-сервер, прослушивающий порт ***3000***.
2. Приложение **09-01** предназначено для предоставления аутентифицированного доступа к ресурсу.
3. Приложение **09-01** должно реализовывать стратегию аутентификации через социальные сети, то есть OAuth с помощью пакета **passport** (провайдер любой: *Google*, *VK*, *Twitter*, *Facebook*, *Github* и т.д.).
4. Приложение **09-01** должно реагировать на запросы к следующим ресурсам.

|  |  |
| --- | --- |
| GET /login | Вернуть страницу с кнопкой для аутентификации через выбранную соцсеть. |
| GET /auth/{provider\_name}/ | Производить аутентификацию. Перенаправлять пользователя на на сайт выбранного провайдера для аутентификации. По завершении провайдер должен переадресовать пользователя обратно в приложение по адресу /auth/{provider\_name}/callback (redirect URL). |
| GET /auth/{provider\_name}/callback | Переадресовать пользователя на /resource в случае, если доступ был одобрен, в противном случае – на /login. |
| GET /logout | Отключить аутентифицированный доступ к ресурсу. |
| GET /resource | Если пользователь аутентифицирован, то отправить сообщение RESOURCE и информацию об аутентифицированном пользователе (идентификатор, имя пользователя и др.).  При попытке неаутентифицированного доступа выполнить переадресацию на /login. |
| Остальные URI | Сообщение со статусом 404 |

1. Тестирование выполнить с помощью браузера.

**Задание 02** Ответьте на следующие вопросы

1. Опишите общую схему OAuth-аутентификации.
2. **Регистрация приложения**: Владелец ресурса (например, сервис, предоставляющий доступ к данным пользователя) регистрирует свое приложение на сервере авторизации. В процессе регистрации создается идентификатор клиента (client ID) и секретный ключ (client secret), которые приложение будет использовать для аутентификации.
3. **Запрос разрешения**: Когда пользователь пытается получить доступ к ресурсу через приложение, оно направляет его на страницу сервера авторизации с запросом разрешения. Запрос может содержать информацию о необходимых разрешениях и другие параметры.
4. **Аутентификация пользователя**: Пользователь аутентифицируется на сервере авторизации, вводя свои учетные данные (логин и пароль) или используя другие методы аутентификации, такие как социальные сети или одноразовые токены.
5. **Предоставление разрешения**: После успешной аутентификации сервер авторизации отображает пользователю запрос разрешения, в котором указаны запрашиваемые разрешения и информация о приложении, запрашивающем доступ. Пользователь может согласиться или отказаться в предоставлении доступа.
6. **Выдача временного кода**: Если пользователь согласен предоставить доступ, сервер авторизации выдает временный код (authorization code) приложению.
7. **Обмен временного кода на токен доступа**: Приложение использует временный код для обмена его на токен доступа (access token). Для этого оно отправляет запрос на сервер авторизации, включая временный код и свои учетные данные (client ID и client secret).
8. **Получение токена доступа**: Если запрос на обмен временного кода на токен прошел успешно, сервер авторизации выдает приложению access token.
9. **Доступ к ресурсу**: Приложение может использовать полученный access token для доступа к защищенным ресурсам на сервере ресурсов. Этот токен передается в каждом запросе к ресурсам как механизм авторизации.
10. **Обновление токена доступа (не всегда)**: Access token имеет ограниченное время жизни. По истечении этого времени он становится недействительным. Чтобы продлить доступ, приложение может запросить обновление токена, отправив запрос на сервер авторизации с использованием refresh token (если он был выдан).

