FORMS-аутентификация является одним из способов аутентификации пользователей веб-приложений. В случае использования сессий или токенов, схема остаётся примерно такой же, но меняется механизм хранения состояния аутентификации.

### Схема FORMS-аутентификации с использованием сессий:

1. \*\*Пользователь отправляет данные аутентификации (обычно логин и пароль) на сервер через форму HTML.\*\*

2. \*\*Сервер проверяет правильность введенных данных. Если данные верны, сервер создает сессию для пользователя.\*\*

3. \*\*Идентификатор сессии обычно сохраняется в куки браузера или передается через URL.\*\*

4. \*\*При каждом запросе от клиента, сервер проверяет наличие сессии и ее актуальность. Если сессия действительна, сервер позволяет пользователю получить доступ к защищенным ресурсам.\*\*

5. \*\*Сессия обычно хранится на сервере, в памяти или в базе данных, и имеет время жизни.\*\*

### Схема FORMS-аутентификации с использованием токенов:

1. \*\*Пользователь отправляет данные аутентификации на сервер через форму HTML.\*\*

2. \*\*Сервер проверяет правильность введенных данных. Если данные верны, сервер создает токен аутентификации.\*\*

3. \*\*Токен передается обратно клиенту. Он может быть встроен в заголовок запроса или сохранен в куки.\*\*

4. \*\*При каждом запросе от клиента, токен передается обратно серверу. Сервер проверяет подлинность токена и разрешает доступ, если токен действителен.\*\*

5. \*\*Токен обычно имеет ограниченное время жизни и может содержать информацию о пользователе (например, идентификатор пользователя, роли и т.д.).\*\*

Существенное отличие между сессиями и токенами заключается в том, где хранится информация о состоянии аутентификации. В случае сессий, это хранится на сервере, в то время как токены хранятся на клиенте и передаются обратно серверу для каждого запроса. Оба подхода имеют свои преимущества и недостатки, и выбор зависит от конкретных требований и контекста приложения.