**Текст для выступления Калашников А.С. ИУК4-52Б**

**Актуальность** темы курсового проекта обуславливается активным уходом с Российского рынка иностранных веб-приложений.

**Объектом** работы является веб-разработка, **предмет** – личный кабинет веб-приложения.

**Цель** проекта – разработка личного кабинета пользователя для документирования сети в веб-приложении

Для достижения поставленной цели решаются следующие задачи:

1. Выполнить анализ предметной области;
2. Определить оптимальную структуру системы;
3. Осуществить выбор средств реализации программного продукта, соответствующего выбранной структуре;
4. Разработать веб-приложение;
5. Осуществить тестирование компонентов;
6. Разработать сопроводительную документацию.

**Работа с токенами**

Когда клиенту нужно отправлять запросы на микросервис он в заголовке вместе с body прикладывает jwt-токен, и микросервирс, который должен обработать по какому-то URL какое-то действие он из заголовка забирает токен и проверяет подпись поскольку у него есть ключ, которым он подписан, если же токен совпадает, то мы можем выполнять действия.

Получается, кто обладает jwt-токеном может от имени какого-то пользователя выполнять произвольные действия в системе. Соответственно может сложится такая ситуация, когда какой-то злоумышлениик посредник исходя из перехваченного пакета может выцепить данный jwt-токен и использует его, чтобы скомпрометировать как-то пользователя. Данную проблему можно решить наиболее популярным и наиболее простым подходом при помощи refresh-токенов. Когда пользователь авторизируется на нашем сервере он в ответ помимо jwt-токена получает refresh-токенов. Refresh-токен представляет из себя набор символов – уникальный ключ.

Сервер Авторизации сохраняет у себя что у этого пользователя этот refresh-токен при этом access-токен, который нужен для доступа к нашему закрытому ресурсу он делается короткоживущим 10 минут и в нем указываем время его жизни. Соответственно имея access-токен, клиент может отправлять запросы к нашему серверу в течение 10 минут. При выполнении запроса к серверу по истечению времени access-токен отправляется уже просроченный на наш сервер ресурсов. Наш сервер ресурсов расшифровывает пришедшие токены и может проверить время жизни и если токен просрочен вместо того чтобы вернуть данные он вернет ошибку клиенту, что токен просрочен. В данном случае клиент передает jwt-токен и refresh-токен обратно серверу авторизации. Refresh-токен же нигде и никуда не передавался он один раз отдался клиенту и клиент его у себя хранит поэтому он не может утечь. Передавая серверу jwt-токен и refresh-токен серверу авторизации для определения пользователя. Сервер авторизации сравнивает совпадает ли refresh-токен, который хранит у себя в базе. Если они совпадают, то сервер генерирует новый access-токен и refresh-токен и дальше в течение 10 минут клиент общается с сервером при помощи access-токен.

Если злоумышленник захватывает access-токен, он в течение жизни токены может выполнять какие-то злодейства, поэтому access-токен делается короткоживущим. По истечению 10 минут он получит ошибку, ему нужно будет обновиться, но он не сможет получить новый jwt-токен от сервиса, так как не имеет refresh-токен, чтобы совершить новые злодейства ему нужно будет снова захватить новый access-токен

Такой механизм достаточно широко распространенный позволяет обезопасить в целом наше приложение

Для того чтобы каждый микросервис проверял подписи нашего jwt-токена, но ключом которым мы подписываем, может обладать только один микросервис, который проводит авторизацию, то здесь выполняется ассимитричный алгоритм шифрования. Ассимитричный алгоритм шифрования – где для шифрования и для дешифрования данных используются разные пары ключей, они взаимосвязаны, но по одному нельзя получить второй. Шифрование данных мы проводим одним ключом закрытым, а расшифровываем другим открытым. В данном веб-приложении мы использовали алгоритм RSA. Единственный микросервис, который используется для авторизации обладает секретным ключом, которым будут подписываться токены. Также он обладает публичным ключом, которым можно будет расшифровать эти токены, проверить валидность пришедших данных.

Сначала клиент авторизуется в нашей системе наш микросервис генерирует jwt-токен подписывает его секретным ключом и отправляет клиенту refresh-токен. Дальше наш клиент может взаимодействовать с произвольным микросервисом, который обладает публичным ключом с помощью которого можно проверить наш токен. После чего на эти данные приходит либо какойто ответ, либо сообщается ошибка о просрочке jwt-токена. При просрочке наш клиент обращается к серверу и отправляет jwt-токен и refresh-токен и в ответ получает новый jwt-токен и refresh-токен

**Механизм смены языков**

От интерфейса реализуются объекты – языки. Каждый язык соответствует одинаковый набор переменных типа string. Каждая страница имеет свой массив с набором переменных.

Также помимо интерфейса был создан тип данных Language и список данного типа, в котором хранятся все языки, которые есть в нашем веб-приложении.   
 Весь перевод слов хранится в словаре Для каждого языка имеется свой словарик.

Переключение языка осуществляется с помощью изменения состояния «language» типа «Language» при изменении состояния, начальное значение равное «en» меняется на выбранный пользователем язык из созданного списка языков.

**Заключение**

В ходе выполнения курсового проекта были получены обширные навыки в сфере разработки веб-приложений, работы с ООП, проектирования и реализации продукта.

Результатом выполнения курсовой работы была создана система, позволяющая производить регистрацию и авторизацию пользователей, производить настройки аккаунта. Благодаря jwt и RSA система в дальнейшем может взаимодействовать с любыми микросервисами. Результат соответствует всем заранее определенным требованиям технического задания. Внедрение данной системы в приложение позволит увеличить безопасность и предотвращает несанкционированный доступ к ресурсам.

Поскольку поставленная задача была выполнена, в дальнейшем планируется перейти к использованиям баз данных для хранения информации.