План Б: накидать кнопок, текстовых полей, чтобы оно выглядело как что-то работающее. Необязательно, чтобы оно работало!!! Начать это реализовывать с 02.06.2021. Сразу делать слайды. Описывать всё, что там есть, всё, что понимаю. Прокатит. Код всё равно показывать никому не надо.

https://www.youtube.com/watch?v=jLJVMD2GBAA:

# Identity

Это фреймворк для работы с аутентификацией.

Есть два подхода для реализации:

1 автоматическая регистрация. Можно сделать на сайте Azure portal page for app registration.

2 ручная настройка.

https://www.youtube.com/watch?v=Uco05GknIUU

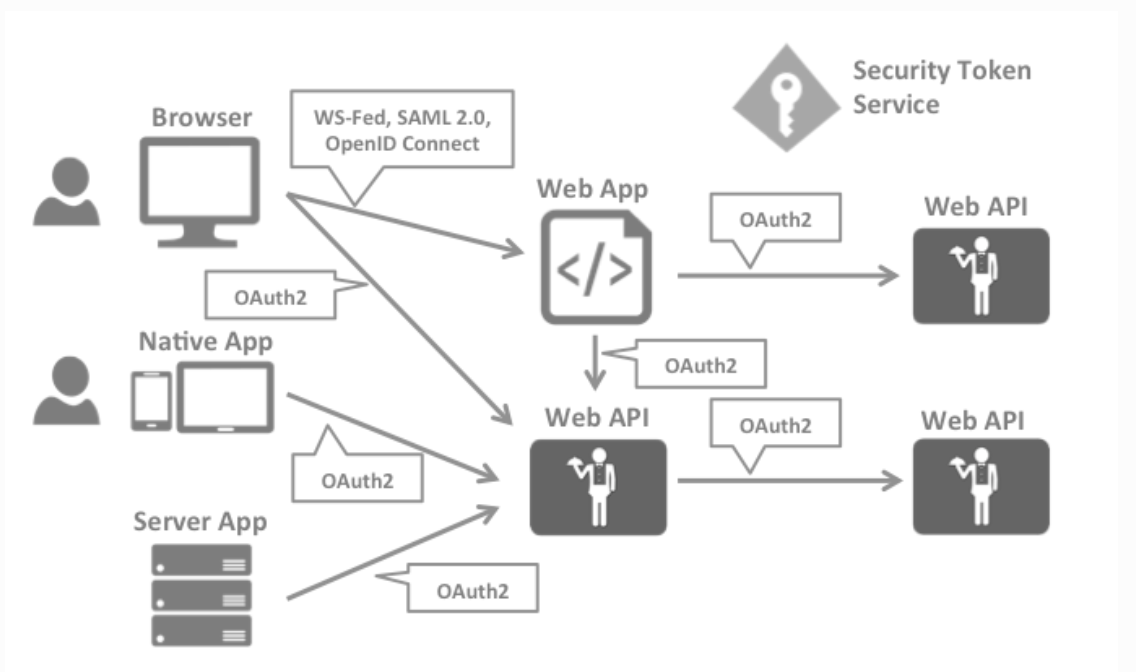
Части программ общаются друг с другом с помощью токенов каких-то… security token service. Identity был создан(несмотря на наличия множества уже существующих решений) для того, чтобы придать гибкости приложению, чтобы пользователи могли выполнять больше действия так, как им хочется. Это централизованное решение. Имеется в виду, что пользователь входит один раз, получает токен и вносит нужные ему изменения.

Identity server доставляется с помощью NuGet пакета. Ответ на вопрос, как Identity связывает ПО в том, что он преимущественно реализуется как промежуточное ПО. Идея в том, что ты создаёшь хост с этим Identity Server, заходишь в NuGet, прикрепляешь промежуточное ПО в очередь запроса(pipeline). В итоге работа данного промежуточного ПО(Identity Server) – добавить endpoints для ранее упоминавшихся протоколов(open ID connect, Oauth2). Т.е. идея в том, что я не беру своё приложение и добавляю в него Identity Server, а в том, что я делаю отдельный Хост, т.к. реализую я STS(Security Token Service, Сервис токенов для безопасности). Т.е это новое веб-приложение, работающее как бы параллельно с моим основным приложением.

Для этого кстати есть шаблоны. Токены – это то, что мы доставляем назад в приложения для доказательства аутентичности пользователя и токены – это то, что приложение будет использовать для обращения к API. Токен содержит информацию о пользователе и он также подписан, т.е. есть цифровая подпись, созданная с помощью STS. В этом ценность открытых систем, в возможности использовать подобные настройки(управление временем сессии пользователя и другие вещи). А точнее получается, что Identity Server – это фреймворк, позволяющий построить свою STS. Т.е. это не просто STS из коробки, это также набор инструментов для построения своей собственной STS. Т.е. об Identity Server можно говорить как о фреймворке, это библиотека, которую подключают и строят новое приложение со своим поведением. Это поведение в итоге соотвтетсвует нужным разработчикам требованиям.

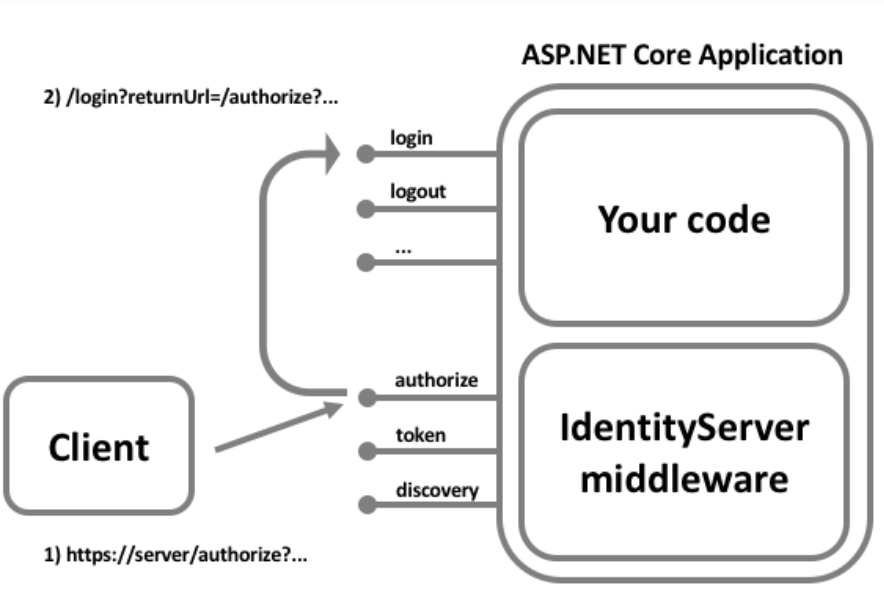
Если сравнивать с другими подобными решениями, то для Java, например популярна Pink Federate. Azure Active Directory – облачное STS, предлягаемая компанией Майкрософт. Также есть OCTA, AUTH0. Т.е., видимо, можно создать свой хост, а можно доверить это облачным сервисам.

В итоге мы имеем централизованное управление с одиночным вхождением пользователя в систему(и возможностью проводить множество операций после этого вхождения).



Т.о. в итоге мы будем управлять сервисом безопасности с токенами. Web-api и web-app, которые я создам будут общаться друг с другом посредством этого сервиса. Всё это будет работать в соответсвии с Open Id connect & Oauth2 protocol. Красота решения в том, что Identity можно использовать в программе, но если потребуется, то можно будет его сменить на подобный сервис. Identity server будет сервисом для использования токенов.

В Visual Studio можно зайти в файл – новый проект – Identity Server. Т.о. есть промежуточное ПО(middleware), реализующее эти protocol endpoints. Далее наш клиент в виде приложения типа Model View Controller(приложение, желающее знать, кем является наш пользователь) будет говорить в виде протокола с Identity Server.



Также можно использовать и другое ПО для авторизации. На рисунке клиент шлёт запрос на авторизацию, Identity Server пересылает его на настроенную пользователем страничку авторизации. Клиент вводит все необходимые данные. Потом ты опять пересылаешь клиента(пользователя) на конечную точку авторизации(authorize endpoint) в нижней части картинки. Конечная точка авторизации – это протокол конечной точки, где пользователю показывается пользовательский интерфейс. Up to 20: 00

Можно действовать так: создаю отдельное приложение в проекте, прописываю в нём в Startup.cs в методе Configure app.UseIdentityServer(); после app.UseStaticFiles(); при этом Identity Server подключён как NuGet package. Пока Identity Server не знает ничего про наших клиентов и пользователей. Его надо научить этому.

В целом asp.net Identity – это библиотека или фреймворк, созданная для управления базой данных, содержащей информацию об аутентификации пользователей. Эта информация в первую очередь: пароль. Также там много функциональности реализовано, такой как подтверждение почты, двухфакторная аутентификация, подтверждение номера мобильного телефона и т.п. Также удобно думать об asp.net как о библиотеке, говорящей с базой данных, содержащей наших пользователей. Однако Identity не реализует протоколы безопасности(?). app.UseIdentityServer также добавляет ПО для работы с аутентификацией посредством файлов куки в наше Хост приложение. Однако можно и самому реализовать эту функциональность.

Половина Identity Server настраивается с помощью системы внедрения зависимостей. Identity Server предоставляет набор сервисов, которые доступны для внедрения как зависимости. Для подключения этих сервисов нужно в классе Startup.cs, в методе ConfigureServices добавить services.AddIdentityServer(); это сконфигурирует сервисы ядра. После этого нужно будет вызвать некоторые расширяющие методы, чтобы настроить нужную функциональность.

Identity Server очень гибкий и был сконструирован для того, чтобы быть расширяемым. Поэтому нужны конфигурационные данные, нужные IdSe. Причём нет строгого требования по тому, где хранить эти данные. Поэтому список MVC приложений, список API может быть в базе данных, может быть в файле JSON или можно даже его прописать в виде кода в приложении.

Следующий расширяющий метод регистрирует наше хранилище в памяти для этих конфигурационных данных(с тестовыми пользователями, клиентами и API). Services.AddIdentityServer().AddTestUsers(TestUsers.Users); AddTestUsers – наш расширяющий метод, который подключается к списку пользователей, прописанных в коде(в файле TestUsers 28:56).

32:50

На 43:00 примерно вроде объясняется scope для доступа клиентов к полям пользователей…

1:33:10 здесь про аутентификацию через гугл