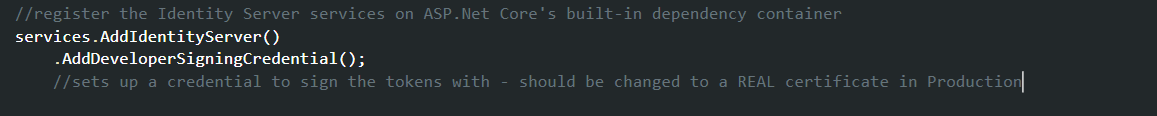
1. Create a new ASP.NET Core Web Application with the suffix “.IDP”
2. Install IdentityServer4 via npm
3. We have to configure this
4. Go to the Startup class in **Startup.cs**
5. 
6. Add test users, resources and clients (add a class for this)
7. **How do we get claims to be returned in an identity token?**
8. Claims are related to scopes
9. **IEnumerable** type is used to return any kind of list that is able to be iterated (array or actual list type). Return types of **IEnumerable** is just a basic way of returning lists and arrays. (Lists that don’t inherit from IEnumerable cannot use “foreach” loop to iterate through them)

* After we have our UI, in OpenID connect, there is **no encryption** of data that is sent over the wire. Tokens are signed but **not encrypted.**

**Summary**

* Different types of clients

1. A **confidential client** can safely store secrets (server-side application)
2. A **public client** can’t safely store secrets (JS-based, or mobile-apps)

* Depending on the **client type,** a different flow is advised
* A **flow** can be seen as how an application can achieve authentication and authorization
* Different types of flows

1. Authorization
2. Implicit
3. Hybrid

* Flows use one or more endpoints

1. **Authorization endpoint** (IDP) - Used by the client application to obtain authentication and/or authorization
2. **Token endpoint** (IDP) - Used by the client application to programmatically request tokens
3. **Redirection endpoint** (lives at the client level) - where the tokens are delivered to from the authorization endpoint

**TLS is a requirement for these systems!**

**The Hybrid Flow**

1. Web App creates Authentication request with “response\_type = code id\_token” and sends it to the IDP authorization endpoint
2. At the IDP, the user authenticates (for example, by providing username + password)
3. The user is authenticated at the level of the IDP
4. The IDP optionally asks the user for consent (it will ask “do you want to allow the information to get your profile information?”)
5. IDP sends back data via **form\_post** or **redirect\_uri** with the authorization code and the id\_token
6. The web client validates the token that it got from the authorization endpoint
7. The middleware calls the token endpoint and asks for **authorization code, clientid and clientsecret**
8. In response, the middleware gives back an id\_token, which is also validated
9. The middleware can also return access tokens and refresh tokens as extra layers of security
10. One of the validation checks for this id\_token is to check if the user matches the user of the previous token
11. Middleware extracts the user’s identifier from the id\_token

* By default, IdentityServer doesn’t include identity claims (like first\_name, last\_name) (except sub) in the identity token unless we specifically ask for this
* Not including these claims in the id\_token keeps it small, avoiding browser URI length restrictions

**UserInfo Endpoint (IDP level)**

* Used by the client application to request additional user claims
* Requires an **access token** with scopes related to the claims that have to be returned