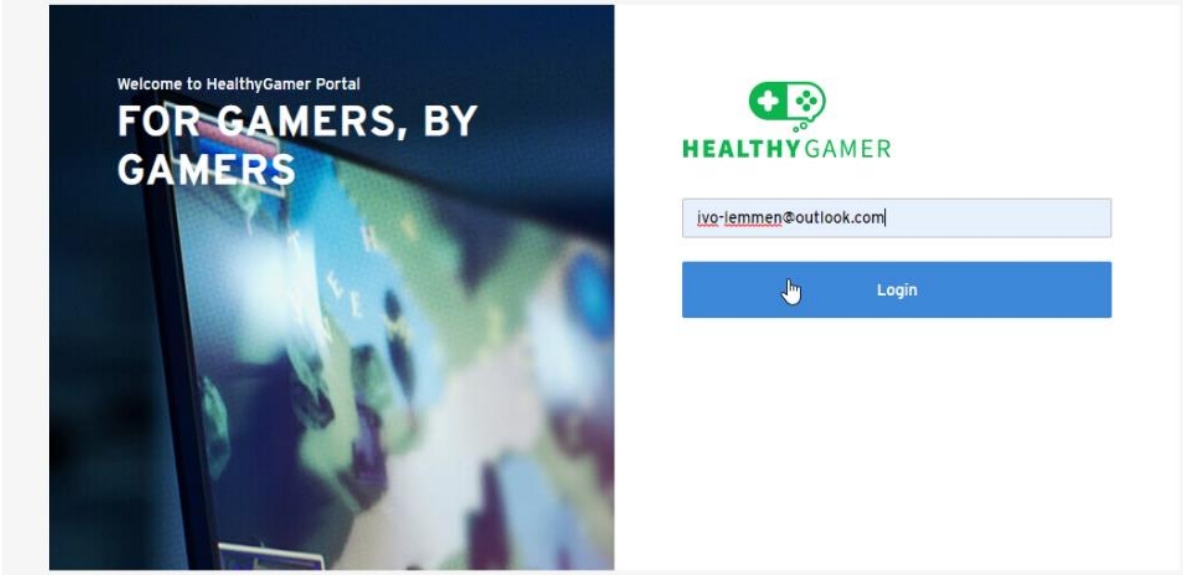
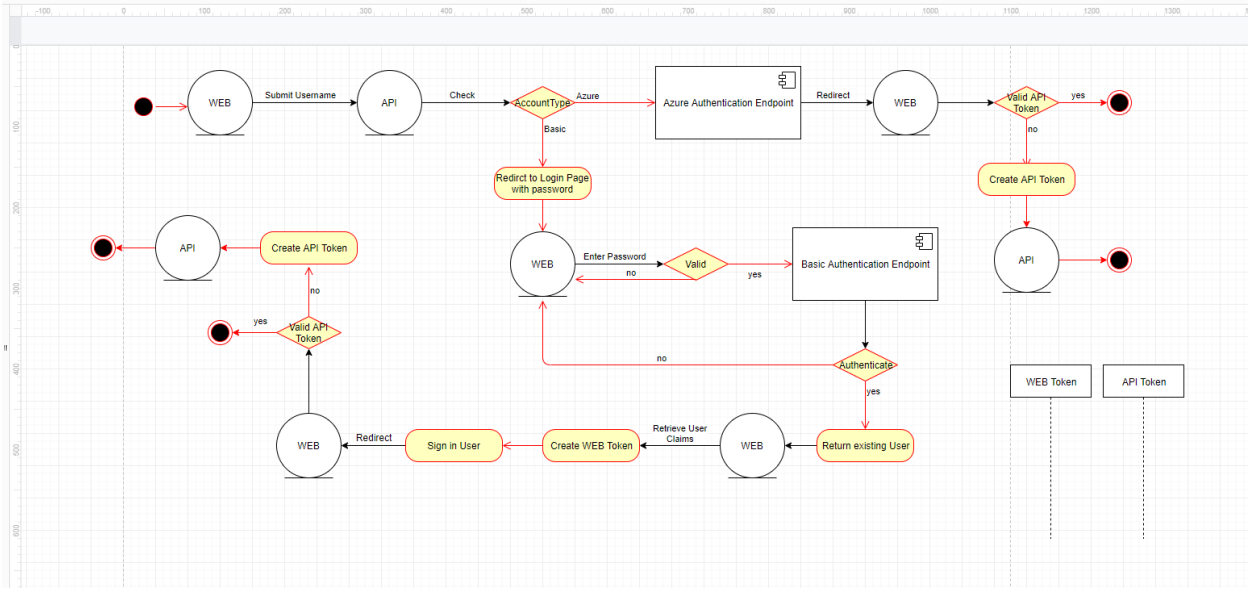
Contents

Authenticatie



1. De Login Pagina van HealthyGamer Portaal



2. De sequentie diagram tussen de werking van de authenticatie binnen WEB en API

Applicatie landingspagina

In onze applicatie wordt de gebruiker begroet door een login pagina, omdat gekozen is om alleen geauthentiseerde gebruikers toegang te geven aan de applicatie. Mocht je nog ingelogd zijn, dan wordt je doorverwezen naar de dashboard pagina.

Beveiliging

Naar mijns inziens helpen alle kleine beetjes in het vormen van een robuust en veilig systeem. In dit onderdeel ga ik uitleg geven over de gebruikte vormen van veiligheid en gegevenswaarborging binnen de applicatie.

- *Gebruiker credentials*
    - Als benodigdheden om succesvol toegang te krijgen tot de applicatie en insgelijks de API moeten gebruikers een email adres en wachtwoord aanvoeren. Deze wordt encrypt opgestuurd naar de API, die deze valideert en met een http succes response terugkoppelt.

- *Encryptie*
    - Er van uitgaande dat er mensen zijn die proberen inzage te krijgen in data die niet bedoelt is voor hun, wordt encryptie gebruikt. Alleen de partijen voor wie de data bedoelt is moeten deze kunnen inzien.
      Door het aanvoeren van een unieke sleutel kan de data worden verstopt in een cryptografisch gegenereerde string. Dit is een van de parameters van de encryptie provider. De opbouw wordt gedaan door de uitvoering van een wiskundige formule die deze parameters gebruikt. Andere benodigde parameters zijn een modulus en exponent.

- *Beveiligde zone*
    - De applicatie kan opgedeeld worden in verschillende lagen. Elke laag voert een bepaalde werkzaamheid uit. De beveiligde zone is een laag waar authenticatie functionaliteit uitgevoerd wordt, liggende onder de acties die zichtbaar zijn voor gebruikers. Deze functies kunnen uitgevoerd worden door middleware of ‘event delegates’ vanuit verschillende ‘handlers’, bijvoorbeeld: wanneer een valide authenticatie token binnen komt, koppel ik een functie aan die actie waardoor ik vervolgens de token kan manipuleren.

- *Authenticatie token*
    - Voor veiligheidsredenen krijg je wanneer je wordt ingelogd een uniek identificatie nummer toegekend, die bepaald in welke mate je toegang hebt. Dit object wordt opgeslagen in de authenticatie ticket en kan niet aangepast worden van buiten af.

- *Autorisatie claims*
    - Claims in de context van autorisatie wordt mee bedoeld de informatie die opgeslagen is in een geauthentiseerde gebruiker object (user principal), dit kan bevatten: rollen en rechten, naam, tijd van inloggen, duur van toegang, of andere informatie die je graag wilt bijhouden en toegankelijk wilt houden. Maar geen gevoelige informatie.

- *User Principal*
    - Wie jij als user bent en waar je toegang tot hebt binnen de applicatie, wordt opgeslagen in een 'User Principal'. Onder principal verstaan we een collectie van de meest belangrijke informatie. Vaak staan hier de autorisatie claims in, maar ook tijd dat je toegang hebt en je gebruikersnaam of email. Dit object wordt opgeslagen in de authenticatie ticket.

- *Authenticatie ticket*
    - Wanneer je bent ingelogd binnen de applicatie wordt je 'in memory' als user bijgehouden in de vorm van een authenticatie ticket. Deze wordt in de beveiligde zone aangemaakt. Dit ticket houd bij: de levensduur, Authenticatie Tokens en de 'User Principal'.

Validatie van ingelogde gebruiker

- WEB tokens
    - Doormiddel van tokens die de informatie van een gebruiker bevatten wordt toegang verleend binnen de aangegeven paden van een website. Ook zorgt dit voor het makkelijk toegankelijk houden van ongevoelige informatie. Dit token wordt maar een keer uitgedeeld en wanneer deze niet meer geldig is of de gebruiker wordt geblokkeerd, dan moet opnieuw ingelogd worden en een nieuw token aangemaakt worden.

- *API tokens*
    - Toegang aan de API wordt verleend aan gebruikers met een valide WEB Token en heeft een korte levensduur. Zonder dat de gebruiker het merkt, wordt deze opnieuw aangemaakt en opgeslagen in de User Principal. Bij elke aanvraag naar de API wordt dit token meegestuurd binnen de autorisatie header van de aanvraag. Hierdoor kan bij aankomst binnen de API gecontroleerd worden of de gebruiker toegang heeft om de actie uit te voeren en of dit een valide token is. Om gebruikers van buiten af, of van aangepaste http data te blokkeren.

JWT

Eigenschappen van JSON Web Token:

```
/// <summary>
/// "iss" (Issuer) Claim
/// </summary>
/// <remarks>The "iss" (issuer) claim identifies the principal that issued the
///   JWT.  The processing of this claim is generally application specific.
///   The "iss" value is a case-sensitive string containing a StringOrURI
///   value.  Use of this claim is OPTIONAL.</remarks>
public string Issuer { get; set; }

/// <summary>
/// "sub" (Subject) Claim
/// </summary>
/// <remarks> The "sub" (subject) claim identifies the principal that is the
///   subject of the JWT.  The claims in a JWT are normally statements
///   about the subject.  The subject value MUST either be scoped to be
///   locally unique in the context of the issuer or be globally unique.
///   The processing of this claim is generally application specific.  The
///   "sub" value is a case-sensitive string containing a StringOrURI
///   value.  Use of this claim is OPTIONAL.</remarks>
public string Subject { get; set; }

/// <summary>
/// "aud" (Audience) Claim
/// </summary>
/// <remarks>The "aud" (audience) claim identifies the recipients that the JWT is
///   intended for.  Each principal intended to process the JWT MUST
///   identify itself with a value in the audience claim.  If the principal
///   processing the claim does not identify itself with a value in the
///   "aud" claim when this claim is present, then the JWT MUST be
///   rejected.  In the general case, the "aud" value is an array of case-
///   sensitive strings, each containing a StringOrURI value.  In the
///   special case when the JWT has one audience, the "aud" value MAY be a
///   single case-sensitive string containing a StringOrURI value.  The
///   interpretation of audience values is generally application specific.
///   Use of this claim is OPTIONAL.</remarks>
public string Audience { get; set; }

/// <summary>
/// "nbf" (Not Before) Claim (default is UTC NOW)
/// </summary>
/// <remarks>The "nbf" (not before) claim identifies the time before which the JWT
///   MUST NOT be accepted for processing.  The processing of the "nbf"
///   claim requires that the current date/time MUST be after or equal to
///   the not-before date/time listed in the "nbf" claim.  Implementers MAY
///   provide for some small leeway, usually no more than a few minutes, to
///   account for clock skew.  Its value MUST be a number containing a
///   NumericDate value.  Use of this claim is OPTIONAL.</remarks>
public DateTime NotBefore => DateTime.UtcNow;
```

```
        /// <summary>
        /// "iat" (Issued At) Claim (default is UTC NOW)
        /// </summary>
        /// <remarks>The "iat" (issued at) claim identifies the time at which the JWT was
        ///   issued.  This claim can be used to determine the age of the JWT.  Its
        ///   value MUST be a number containing a NumericDate value.  Use of this
        ///   claim is OPTIONAL.</remarks>
        public DateTime IssuedAt => DateTime.UtcNow;

        /// <summary>
        /// Set the timespan the token will be valid for (default is 8 hours)
        /// </summary>
        public TimeSpan ValidFor { get; set; } = TimeSpan.FromHours(8);

        /// <summary>
        /// "exp" (Expiration Time) Claim (returns IssuedAt + ValidFor)
        /// </summary>
        /// <remarks>The "exp" (expiration time) claim identifies the expiration time on
        ///   or after which the JWT MUST NOT be accepted for processing.  The
        ///   processing of the "exp" claim requires that the current date/time
        ///   MUST be before the expiration date/time listed in the "exp" claim.
        ///   Implementers MAY provide for some small leeway, usually no more than
        ///   a few minutes, to account for clock skew.  Its value MUST be a number
        ///   containing a NumericDate value.  Use of this claim is OPTIONAL.</remarks>
        public DateTime Expiration => IssuedAt.Add(ValidFor);

        /// <summary>
        /// "jti" (JWT ID) Claim (default ID is a GUID)
        /// </summary>
        /// <remarks>The "jti" (JWT ID) claim provides a unique identifier for the JWT.
        ///   The identifier value MUST be assigned in a manner that ensures that
        ///   there is a negligible probability that the same value will be
        ///   accidentally assigned to a different data object; if the application
        ///   uses multiple issuers, collisions MUST be prevented among values
        ///   produced by different issuers as well.  The "jti" claim can be used
        ///   to prevent the JWT from being replayed.  The "jti" value is a case-
        ///   sensitive string.  Use of this claim is OPTIONAL.</remarks>
        public Func<Task<string>> JtiGenerator => () =>
Task.FromResult(Guid.NewGuid().ToString());
```