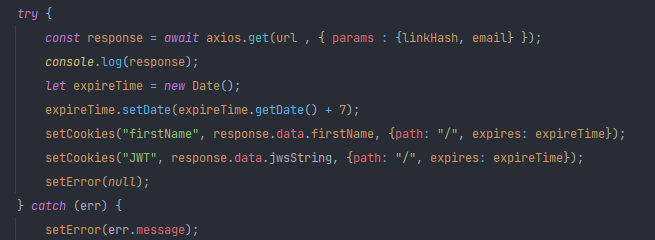
JWT Tokens

## Inleiding

JSON Web Tokens (Vanaf nu in dit document benoemd als: JWT), en JSON Web Signatures (vanaf nu in dit document benoemd als: JWS). is de techniek die we gebruikt hebben om te valideren wie een user is.

Deze JWS ( een Signed JWT ) zijn vanaf het moment dat de user ingelogd is nodig in de back end om te kunnen valideren wie de user is.

# Structuur Front-End

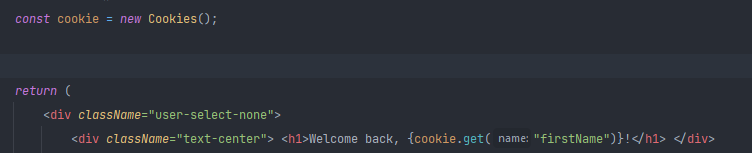


De front end is relatief simpel opgezet gezien het vooral gaat om data opvragen en geven.

In dit bovenstaand voorbeeld probeert een user in te loggen, vervolgens wanneer dit klaar is, geeft het een error message als er iets fout is gegaan. Maar als het goed is gegaan worden er 2 cookies gemaakt: FirstName/ JWT.

Deze blijven in de browser totdat ze expired zijn. Of wanneer het programma de opdracht geeft de cookies te verwijderen.   
(Path: /, betekent dat alle pagina’s op ons domein toegang hebben tot deze cookie)

Als je een cookie wil oproepen, of mee wil geven in JSON kan dit vervolgens heel simpel:



Hierom is het heel belangrijk om de cookies een duidelijk naam te geven!

# Structuur Back-End

De JWT Service Layer is opgedeeld in een aantal methods:

* GenerateJWS – Return Type: UserDTO (Met JWS String)
* VerifyJWS (2x) - Returns: FillJWS
* FillJWS – Return Type: JwsData

GenerateJWS

GenerateJWS heeft als parameter UserDTO. Afhankelijk van de data die de front end meestuurt worden een aantal claims gemaakt. ( Claims: op basis van de token beweert de user dat hij/zij: ID 2 heeft ) Deze wordt in combinatie met een secretkey ( die voor nu elke keer bij het opstarten van de program automatisch generate wordt) gesigned. Hierdoor maken we dus een JWS.   
  
Deze methode wordt opgeroepen op het moment dat de user succesvol ingelogd heeft.

VerifyJWS

VerifyJWS heeft 2 opties als parameter:  
JwsDTO of String

Wanneer de String method wordt opgeroepen wordt er een JwsDTO object gemaakt en daarna alsnog de VerifyJWS met de parameter JwsDTO uitgevoerd.

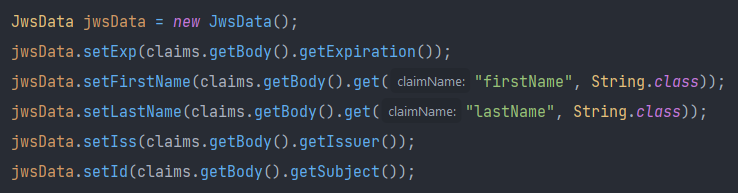
De reden hiervoor is omdat de Front End mogelijk andere DTO’s kan versturen waarvan de jwsString 1 van de fields is.

Wanneer de VerifyJWS method met de JwsDTO wordt gecalled, wordt de JwsString geverified. Dit wordt gedaan door middel van de secretkey. Indien de JWS invalid is, word de Exception: JWTException gecalled. ( Meer informatie hierover bij het gedeelte: Error Throws/Handling )

Als de JWS valid is, wordt de method FillJWS uitgevoerd.

## FillJWS

FillJWS returns een object waarin alle claims staan.



## Error Throws/Handling

Possible Error: JWT Exception.

Indicates: JWS Is invalid.

Possible causes:

* Een user heeft de JWS proberen te manipuleren.
* De JWS is verlopen
* JWS niet ondertekend door ons.

Throwed by: JWTService

Catched by: Not Implemented! ( Zie hiervoor Future Development )

Expected Resolve: HTTP ERROR: FORBIDDEN

# Future development

Op het moment dat een User iets wil uitvoeren ( bijvoorbeeld zijn persoonlijke meetstation wil aanpassen ). Gebruik vanaf nu dan de JWS Functions om er achter te komen wie de User is.

De Rest Controller heeft dus een DTO nodig met de benodigde gegevens + een jwsstring.   
Door middel van de JWSString wordt bepaald wie de user is, worden de claims in een object : JwsData gezet. En kan je dus een database request doen met de gegevens uit de JWS.

Wanneer de user geverified wordt is een try catch recommended. De meest logische error handling zou een HTTP Error: Forbidden zijn.