Programmaontwerp

Er moet een programma ontwikkeld worden waarmee opmerkingen genoteerd kunnen worden over eindexamenleerlingen, door docenten, als input voor het praatje (door de mentor) bij de diplomauitreiking.

Eisen:

* Opmerkingen mogen uitsluitend gemaakt kunnen worden door docenten.
* Van opmerkingen moet altijd met zekerheid bepaald kunnen worden door wie die gemaakt is en wanneer.
* Een opmerking heeft altijd betrekking op een en slechts een leerling.
* Degene die ingelogd is in het programma is automatisch degene die de opmerking toevoegt. Er kunnen dus niet door een docent opmerkingen namens een collega gemaakt worden. De andere collega moet zelf inloggen om te zorgen dat zijn ‘handtekening’ eronder komt te staan.
* Bij inloggen kan er een docentcode uit een vooraf ingevulde lijst geselecteerd worden. Er is geen mogelijkheid zelf een account aan te maken.
* De examenkandidaten staan in een lijst waaruit geselecteerd kan worden. Er kunnen door docenten geen leerlingen toegevoegd worden of leerlinggegevens gewijzigd worden.
* De lijst waaruit gekozen kan worden is gesorteerd op stroom (eerst 5H dan 6V) en daarbinnen oplopend op voornaam.
* De volgende informatie zit al in het systeem over leerlingen: Assink leerlingnummer, klascode, naam, leerjaar.
* Per leerling worden opmerkingen opgeslagen. Per opmerking wordt vastgelegd door wie die is gemaakt (op basis van inlognaam), opmerking en (optioneel) namen van bestanden die zijn aangeleverd bij de betreffende opmerking. Dat kan bijvoorbeeld een fotootje zijn.
* Die bestanden moeten apart worden aangeleverd. Dat kan via de mail aan … of door die te plaatsen in de Google Drive map in de map van de betreffende leerling.
* Er kan ook vanuit meerdere opmerkingen verwezen worden naar eenzelfde bestandje, bijvoorbeeld een foto waar een aantal leerlingen op staat.
* Bestanden die niet op een specifieke leerling betrekking hebben moeten worden geplaatst in de map algemeen H5 of algemeen V6.

Technisch ontwerp

* Er wordt een hoofdpagina onderscheiden met daarbinnen twee schermen (html-forms): een loginscherm (loginForm) en een opmerkingenscherm (opmForm).
* Het switchen tussen de weergave van de beide schermen wordt afgehandeld door een schermcontroller (formController).
* Het inloggen wordt afgehandeld door een inlogcontroller (loginFormController), het noteren van een opmerking bij een leerling wordt afgehandeld door een andere controller (opmFormController).
* Als het wachtwoord overeenkomt met het wachtwoord van de user in de database kan de Firebase autorisatieprocedure gestart worden. Zie de stappen hierna. Zo niet dan wordt gemeld “Dit is niet de juiste code bij deze gebruikersnaam”.
* Bij de autorisatie wordt een token aangemaakt voor degene die inlogt. In die token zit versleuteld opgeslagen de unieke user id (*uid*, de vier-letter Assink docentcode, voorafgegaan door ‘custom:’) en zijn autorisatielevel (*level*), die bepaalt wat hij mag (standaard level: ‘docent’).
* De Firebase method *createToken* van de klasse *FireBaseTokenGenerator* wordt gebruikt voor het maken van de token. Als parameter moet die method een object meekrijgen met daarin de nodes *uid* en *level* met bovengenoemde waarden.
* Gebruik van de Firebase method *authWithCustomToken* zorgt ervoor dat er een object *authData* wordt aangemaakt met als childnodes deze velden *uid* en *level*.
* Er moet een functie *authHandler* geschreven worden voor het afhandelen van het resultaat van de autorisatiepoging, die meegegeven wordt als callback-functie bij de aanroep van *authWithCustomToken*. Die meldt in de console “Inloggen is mislukt” bij error en anders “Inloggen geslaagd met payload: “+authData.
* Ook moet er een callback-functie *authDataCallback* geschreven worden die getriggerd wordt bij iedere wijziging van autorisatiestatus. Dat wordt gedaan door de functie mee te geven als parameter aan de methode onAuth van het object dat de referentie naar de database bevat.
* Dat object krijgt de naam *ref* en bevat een instantie van de klasse die *Firebase* heet en die geïnitieerd wordt (aanroepen van de constructor van de klasse) met als paramater de naam van de database: *<jouw-database>.firebaseIO.com*.
* *authDataCallback* meldt in de console: “User “+authData.uid+” is ingelogd met “+authData als de autorisatiestatus is gewijzigd naar ingelogd en meldt “User is uitgelogd” als de status uitgelogd is geworden.
* Tenslotte moet er met aanroep van de Firebase method *unauth* nog voor gezorgd worden dat als de verbinding verbroken wordt er eerst nog even uitgelogd wordt. Checken of de verbinding verbroken wordt kan met de Firebase method *onDisconnect.*
* Ook moet *unauth* aangeroepen worden wanneer op de knop Uitloggen geklikt wordt, maar die knop verschijnt pas in het volgende scherm.
* Als de autorisatie geslaagd is dan zorgt de formController ervoor dat er van scherm wordt gewisseld. Nu wordt dus niet meer de loginForm maar de opmForm getoond. De controle over het verloop van het proces wordt nu overgedragen van de loginFormController naar de opmFormController.
* Een van de objecten in de opmForm is het gedeelte van het scherm waar wordt getoond wat er al voor opmerkingen zijn gemaakt bij de betreffende leerling.
* Als er geen leerling is geselecteerd moet daar niets staan. Met de jQuery event method *triggerHandler* kan de handler getriggerd worden die anders alleen bij *change* van de inhoud van veld leerling ofwel jQuery object *$(“#Leerling”)* getriggerd wordt.
* Als er wel een leerling is geselecteerd maar er zijn nog geen opmerkingen bij die leerling dan komt op die plek in het scherm te staan “Nog geen opmerkingen voor deze leerling”. Als er wel opmerkingen zijn komt daar te staan “Dit werd er al gezegd: “ met daaronder een lijst (html tag *table*) met de kolommen *opmerking*, *bestanden* en *door* in de *tbody* van de *table.* De kolomnamen komen in de *thead* van de *table* te staan. Tip: opmaak van kolommen moet meegegeven worden aan de th tags in de thead sectie van de table, niet aan de cellen die de waarden bevattten.
* Wanneer er op de knop *opslaan* geklikt wordt moet de invoer gevalideerd worden. Het veld leerling mag niet de default waarde ‘Maak een keuze…’ bevatten en er moet een leerling geselecteerd zijn (niet een van de headers ‘5H’ of ‘6V’ of ‘’). Ook mag het veld opmerking niet leeg zijn en mag het veld niet de default waarde ‘Typ hier jouw opmerking over de leerling die je geselecteerd hebt’ bevatten. Bij een van deze invoervalidatiefouten moet er een melding (alert) in beeld komen met de tekst “alle velden met \* zijn verplicht”. Bij de velden leerling en opmerking moet een sterretje staan.
* Veranderen van de inhoud van het scherm kan door componenten (html elementen, objecten) te tonen of te verbergen of door de inhoud van een component bij te werken. jQuery kent voor dat laatste de methods *.text()*, *.html()* en val*()*, zie [hier](http://www.w3schools.com/jquery/jquery_dom_set.asp). Tonen / verbergen kan met *.show()* en *.hide()*.

De bouwstenen van de webapplicatie:

Het programma omvat een user interface, een database en een programma dat ervoor zorgt dat er op alle mogelijke gebeurtenissen (events) gereageerd wordt.

De app is rechtstreeks bereikbaar op het volgende adres:

<https://torrid-inferno-6428.firebaseapp.com/llopmapp/index.html>

**Gegevensopslag**

De database is de plek waar alle informatie wordt opgeslagen. Dat gebeurt in een beveiligde omgeving, gehost door Firebase, zie [www.firebase.com](http://www.firebase.com). De informatie is als volgt gestructureerd:

Users : {

BAAJ: {

Auth: {

level: “docent”,

uid: “custom:BAAJ”

},

Expires: 1430070852,

Password: 1234,

Provider: “custom”,

Token: “zI1NiJ9.eyJ2Ijo…”,

Uid: “custom:BAAJ” },

BEUN: { etc

}

Leerling: {

116978: {

Klas: “H5H1”,

Klaskort: “5H”,

Leerlingdropdown: “116978 Anne Karssies (H5H1)”,

Naam: “Anne Karssies”,

Opmerkingen: {

-Jm\_NZnFtaWy2tAaAzlo: {

Bestanden: “”,

Door: “BAAJ”,

Opmerking: “een opmerking” },

etc

}

},

127225: { etc

}

**De buitenkant (user interface)**

De user interface omvat twee schermen, te weten een loginscherm en een invoerscherm voor de opmerkingen.

In de html-broncode wordt vastgelegd uit welke elementen de pagina’s bestaan. De html-code voor beide schermen staat samen in één bestand. In de header <head></head> staan de verwijzingen naar drie JavaScript libraries die programmaonderdelen bevatten waar in de applicatie gebruik van wordt gemaakt. Te weten:

* Een library om gebruik te kunnen maken van het Firebase-framework, wat met name gericht is op het ophalen en bewaren van gegevens in de door Firebase gehoste objectgeoriënteerde database.
* Een library om gebruik te kunnen maken van jQuery-opdrachten. Die worden gebruikt om goed met de elementen om te kunnen gaan waar de html-pagina uit bestaat. Alle onderdelen van de pagina worden daardoor beschouwd als objecten.
* Een specifieke library voor het genereren van sleutels voor beveiliging van de toegang tot de database.

Verder staan er in de header nog de verwijzingen naar de css-styleheet (zie hierna) en het programma dat ervoor zorgt dat er op alle gebeurtenissen gereageerd wordt (zie verderop).

De hoofdonderdelen van de html-pagina zijn (tussen haakjes staan de id’s van de elementen genoemd):

* Een element (*mainform*) waar alles in zit, te weten het Assink-logo, twee formulieren (het inlog- en opmerkingenscherm) en een gedeelte voor het tonen van opmerkingen.
* Een loginscherm (*loginform*) met daarin een containerelement (*logincontainer*) waarbinnen een fieldset is geplaatst met daarbinnen twee invoerveldjes (een selectieveld (*Door*) en een vrij invoerveld (*Code*)) en een knop (*login*).
* Een opmerkingschem (*form*) met daarin drie invoerveldjes (*Leerling, Opmerking* en *Bestanden*) en een container (*buttoncontainer*) met twee knoppen (*submit* en *logout*)
* Twee containers voor het tonen van de lijst met opmerkingen van de leerling. Eén container is de variant voor leerlingen waar al opmerkingen voor gemaakt zijn (*opmerkingcontainer*) en de andere is de variant om te tonen bij de leerlingen bij wie nog geen opmerking is gemaakt (*geenopmerkingcontainer*).

De opmaak van de schermen wordt vastgelegd in een css-bestand met de naam opmerkingform.css. Daarin wordt per (groep van) elementen bepaald hoe die eruit moet komen te zien. Opmaak die voor een specifiek element wordt bepaald heeft prioriteit boven een definitie op een algemener niveau. Er wordt bijvoorbeeld voor alle input elementen van het type tekst bepaald dat ze een breedte van 100% (ten opzichte van het element waarin ze geplaatst zijn) hebben, maar voor het veld met id *Door* (wat ook een input veld van het type text is) wordt in afwijking daarvan bepaald dat die een vaste breedte van 199 pixels heeft.

Het css-bestand bevat layoutkenmerken voor elementen in het algemeen (bijvoorbeeld *table*), voor elementen van een bepaalde klasse (bijvoorbeeld *.loginlabel*) en voor specifiek met id genoemde elementen (bijvoorbeeld *#Door*).

De schermen zien er zo uit:





**Het programma**

Tenslotte wordt in een programma geschreven in JavaScript met gebruikmaking van jQuery- en Firebaseopdrachten, bepaald hoe het allemaal moet werken.

Wat kan er allemaal gebeuren (events)?

|  |  |
| --- | --- |
| **Event** | **Event-listener** |
| htmlpagina is geladen in browser | $(document).ready() |
| er is een poging gedaan om in te loggen op de Firebase-database | authWithCustomToken() |
| - mislukt |  |
| - geslaagd |  |
| autorisatiestatus is gewijzigd | onAuth() |
| - er is ingelogd |  |
| - er is uitgelogd |  |
| er moet van scherm gewisseld worden | formController() |
| - naar loginForm |  |
| - naar opmForm |  |
| er is op login geklikt | $(“#login”).click() |
| - invoervalidatie ok |  |
| - invoervalidatie niet ok |  |
| de waarde van veld leerling is gewijzigd | $(“#Leerling”).change() |
| - geen leerling |  |
| - leerling zonder opmerkingen |  |
| - leerling met opmerkingen |  |
| er is op opslaan geklikt | $(“#submit”).click() |
| - invoervalidatie ok |  |
| - invoervalidatie niet ok |  |
| er is op logout geklikt | $(“#logout”).click() |
| - invoervalidatie ok |  |
| - invoervalidatie niet ok |  |
| de verbinding met de Firebase-database is verbroken | onDisconnect() |

Hoe moeten die events afgehandeld worden (event-handlers)?

|  |  |
| --- | --- |
| **Event** | **Event-handler** |
| htmlpagina is geladen in browser | runScript() |
| er is een poging gedaan om in te loggen op de Firebase-database | authHandler(error, authData) |
| - mislukt |  |
| - geslaagd |  |
| autorisatiestatus is gewijzigd | authDataCallback(authData) |
| - er is ingelogd |  |
| - er is uitgelogd |  |
| er moet van scherm gewisseld worden | switchForm(log) |
| - naar loginForm |  |
| - naar opmForm |  |
| er is op login geklikt | loginClickHandler() |
| - invoervalidatie ok |  |
| - invoervalidatie niet ok |  |
| de waarde van veld leerling is gewijzigd | changeHandler() |
| - geen leerling |  |
| - leerling zonder opmerkingen |  |
| - leerling met opmerkingen |  |
| er is op opslaan geklikt | submitClickHandler() |
| - invoervalidatie ok |  |
| - invoervalidatie niet ok |  |
| er is op logout geklikt | logoutClickHandler() |
| - invoervalidatie ok |  |
| - invoervalidatie niet ok |  |
| de verbinding met de Firebase-database is verbroken | disconnectHandler() |

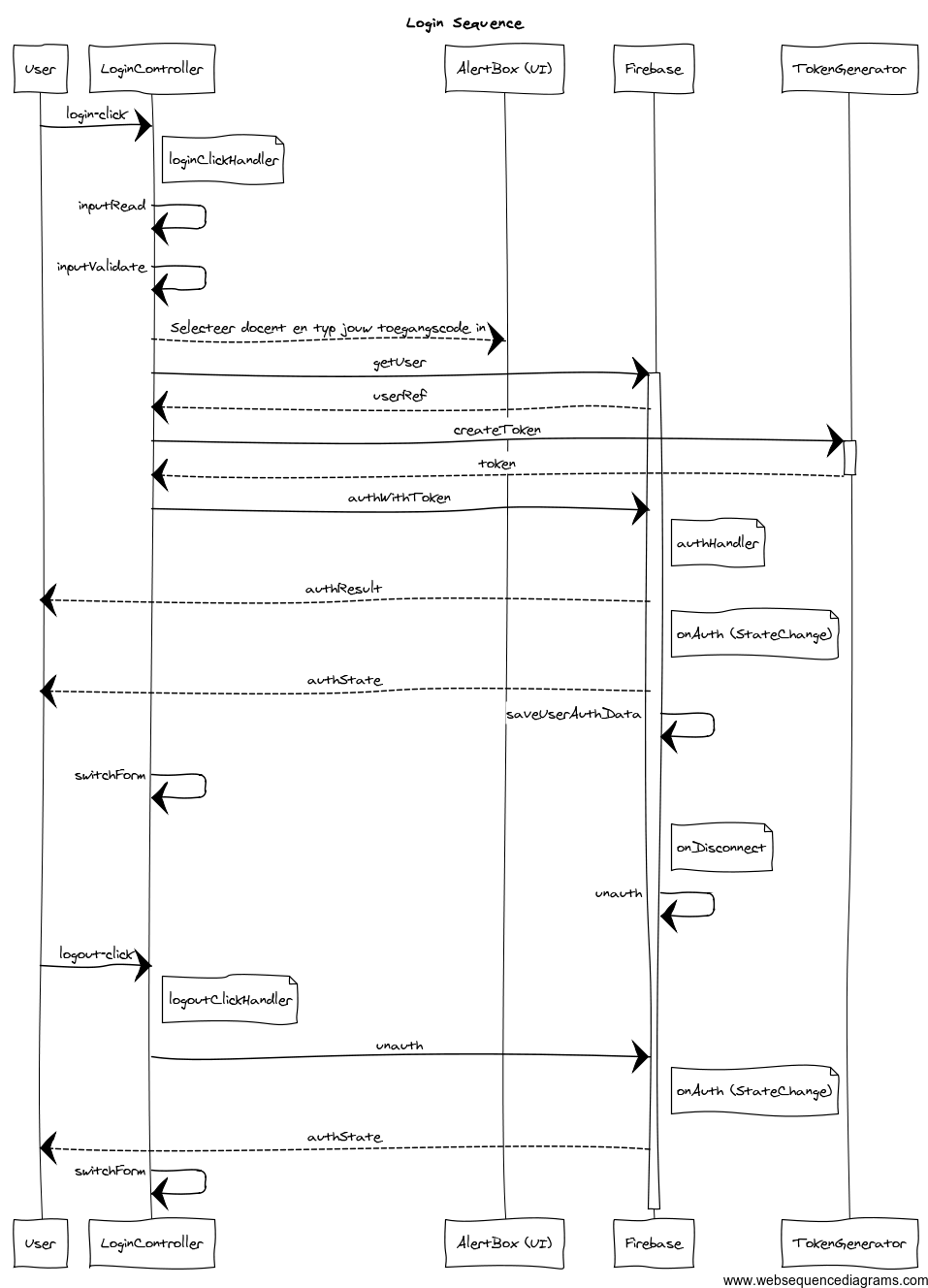
Wat moeten die event-handlers doen (taken)?

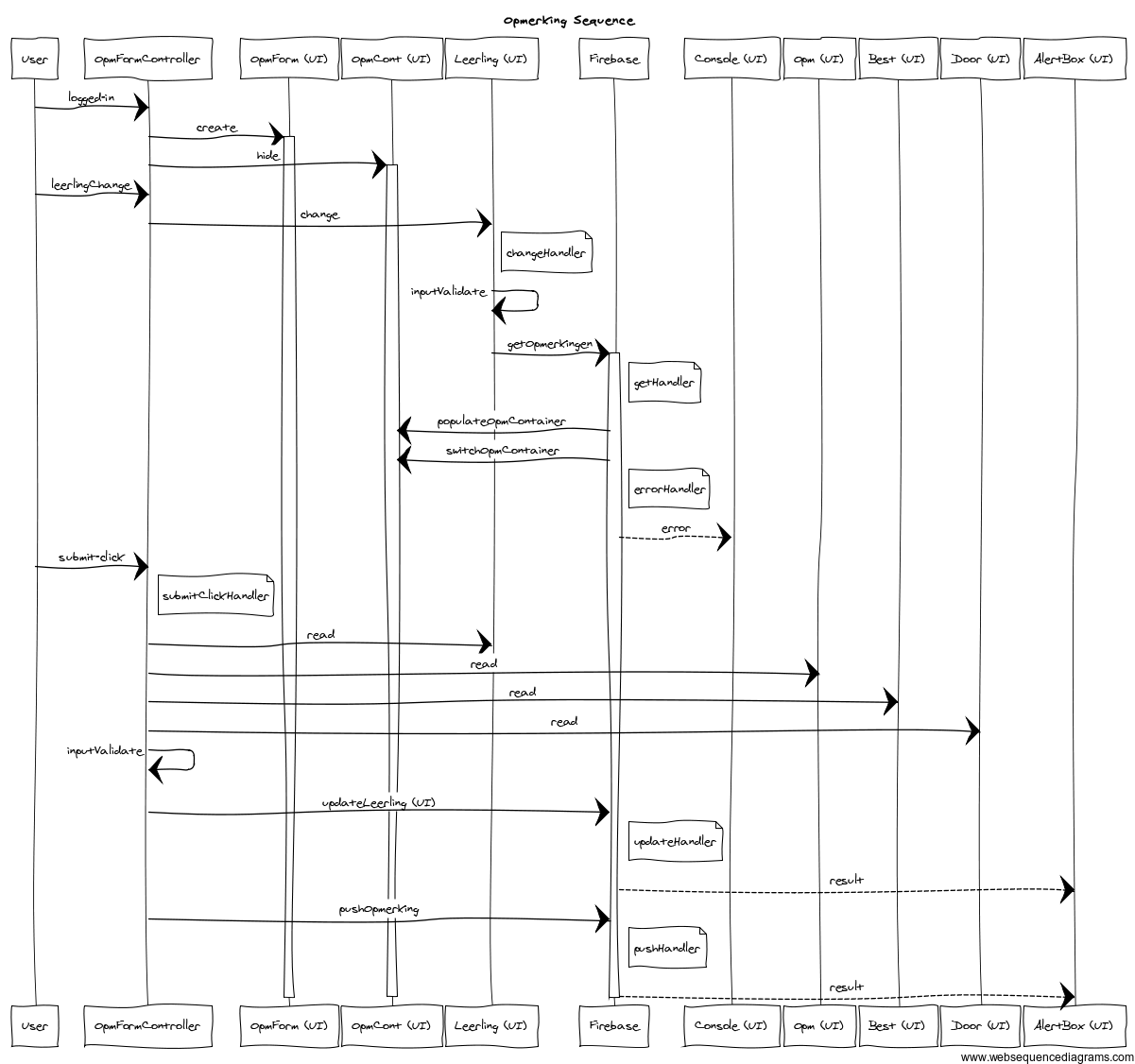
|  |  |
| --- | --- |
| **Event** | **Taken** |
| htmlpagina is geladen in browser (nadat pagina voor het eerst is opgeroepen in de browser of na vernieuwen van de webpagina) | - zorgen dat login-scherm getoond wordt  - object (*ref*) met Firebasedatabase-referentie maken  - handlers inlezen (‘noteren’ voor welke events er handlers zijn)  - callbacks registreren (handlers koppelen aan events) |
| er is een poging gedaan om in te loggen op de Firebase-database | Objecten *error* en *authData* ophalen |
| - mislukt | “Login failed” met error → console |
| - geslaagd | “Authenticated (…)” met authData melden in console |
| autorisatiestatus is gewijzigd | *authData* ophalen |
| - er is ingelogd | - "User " + authData.uid + " is logged in with " + authData.provider → console  - authorisatiegegevens van user bijwerken in database node users/<uid>  - switchen naar loginForm |
| - er is uitgelogd | - "User is logged out" → console  - switchen naar opmForm |
| er moet van scherm gewisseld worden | Velden in beide forms resetten en beide opmcontainers verbergen, vervolgens: |
| - er is uitgelogd | - loginForm tonen  - opmForm verbergen |
| - er is ingelogd | - opmForm tonen  - loginForm verbergen |
| - verbinding is verbroken | - beide verbergen |
| er is op login geklikt | - input lezen (‘inputRead’)  - checken of velden user en password zijn ingevuld: inputValidateLogin  - zo ja: gegevens vergelijken met user info in de database (getUser→getHandler) |
| - invoervalidatie ok | - token maken voor user (createToken)  - user autoriseren met token (authWithCustomToken) |
| - invoervalidatie niet ok | - "Gebruikersnaam en code komen niet overeen" melden in AlertBox |
| - fout bij ophalen usergegevens | "Ophalen van gebruikersgegevens is mislukt: " + errorObject.code → AlertBox |
| de waarde van veld leerling is gewijzigd | - overige velden resetten  - kleurtje van veld leerling → lightblue  - input lezen  - checken of er een leerling in veld leerling is geselecteerd  \* zo nee: zie ‘geen leerling’  \* zo ja: getOpmerkingen(leerling)  - opmtablebody resetten / (her)vullen (populateOpmContainer)  - bepalen welke container getoond moet worden (switchOpmContainer) |
| - geen leerling | - opmtablebody resetten  - beide opmContainers verbergen |
| - leerling zonder opmerkingen | - opmContainer verbergen  - geenOpmContainer tonen |
| - leerling met opmerkingen | - geenOpmContainer verbergen  - opmContainer tonen |
| er is op opslaan geklikt | - input lezen  - checken of velden leerling en opmerking zijn ingevuld: inputValidateOpm |
| - invoervalidatie ok | - leerling zo nodig toevoegen aan database (updateLeerling→ updateHandler)  - melden als dat mislukt → AlertBox  - opmerking (velden opmerking, bestanden en user) toevoegen bij leerling (pushOpmerking→ pushHandler) |
| - invoervalidatie niet ok | "Alle velden met \* zijn verplicht" → AlertBox |
| er is op logout geklikt | - autorisatie status wijzigen (*unauth*)  - offAuth? |
| de verbinding met de Firebase-database is verbroken | - autorisatie status wijzigen (*unauth*)  - beide forms verbergen (switchForm) |

Programmeurs (ontwerpers van programmacode) nemen meestal niet de tijd om dit zo uitgebreid op te schrijven. Ddenk aan de tijd die het kost om deze documentatie te genereren en bij te werken als iemand bedacht heeft bijvoorbeeld een naam van het een of ander even te wijzigen of een stukje programmalogica toch net even anders te organiseren… Maar wat ze vaak wel doen is een schemaatje maken waar een en ander kort wordt samengevat. Met gebruik van een of ander tooltje kan zo een schemaatje gegenereerd worden en op die manier is het up-to-date houden van die documentatie redelijk te doen.

Voor het tekenen van wat een sequence diagram genoemd wordt heb ik hier gebruik gemaakt van een gratis on-line tooltje op <https://www.websequencediagrams.com/> .

De code die dat schema genereert is bijgevoegd. Als je de code van een van de schemaatjes op de site in het linkerdeel van het scherm plakt krijg je het schemaatje te zien zoals afgebeeld op de volgende pagina.





**Code voor UML Sequencediagrammen**

title Login Sequence

User->LoginController: login-click

note right of LoginController: loginClickHandler

LoginController->LoginController: inputRead

LoginController->LoginController: inputValidate

LoginController-->AlertBox (UI): Selecteer docent en typ jouw toegangscode in

LoginController->+Firebase: getUser

Firebase-->LoginController: userRef

LoginController->+TokenGenerator: createToken

TokenGenerator-->-LoginController: token

LoginController->Firebase: authWithToken

note right of Firebase: authHandler

Firebase-->User: authResult

note right of Firebase: onAuth (StateChange)

Firebase-->User: authState

Firebase->Firebase: saveUserAuthData

LoginController->LoginController: switchForm

note right of Firebase: onDisconnect

Firebase->Firebase: unauth

User->LoginController: logout-click

note right of LoginController: logoutClickHandler

LoginController->Firebase: unauth

note right of Firebase: onAuth (StateChange)

Firebase-->User: authState

LoginController->LoginController: switchForm

title Opmerking Sequence

User->OpmFormController: logged-in

OpmFormController->+OpmForm (UI): create

OpmFormController->+OpmCont (UI): hide

User->OpmFormController: leerlingChange

OpmFormController->Leerling (UI): change

note right of Leerling (UI): changeHandler

Leerling (UI)->Leerling (UI): inputValidate

Leerling (UI)->+Firebase: getOpmerkingen

note right of Firebase: getHandler

Firebase->OpmCont (UI): populateOpmContainer

Firebase->OpmCont (UI): switchOpmContainer

note right of Firebase: errorHandler

Firebase-->Console (UI): error

User->OpmFormController: submit-click

note right of OpmFormController: submitClickHandler

OpmFormController->Leerling (UI): read

OpmFormController->Opm (UI): read

OpmFormController->Best (UI): read

OpmFormController->Door (UI): read

OpmFormController->OpmFormController: inputValidate

OpmFormController-->AlertBox (UI): Alle velden met een \* zijn verplicht

OpmFormController->Firebase: updateLeerling (UI)

note right of Firebase: updateHandler

Firebase-->AlertBox (UI): result

OpmFormController->Firebase: pushOpmerking

note right of Firebase: pushHandler

Firebase-->AlertBox (UI): result

Firebase-->AlertBox (UI): Bedankt! Jouw opmerking is opgeslagen.

**HTML Broncode (login- en opmerkingscherm)**

<!DOCTYPE html>

<html>

<head>

<meta http-equiv='Content-Type' content='text/html; charset=utf-8'/>

<title>Assink Leerlingopmerkingenapp</title>

<script src="https://cdn.firebase.com/js/client/2.2.1/firebase.js"></script>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.js"></script>

<script src="https://rawgit.com/firebase/firebase-token-generator-node/master/dist/firebase-token-generator.js"></script>

<link rel="stylesheet" href="css/opmerkingform.css" />

<script src="js/opmerkingform.js"></script>

<!-- <script src="js/login.js"></script> -->

</head>

<body>

<div id="mainform">

<!-- <h2>Assink Leerlingopmerkingenapp</h2> -->

<!-- Required div starts here -->

<img id='logo' src='images/logo-het-assink.png'>

<form id="loginform">

<div id="logincontainer">

<fieldset>

<legend>Login:</legend>

<p class="loginlabel">Docent</p>

<select name='Door' id='Door' class="loginfield" size='1' >

<option value='Maak een keuze...'>Maak een keuze...</option>

<option value='BAAJ'>BAAJ Jolanda Baart</option>

<option value='BEUN'>BEUN Jos Beun</option>

etc

</select><br/>

<p class="loginlabel">Code</p>

<input type="text" id="Code" class="loginfield"/>

</fieldset>

<br/>

<div class="buttoncontainer">

<input type="button" id="login" value="Login"/>

<!-- <input type="button" id="logout" value="Logout"/> -->

</div>

</div>

</form>

<form id="form" accept-charset='UTF-8'>

<h3>Leerlingopmerkingenapp</h3>

<label>\* Leerling:</label>

<br/>

<select name='Leerling' id='Leerling' size='1'>

<option value='Maak een keuze...'>Maak een keuze...</option>

<option value='5H:'>5H:</option>

<option value='127623 Aaron Freriksen (H5H3)'>127623 Aaron Freriksen (H5H3)</option>

<option value='116978 Anne Karssies (H5H1)'>116978 Anne Karssies (H5H1)</option>

<option value=''></option>

<option value='6V:'>6V:</option>

<option value='117867 Anna olde Bolhaar (H6V2)'>117867 Anna olde Bolhaar (H6V2)</option>

etc

</select><br/>

<br/>

<label>\* Opmerking:</label>

<br/>

<textarea placeholder="Jouw opmerking over deze leerling..." id="Opmerking"></textarea><br/>

<br/>

<label>Bestanden:</label>

<br/>

<textarea placeholder="Map en naam van bestand(en) bij deze opmerking in Google Drive Assink leerlingopmerkingen-map..." id="Bestanden"></textarea><br/>

<br/>

<div class="buttoncontainer">

<input type="button" id="submit" value="Opslaan"/>

<input type="button" id="logout" value="Logout"/>

</div>

<br/><br/>

<div id="opmerkingcontainer">

<label style="display:block; float: left;">Dit werd er al gezegd:</label><br/><br/>

<div id="tablearea" >

<table id="opmtable" cellspacing="0">

<thead>

<th id="thopmerking">opmerking</th>

<th id="thbestanden">bestanden</th>

<th id="thdoor">door</th>

</thead>

<tbody id="opmtablebody">

</tbody>

</table><br/>

</div>

</div>

<div id="geenopmerkingcontainer">

<label style="float: left;">Nog geen opmerkingen voor deze leerling</label><br/><br/><br/>

</div>

</form>

</div>

</body>

</html>

**CSS opmaakcode**

@import url(https://fonts.googleapis.com/css?family=Fauna+One|Muli);

#Door, #Code, #login, #logout, .loginlabel{

display: inline-block;

}

#login {

margin-left: 75px;

}

#loginform {

margin-top: 50px;

margin-left: 50px;

padding: 30px;

width: 350px;

border: 6px solid #39A7DA;

border-radius: 50px;

}

#logincontainer {

margin: 50px 20px 20px 20px;

}

.buttoncontainer {

margin: 20px auto 0 auto;

width: 245px;

}

#Code {

width: 197px;

padding: 0px;

border: 1px solid rgb(169, 169, 169);

}

#Door {

width: 199px;

}

.loginfield {

float: right;

}

table {

width: 350px;

text-align: left;

background-color: lightblue;

color: red;

font-Weight:400;

border-radius: 5px;

white-space: pre-wrap;

word-wrap: break-word;

table-layout: fixed;

}

td {

vertical-align: top;

color: black;

border-top: 1px solid #8AB8E6;

}

#thopmerking {

width: 200px;

}

#thbestanden {

width: 80px;

}

#thdoor {

width: 50px;

}

#logo{

position:absolute;

left:16px;

top:24px;

width:99px;

height:101px;

z-index:9;

}

#form{

background-color:#77B655;

color:#D2011D;

border-radius:50px;

border:6px solid #39A7DA;

padding: 4px 30px;

font-Weight:700;

width:350px;

font-size:12px;

float:left;

height:auto;

margin-left: 50px;

margin-top: 50px;

}

label{

font-size:15px;

}

h3{

text-align:center;

font-size:21px;

}

div#mainform{

width:350px;

margin:0px;

font-family: 'Fauna One', serif;

}

input[type=text]{

width:100%;

height:40px;

margin-top:10px;

border:none;

border-radius:3px;

padding:5px;

}

textarea{

width:100%;

height:60px;

margin-top:10px;

border:none;

border-radius:5px;

padding:5px;

resize:none;

}

input[type=radio]{

margin: 10px 5px 5px 5px;

}

input[type=button]{

width:100px;

height:40px;

margin:10px;

background-color:#EE8800;

border-radius:3px;

font-family: 'Fauna One', serif;

font-Weight:bold;

font-size:18px;

color:#003C8E;

}

select{

height:40px;

margin-top:10px;

border-radius:3px;

}

**Het programma (JavaScript / jQuery en met gebruikmaking van Firebase framework)**

$(document).ready(function(){

switchForm("logout");

// Monitoring User Authentication State

// Use the onAuth() method to listen for changes in user authentication state

// Create a callback which logs the current auth state

function authDataCallback(authData) {

if (authData) {

console.log("User " + authData.uid + " is logged in with " + authData.provider);

ref.child('loapp\_users').child(authData.uid.replace('custom:', '')).update(authData);

switchForm("login");

} else {

console.log("User is logged out");

switchForm("logout");

}

}

// Register the callback to be fired every time auth state changes

ref = new Firebase("https://torrid-inferno-6428.firebaseio.com");

ref.onAuth(authDataCallback);

ref.onDisconnect(function() {

ref.unauth(); // voorkomen dat ie ingelogd blijft

switchForm();

});

// LOGIN

// The code to authenticate a user varies by provider and transport method, but they all have similar signatures and

// accept a callback function. Use it to handle errors and process the results of a successful login.

// Create a callback to handle the result of the authentication

function authHandler(error, authData) {

if (error) {

console.log("Login Failed!", error);

} else {

console.log("Authenticated successfully with payload:", authData);

}

};

function switchForm(log) {

$("#opmerkingcontainer").hide();

$("#geenopmerkingcontainer").hide();

$('#loginform')[0].reset();

$('#form')[0].reset();

if (log == "login") {

$("#loginform").hide();

$("#form").show();

} else if (log == "logout") {

$("#form").hide();

$("#loginform").show();

} else {

$("#loginform").hide();

$("#form").hide();

}

}

$("#login").click(

function loginClickHandler() {

// inputRead:

\_user = $("#Door").val();

var \_level = "docent";

var \_password = $("#Code").val();

if (inputValidateLogin(\_user, \_password)) {

getUser(\_user, \_password);

}

event.preventDefault();

}

);

function inputValidateLogin(user, password) {

if(

user == "Maak een keuze..." || user == "" ||

password == ""

) {

alert("Selecteer docent en typ jouw toegangscode in");

return false;

} else {

return true;

}

};

function getUser(user, password) {

var usersRef = new Firebase("https://torrid-inferno-6428.firebaseio.com/loapp\_users");

// Authenticate users with a custom Firebase token

var userRef = usersRef.child(user);

// Attach an asynchronous callback to read the data at our user reference

userRef.once("value",

function getHandler(snapshot) {

if (snapshot.val().password == password) {

var token = createToken(user);

ref.authWithCustomToken(token, authHandler);

} else {

alert("Gebruikersnaam en code komen niet overeen");

ref.unauth();

}

},

function errorHandler(errorObject) {

alert("Ophalen van gebruikersgegevens is mislukt: " + errorObject.code);

}

);

}

function createToken(user) {

var tokenGenerator = new FirebaseTokenGenerator("HhQ4rZxUmEmV40zdpelTj73qFyxGEEP41Eh8ld7K");

var updatedObj = {

"uid": "custom:"+user,

"level": "docent"

}

return tokenGenerator.createToken(updatedObj);

};

$("#logout").click(

function() {

ref.unauth(); // dit triggert authDataCallback met authData = null

// ref.offAuth(authDataCallback); // dat is dubbelop, is al geregistreerd bij ref.onAuth()!

event.preventDefault();

}

);

$("#Leerling").change(function changeHandler(){

$("#Opmerking").val("");

$("#Bestanden").val("");

$(this).css("background-color", "lightblue");

var leerlingNummer = this.value.substring(0,6).trim();

if (this.value !== "Maak een keuze..." &&

this.value !== "5H:" &&

this.value !== "6V:") {

getOpmerkingen(leerlingNummer);

} else {

$("#opmerkingcontainer").hide();

$("#geenopmerkingcontainer").hide();

$("#opmtablebody").html("");

}

});

$("#submit").click(function submitClickHandler(){

var \_leerling = $("#Leerling").val();

var \_opmerking = $("#Opmerking").val();

var \_bestanden = $("#Bestanden").val();

var \_user = ref.getAuth().uid.replace("custom:","");

success = true;

if (inputValidateOpm(\_leerling, \_opmerking)) {

event.preventDefault();

var leerlingNummer = updateLeerling(\_leerling);

pushOpmerking(leerlingNummer, \_opmerking, \_bestanden, \_user);

}

});

function inputValidateOpm(leerling, opmerking) {

if (

leerling == "Maak een keuze..." || leerling == "" || leerling == "5H:" || leerling == "6V:" ||

opmerking == "Jouw opmerking over deze leerling..." || opmerking == ""

) { //errorHandler

alert("Alle velden met \* zijn verplicht");

return false;

} else {

return true;

}

}

function updateLeerling(leerling) {

ref = new Firebase("https://torrid-inferno-6428.firebaseio.com/");

var leerlingNummer = leerling.substring(0,6).trim();

var updatedObj = {

"naam": leerling.substring(7,33).trim(),

"klas": leerling.substring(34,38),

"klaskort": leerling.substring(35,37),

"leerlingdropdown": leerling};

ref.child("leerling").child(leerlingNummer).update(updatedObj, function updateHandler(data) {

if (data) { alert(data); success = false; }

});

return leerlingNummer;

};

function pushOpmerking(leerlingNummer, opmerking, bestanden, user) {

ref.child("leerling").child(leerlingNummer).child("opmerkingen").push({

"opmerking": opmerking,

"bestanden": bestanden,

"door": user},

function pushHandler(data) {

if (data) { alert(data); success = false; }

if (success) {

alert("Bedankt! Jouw opmerking is opgeslagen.");

$("#Leerling").triggerHandler("change");

/\*

$('#form')[0].reset(); //To reset form fields

$("#opmerkingcontainer").hide();

$("#geenopmerkingcontainer").hide();

\*/

}

});

};

function getOpmerkingen(leerling) {

var leerlingenRef = new Firebase("https://torrid-inferno-6428.firebaseio.com/leerling");

var leerlingRef = leerlingenRef.child(leerling).child("opmerkingen");

leerlingRef.once("value",

function getHandler(snapshot) {

populateOpmContainer(snapshot);

switchOpmContainer(snapshot);

},

function errorHandler(errorObject) {

alert("Ophalen van opmerkingen is mislukt: " + errorObject.code);

}

);

};

function populateOpmContainer(snapshot) {

$("#opmtablebody").html("");

snapshot.forEach(function(data) {

var cel1 = data.val().opmerking;

var cel2 = data.val().bestanden;

var cel3 = data.val().door;

var tr = "<tr><td>"+cel1+"</td><td>"+cel2+"</td><td>"+cel3+"</td></tr>";

$("#opmtablebody").prepend(tr); // nieuwste bovenaan zetten met prepend ipv append

});

};

function switchOpmContainer(snapshot) {

if (!snapshot.exists()) {

$("#geenopmerkingcontainer").show();

$("#opmerkingcontainer").hide();

} else {

$("#geenopmerkingcontainer").hide();

$("#opmerkingcontainer").show();

}

};

});