Onderzoeks verslag DOOMotica

22 januari 2018

Inhoudsopgave

1 Domotica

De reden dat we dit project doen is zodat we de domotica in een huis zouden kunnen besturen. Hier voor maken we een webpagina waarop op onder andere de functies van Dahaus, spelletjes, tegeltjes voor je favoriete sites en nog wat andere tegeltjes voor je site geschiedenis.

2 Website bouwen

De opdracht voor de studenten is het maken van een website waar de gebruiker een account kan aanmaken en op dit account kan inloggen. Na het inloggen moet de gebruiker zijn favorieten site op de pagina kunnen zetten in de daar aangewezen tegels er voor. verder zijn er nog tegels voor de laatst bezochte sites. verder is er nog plek voor simpele spelletjes en als laatste moet de gebruiker Dahaus kunnen gebruiken.

De web applicatie heeft een aantal eisen:

- Er is een ERD opgesteld voor het Database Management Systeem.
- Voor de volgende functies worden PSD's opgesteld:
  - Login-functie
  - Create User-functie
  - Aanmaken tegels-functie
- Door gebruik te maken van XHTML, CSS en ASP.NET wordt de webapplicatie gerealiseerd

3 Hindernissen

3.1 Afwezigheid docent 'Databases'

In de afgelopen periode is de leraar voor het vak 'Databases' vier weken afwezig geweest. Dit heeft ervoor gezorgd dat de projectgroep tot week 7 geen les heeft gehad in de onderwerpen: Referentiële integriteit en de vragentaal 'SQL.

De studenten hebben dit geprobeerd op te vangen door zelf onderwerpen te verdelen van het lesplan en zelf uit te zoeken. Voor het assessment zijn alle lessen ingehaald en toegepast op de database.

3.2 hashen

hashen houd in dat je een wachtwoord zo vervormt met een speciaal algoritme dit kan alleen 1 kant op waardoor het originele wachtwoord niet terug te vinden is. Tenzij je een super computer heb, maar dan ben je nog steeds wel even bezig. Het probleem waar de studenten voor stonden was dat ze hier nog niet eerder mee hadden gewerkt. en dus zelf moesten kijken hoe het hashen werkt. nadenken te weten hoe het werkt deed hij het nog niet goed,

maar na stapsgewijs erdoor heen gelopen te zijn kwamen ze uiteindelijk bij het probleem. Dat na het ophalen van de database de Hash of wachtwoord niet eerst omgezet werd naar bytes. Waardoor de studenten de eerste 16 tekens als salt zagen en wat er na kwam als hash, maar dit hadden de eerste 16 bytes moeten zijn waardoor bij het vergelijken van beide gehashde wachtwoorden niet gelijk waren. Dit hebben De studenten simpel opgelost door het opgehaalde uit de database meteen om te zetten naar bytes.

3.3 Cookies

In de web-applicatie hebben de studenten cookies gebruikt om ervoor te zorgen dat gebruikers ingelogd blijven. Op basis hiervan is ook een Uitlog-knop gebouwd die de cookie moet terugzetten naar de huidige datum -1 dag. Na het drukken op de knop bleef de cookie bestaan en verwijderde de browser deze dus niet. Zie Figuur 1

In de 9de week(!) kwamen de studenten erachter dat het aan de testbrowser lag. Deze verwijderde de cookies niet wanneer ze waren verlopen, zelfs niet wanneer de gehele browser werd afgesloten. Een oplossing is hiervoor nog niet gevonden.

4 Centrale vraag

In hoe verre zijn De studenten instaat tot het bouwen van een webpagina. Dit houd in een site met inlog systeem die ook gekoppeld is aan een database en werkende functie heeft zoals Dahaus en spelletjes om te spelen voor de gebruiker. Verder wil de opdrachtgever hoe uitgebreid de studenten de webpagina kunnen maken en wat de studenten nog meer verzinnen om als extra dingen er op te zetten. Denk hierbij bijvoorbeeld aan extra functie als een muziek speler.

4.0.1 Connectie

De connectie met DaHaus is gemaakt met behulp van een TCP client. DaHaus is een TCP listener en om met een listener te verbinden moet je een TCP client gebruiken. Er is voor gekozen om de connectie met DaHaus alleen te maken als er een message verstuurd moet worden.

Nadat het bericht is verstuurd wordt de connectie ook meteen weer afgesloten. Het lukte namelijk niet om een connectie te openen en open te houden en op die connectie dan meerdere berichten te sturen en de connectie door middel van een event af te sluiten. Het open houden van de connectie gaf ook het probleem dat als de connectie niet goed werd afgesloten, dan crashde DaHaus. Terwijl nu de connectie automatisch wordt afgesloten waardoor er dus haast geen kans is op een crash van DaHaus.

Om het bericht naar DaHaus te sturen is er een NetworkStream nodig. Deze NetworkStream is aangemaakt door via de aangemaakte TCP client een stream te openen. Op deze stream is het mogelijk om een bericht naar plain text te encoden in ASCII en naar DaHaus te sturen in de vorm van Bytes. Als deze bytes zijn verstuurd moet het programma de reactie van DaHaus opvangen. De reactie van DaHaus is ook in plain text en wordt op de zelfde stream terug gestuurd.

Om deze reactie te lezen resetten de webapplicatie de bytes die de studenten gebruikten voor de message, deze worden dan gevuld met wat er van de stream gelezen wordt. Deze bytes worden dan weer terug gezet naar text. Deze text wordt dan in een string gezet en in de response textbox geprint.

De stream en client worden daarna nog niet gesloten. Eerst wordt de ëxit"message naar DaHaus gestuurd. Hierdoor sluit DaHaus de connectie zelf. Daarna wordt de client en de stream geclosed. Deze exit message wordt dan niet geprint, anders komt er na elk commando "Bye bye" geprint in de response textbox wat het alleen maar onnodig opvult. De stream en client worden dus pas geclosed nadat DaHaus zelf is geëxit. Als de exit message namelijk niet verstuurd werd dan zou de connectie toch nog blijven openstaan bij DaHaus en zou er even goed nog een nieuwe connectie worden gemaakt bij DaHaus bij een nieuwe message. Hierdoor zou er dan een ophoping aan connecties ontstaan bij DaHaus. Dit gaf in kleine tests geen problemen maar bij het opschalen van gebruikers zal het uiteindelijk te veel stress kunnen veroorzaken bij DaHaus.

4.0.2 Commando's

De commando's die naar DaHaus worden gestuurd kan het best via knoppen worden gedaan. Hierdoor hoeft de gebruiker dan namelijk niet alle commando's uit hun hoofd te kennen of op

te zoeken. De makkelijkste optie was om 2 radiobutton lists te maken. Één radiobuttonlist om een object(lamp, window, heater, alle lampen of alle windows) te selecteren. De andere radiobuttonlist zou dan zijn voor de actie die de objecten moeten uitvoeren. Door een lange if else statement te maken is het daarmee mogelijk om voor elke combinatie het bijpassende commando te sturen. Om het commando op te sturen moet de gebruiker dan op de Send button drukken. Als er een ongeldige combinatie wordt gemaakt dan wordt dat in de Response textbox gezet, deze wordt dan niet naar DaHaus gestuurd. De heater wordt van temperatuur veranderd door de Heater en Heater Temp radio buttons te selecteren en in de heater textbox een value te geven. Deze wordt dan gecheckt of het wel een geldig getal is. Als dit getal boven de 35 graden is, dan wordt dit veranderd in 35 graden, dat is namelijk de maximale waarde voor de heater. Zelfde gebeurd met onder de 12 graden, als er een getal onder de 12 ingevuld wordt, dan wordt de heater tot 12 graden gezet. Dat is dus de minimale waarde voor de heater.

4.0.3 Problemen

Bij de connectie maken met DaHaus zijn veel problemen opgetreden. Er is begonnen bij de commando's op te halen door middel van het "help"commando te sturen naar DaHaus via Putty, op blackboard stond hoe dit gedaan moest worden. Dit zelf leverde geen problemen op. Nu was er toegang tot alle commando's. Daarna moest gekeken worden hoe de connectie met DaHaus gemaakt kon worden. Dit werd gedaan door middel van een TCP client. Deze TCP Client creëren koste tijd om te begrijpen, maar gaf daarnaast geen problemen. Daarna is er uitgezocht hoe de commando's gestuurd konden worden. Dit is geprobeerd op meerdere manieren. Ten eerste werd geprobeerd commando's te sturen door een stream te openen en door middel van handmatig de bytes in de stream te sturen. En daarna de reactie te lezen. Hierbij bleef de stream steken op het lezen van de reactie. Het was niet gelukt om dit probleem op te lossen, er is geprobeerd de volgorde van acties te veranderen, maar verder is er geen informatie gevonden over waarom het niet werkte, er waren namelijk ook geen errors die konden uitleggen waarom de connectie vast liep, omdat dit niet werkte is er naar een andere manier gezocht. Deze manier bestond uit een StreamWriter en StreamReader, alleen hierbij trad hetzelfde probleem op. Daarna is er met behulp van Paul "peregrine"getroubleshoot waarom dit niet werkte. Na het proberen van communicatie tussen eigen gemaakte TCP clients en listeners, bleek dat dat wel werkte. Uiteindelijk is er uitgekomen dat het probleem heel simpel bleek. Er moest namelijk een extra regel in de string van het commando meegestuurd worden.

Het gemaakte programma dat commando's verstuurde was een makkelijk console programma dat na 1 commando meteen de connectie afsloot. Er moest dus nu een webpagina gemaakt worden die dan een beetje gebruiks vriendelijk is. Zoals als eerder al gezegd in het kopje Connectie was er eerst geprobeerd om een connectie te maken en open te houden totdat deze gesloten werd. Dit lukte dus niet.

4.1 Spelletjes

De spelletjes die de studenten op de site moeten verwerken zijn simpele spelletjes voor de gebruiker om te spelen op de webpagina zelf. Er kunnen spelletjes op staan zoals:

4.2 Tegeltjes

De tegeltjes worden in de vorm van een vaste vierkant op de webpagina gepresenteerd. In het opmaakbestand (CSS) wordt de middenkolom in een 'grid' veranderd van minimaal 3 kolommen en die worden elk opgevuld met een default plaatje. Op het moment zijn er standaard 9 tegeltjes en deze moeten opgevuld worden met een plaatje en een hyperlink op basis van de ingelogde persoon. (zie Figuur 7)

Eerst heeft de projectgroep 9 tegeltjes geplaatst op de Homepagina en deze geprobeerd op te vullen door middel van een 'SELECT'-SQL. Door in de while-loop elke Imagebutton zijn 'ImageUrl'-eigenschap te vervangen door informatie uit de database. Wat hierbij wel verplicht is, is dat de Imagebutton in kwestie dezelfde naam moeten hebben op een cijfer na.

Als dat lukt dan wil de projectgorep het aantal Tegels aanpassen op basis van de opgeslagen Hyperlinks van de gebruiker. Dus als de gebruiker 5 websites heeft geregistreerd, dat hij/zij dan ook 5 tegels krijgt (zie Figuur 8)

Om tot het eindresultaat te komen heeft de projectgroep een PSD gemaakt, Figuur 9.

In deze functie wordt op basis van de eerder genoemde 'SELECT'-SQL een button gecreëerd zolang de Datareader nog rijen teruggeeft. Er wordt eerst een imagebutton variabele gemaakt en daaraan worden alle eigenschappen toegevoegd: Een ImageUrl voor het icoon, een hyperlink voor het doorsturen en een omschrijving van de website voor de Tooltip.

Dan wordt de variabele van de imagebutton toegevoegd aan het grid in de middelste kolom en onstaat er door het CSS-bestand een net uitziende webdashboard met tegels.

5 Verantwoording gebruikte methodes

5.1 Create user

In dit hoofdstuk wordt het aanmaken van een standaard gebruiker besproken. Als een gebruiker zich nog niet heeft aangemeld, dan wordt hij/zij doorverwezen naar de Login pagina. Op deze manier kunnen gebruikers niet zomaar bij elke pagina komen door simpelweg de naam van de pagina in de URL aan te passen. De projectgroep heeft hiervoor een PSD gemaakt (zie Figuur 2).

In de eerste versies van deze functie werd in de Web.config bestanden bijgehouden of er een cookie aanwezig was genaamd "AuthenticationCookie". Tijdens het testen heeft de projectgroep besloten deze functie tijdelijk uit te zetten, omdat zij na elke pauze van 20 minuten werden teruggestuurd naar de Login pagina. Als reactie hierop heeft de projectgroep op elke pagina zelf een functie gezet die controleerd of de "AuthenticationCookieäanwezig is, en zo niet wordt men terugverwezen naar de login.

Tijdens de realisatie is de cookie meer gebruikt dan het web.config bestand voor de beveiliging van de pagina's.

Om ervoor te zorgen dat er één pagina is om naar terug te verwijzen, worden de login en de "Create useröp dezelfde webpagina geplaatst en wordt er gewisseld door middel van buttons.

Er worden een aantal gegevens van de gebruiker gevraagd, de gebruikersnaam, wachtwoord en e-mail worden gevalideert door een RegularExpressionValidator (regex):

- Gebruikersnaam, deze moet uniek zijn en 6-18 tekens bevatten. Regex:

  ```
  $(^[A-Z|a-z|0-9|_]{6,18}\$)$
  ```

- Wachtwoord, 6-15 tekens waarvan 1 hoofdletter en 1 cijfer.Regex:

  ```
  $(^(?=.*\d)(?=.*[A-Z])(.{6,15})\$)$
  ```

- E-mail, deze moet uniek zijn. regex:

  ```
  $([A-Z|a-z|0-9](\.|_){0,1})+[A-Z|a-z|0-9]\@([A-Z|a-z|0-9])+

  ((\.){0,1}[A-Z|a-z|0-9]){2}\.[a-z]{2,3}$
  ```

Naast de bovenstaande informatie krijgt de gebruiker van het programma een Rolnr, dit nummer geeft aan tot welke pagina's hij/zij toegang heeft. Nu hebben we alles ingevoegd, zie Figuur 5.

Het wachtwoord wordt nu nog in plain text opgeslagen en dat is niet veilig. Als volgende stap gaat de projectgroep de wachtwoorden 'hashen'. Dit wordt gedaan in het volgende figuur (Figuur 6). In dit figuur is ook te zien dat in de database een onherkenbare serie letters en cijfers te zien is.

Na het inschrijven van bovenstaande gegevens wordt de gebruiker terug verwezen naar de Login pagina.

5.2 Inloggen

Voor het inlog gedeelte van het systeem zijn de studenten zelf gaan programmeren. Dit omdat de studenten dit een leuke uitdaging leek. Bij het maken van de inlog hebben de studenten een psd gemaakt zoals te zien is in Figuur 3. Het Programma is opgebouwd uit een aantal delen deze zijn Het ophalen van het wachtwoord uit de Database. Vervolgens het wachtwoord omzetten naar byte, daarna dan het opsplitsen van de Salt en de Hash. vervolgens de Slat voor het ingevulde wachtwoord zetten. Dit Wordt dan gehashed en als laatst word er vergeleken of de hash van het wachtwoord uit de database en het net gehashde wachtwoord het zelfde zijn, als dit zo is dan word er een cookie gemaakt die de gebruikersnaam onthoud en de gebruiker door stuurt naar de homepage. Wanneer dit niet het geval is krijgt de gebruiker een foutmelding dat het wachtwoord niet correct is. De code is te zien in Figuur 4.

6 Analyse van de gegevens

6.1 Dataverzameling

De Studenten hebben voor een groot deel gebruik gemaakt van de gegeven powerpoints van de lessen Webprogrammeren. ook hebben we een aantal dingen opgezocht op internet hier de links die we gebruikt hebben: `https://msdn.microsoft.com/en-us/library/ms525800(v=vs.90).aspx` (bekeken op 19-01-2018, Geschreven door Microsoft),
`https://stackoverflow.com/questions/13058574/check-if-cookie-exists`(Bekeken op 19-01-2018, geschreven door zmbq).

6.2 Analyse

De powerpoints waren van een betrouwbare bron. Ze komen namelijk van hun docent die ze les heeft gegeven in webprogrameren en het is ook 1 van de docenten die het project beoordeeld hierdoor weten de studenten ook meteen dat het op de manier gaat als de opdrachtgever wil. De eerste link is afkomstig van Microsoft een betrouwbare bron, omdat het programma waar de studenten hun webpagina op maken ook van Microsoft is. De Tweede links is van de site stackoverflow op deze site stellen mensen vragen over dingen waar ze niet uitkomen met hun programma's. De antwoorden zijn niet altijd betrouwbaar, maar in dit geval werkte het antwoord wel.

6.3 Deelconclusie/ aanbeveling

De studenten zijn er achtergekomen dat de powerpoints heel handig zijn om bij de hand te hebben voor als je ergens niet op komt. Verder zijn de voorbeelden van Microsoft handig om op te zoeken als je iets niet meer compleet weet. De site stackoverflow is handig als je weet wat je wilt alleen niet weet hoe je het uitvoert. De studenten zouden aanbevelen om zeker de powerpoints te gebruiken en als je het daar niet in kunt vinden kun je altijd nog internet raadplegen.

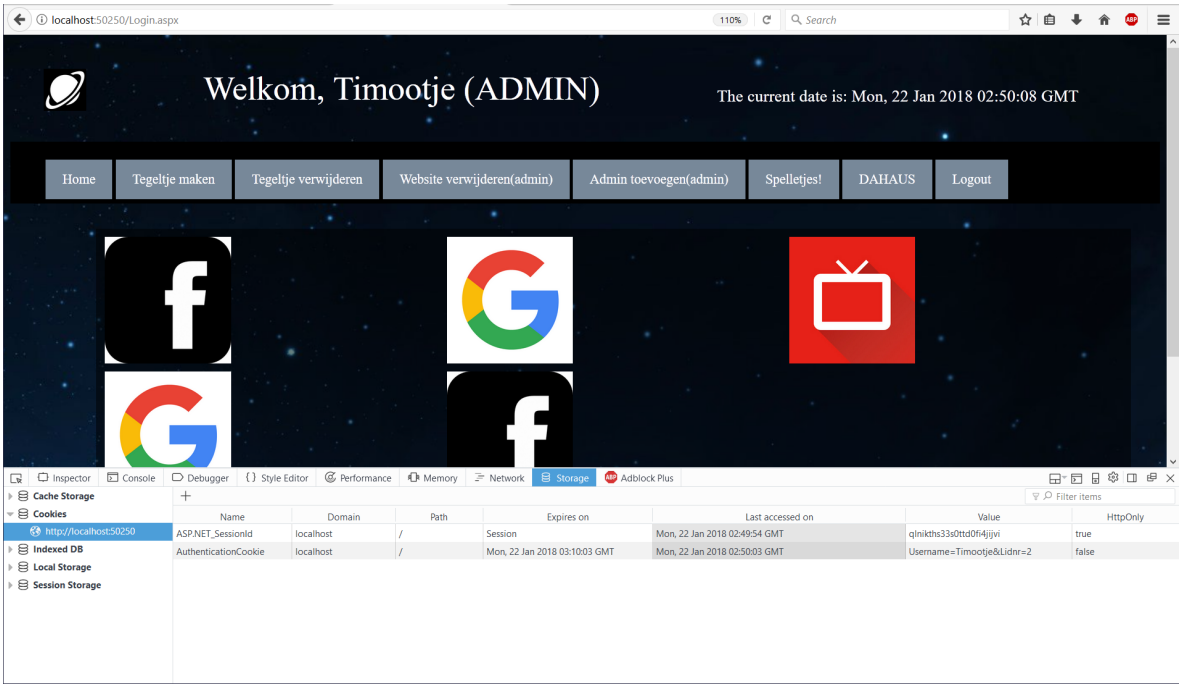
7 Eindconclusie

De studenten zijn er achter gekomen dat het meer werk is dan alleen maar een paar dingen op de pagina slepen zoals in de lessen werd gedaan om, maar dat er veel meer bij komt kijken bij een echte webpagina maken met alle eisen zoals beschreven staat in Hoofdstuk 2 staat beschreven. Verder is het handig als zoals bij de studenten gebeurde dat er een les als database uitvalt voor het begin van de periode dat je er zelf gewoon mee aan de slag gaat dan word je achterstand niet te groot.

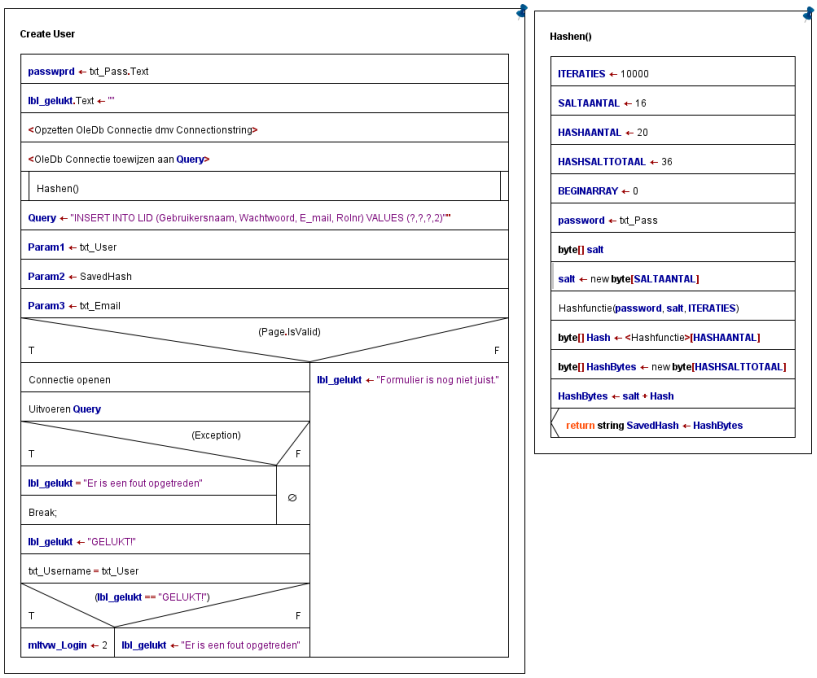
8 Aanbeveling

De aanbeveling van de studenten is om het werk niet te onderschatten en op tijd er mee te beginnen zodat je tijd over kan houden voor extra dingen. Die je op je webpagina wilt zetten. Verder als er een les niet gegeven word om de één of andere reden zoek iemand die er het begrijpt en kan uitleggen en vraag om hulp wanneer nodig, ook aan je groepsgenoten. het is beter hulp te vragen dan het slecht product te leveren.

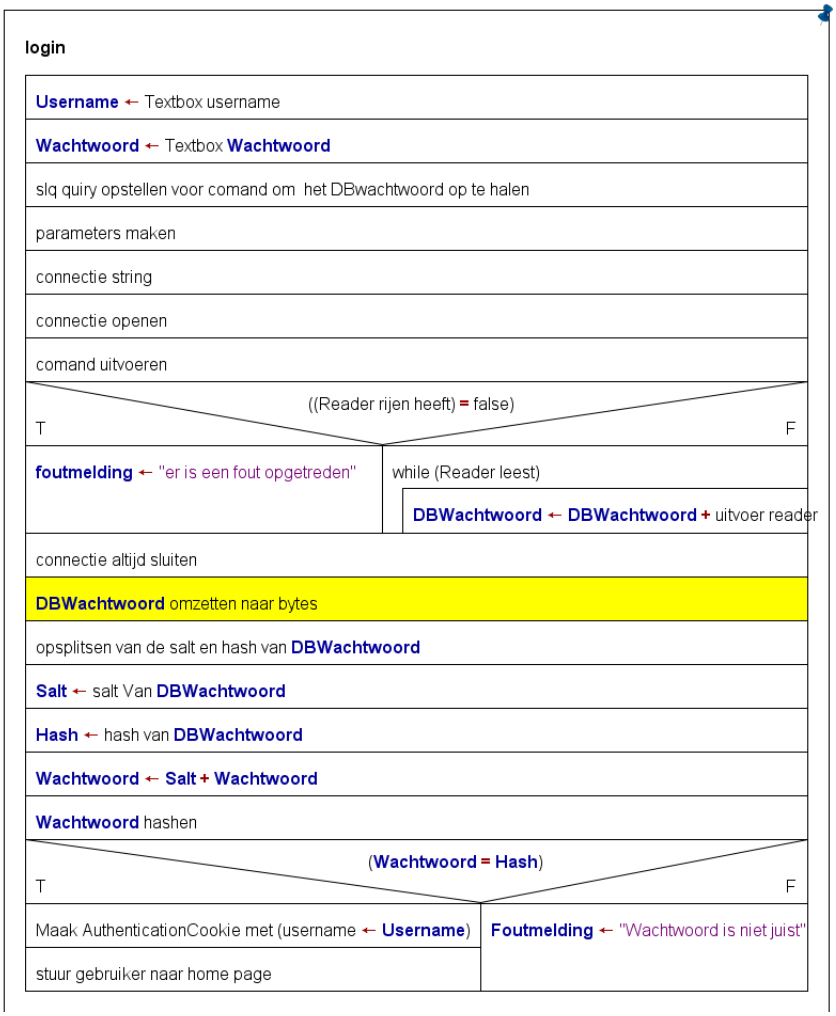
9 bijlagen



Figuur 1: De niet verwijderde cookie.



Figuur 2: PSD van de functie 'Create_user'



Figuur 3: psd van de Login

```
protected void btn_Login_Click(object sender, EventArgs e)
{
    string Wachtwoord = "";
    int RolNr;
    Connectie.ConnectionString = ConfigurationManager.ConnectionStrings["Harry"].ToString();
    Query.Connection = Connectie;

    Query.CommandText = "SELECT Wachtwoord, Rolnr FROM LID WHERE Gebruikersnaam = ? ";
    OleDbParameter Param1 = new OleDbParameter();
    Param1.Value = txt_Username.Text;
    Query.Parameters.Add(Param1);

    try
    {
        Connectie.Open();
        OleDbDataReader Leesding = Query.ExecuteReader();
        while (Leesding.Read())
        {
            Wachtwoord = Leesding["Wachtwoord"].ToString();
            RolNr = Convert.ToInt32(Leesding["Rolnr"]);
        }
    }
    catch (Exception exc)
    {
        lbl_gelukt.Text = exc.ToString();
    }
    finally { Connectie.Close(); }

    //converteren string naar bytes
    byte[] DBhash = Convert.FromBase64String(Wachtwoord);


    //AANMAKEN SALT VAN DBHASH
    byte[] DBsalt = new byte[16];
    Array.Copy(DBhash, 0, DBsalt, 0, 16);
    //Aanmaken en converten hash uit ww
    byte[] Controle_DBhash = new byte[20];
    Array.Copy(DBhash, 16, Controle_DBhash, 0, 20);
    string DBww = Convert.ToBase64String(Controle_DBhash);

    //HASHEN
    var pbkdf2 = new Rfc2898DeriveBytes(txt_Password.Text, DBsalt, 10000);
    byte[] CtHash = pbkdf2.GetBytes(20);
    string ControlePass = Convert.ToBase64String(CtHash);

    if (ControlePass == DBww)
    {
        //toevoegen cookie met username + lidnr
        HttpCookie Koekje = new HttpCookie("AuthenticationCookie");
        Koekje.Values.Add("Username", txt_Username.Text);
        Koekje.Expires = DateTime.Now.AddMinutes(20);
        Response.Cookies.Add(Koekje);

        //doorsturen naar home.aspx
        Server.Transfer("~/MEMBERS/Home.aspx");

    }

}
```

Figuur 4: Code van de login

```
namespace DOOMotica_1._2
{
    2 references | Timo van der Steenhoven, 1 day ago | 1 author, 1 change
    public partial class Login : System.Web.UI.Page
    {
        //aanmaken connectie object, query opbject en 3 parameters voor het aanmaken account
        OleDbConnection Connectie = new OleDbConnection();

        OleDbCommand Query = new OleDbCommand();
        OleDbParameter Param1 = new OleDbParameter();
        OleDbParameter Param2 = new OleDbParameter(); // deze heet express 2, ivm MS-veiligheid enzo
        OleDbParameter Param3 = new OleDbParameter();
        0 references | Timo van der Steenhoven, 1 day ago | 1 author, 1 change
        protected void Page_Load(object sender, EventArgs e)
        {

        }
        0 references | 0 changes | 0 authors, 0 changes
        protected void btn_CreateUser_Click1(object sender, EventArgs e)...

        0 references | 0 changes | 0 authors, 0 changes
        protected void btn_Terug_Click(object sender, EventArgs e)...

        0 references | 0 changes | 0 authors, 0 changes
        protected void btn_Create_Click(object sender, EventArgs e)
        {
            // connectie aanwijzen
            Connectie.ConnectionString = ConfigurationManager.ConnectionStrings["Harry"].ToString();
            Query.Connection = Connectie;

            //opzetten query
            Query.CommandText = "INSERT INTO LID (Gebruikersnaam, Wachtwoord, E_mail) VALUES ('?','?','?')";
            //parameters invoeren
            Param1.Value = txt_User.Text;
            Param2.Value = txt_Pass.Text;
            Param3.Value = txt_Email.Text;
            //parameters toevoegen aan query, op volgorde
            Query.Parameters.Add(Param1);
            Query.Parameters.Add(Param2);
            Query.Parameters.Add(Param3);

            try
            {
                Connectie.Open();
                lbl_gelukt.Text = "";
                Query.ExecuteReader();
                lbl_gelukt.Text = "GELUKT!";
            }
            catch (Exception exc) { lbl_gelukt.Text = exc.ToString(); }
            finally { Connectie.Close(); }
        }
    }
}
```

LID

Lidnr	Gebruikersn	Wachtwoord	E_mail	Click to Add
1	asg123	sadcdas	dsa@dfgklj.mv	
2	USERNAME	PASSWORD1	EMAIL@ADRES	
(New)				

Figuur 5: Opslag plain text en bijbehorende code

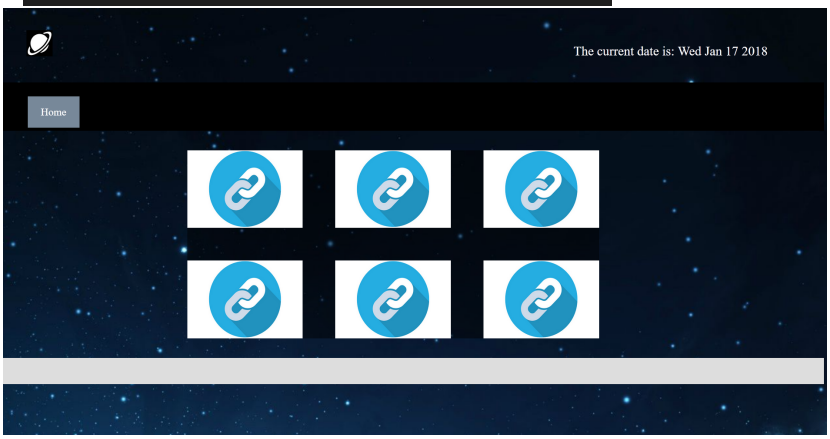
```
//in deze array komt de salt
byte[] salt;
// het genereren van de salt, 16 'random' cijfers
new RNGCryptoServiceProvider().GetBytes(salt = new byte[SALTAANTAL]);
// hashen van het salt + de inhoud van de textbox
Rfc2898DeriveBytes pbkdf2 = new Rfc2898DeriveBytes(txt_Pass.Text, salt, ITERATIES);

// bovenstaande hash wordt in een array geplaatst
byte[] hash = pbkdf2.GetBytes(HASHAANTAL);

//Een nieuwe array maken waar de hash + de salt in wordt opgeslagen
byte[] hashBytes = new byte[HASHSALTTOTAAL];
//De hash en de salt in bovenstaande array plaatsen
Array.Copy(salt, BEGINARRAY, hashBytes, BEGINARRAY, SALTAANTAL);
Array.Copy(hash, BEGINARRAY, hashBytes, SALTAANTAL, HASHAANTAL); //de laatste 2 getallen geven aan d

//nu nog de salthasharray converteren naar een string
string SavedHash = Convert.ToBase64String(hashBytes);
```

ADMIN | LID

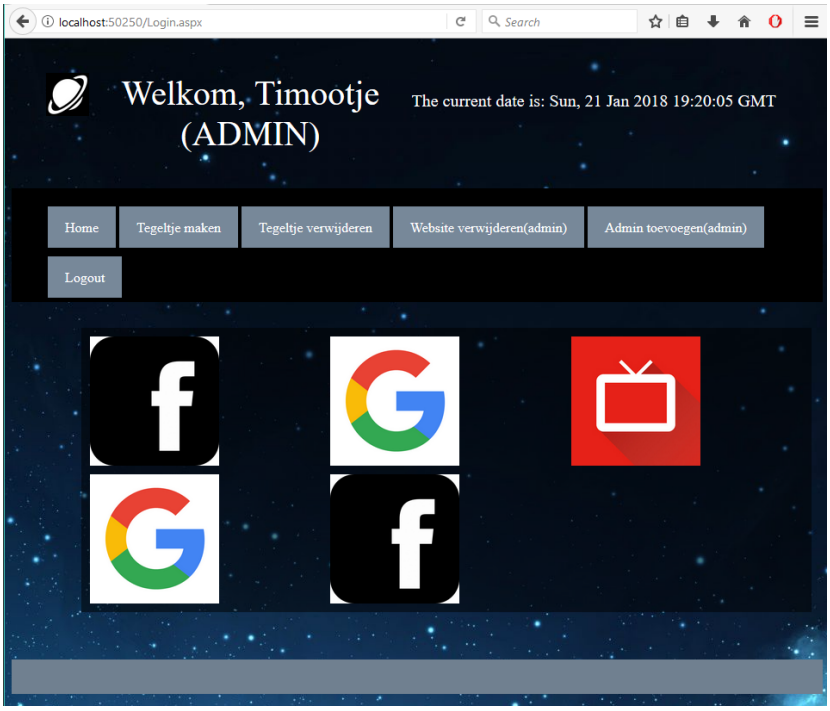
Lidnr	Gebruikersn	Wachtwoord	E_mail	Click to Add
1	asg123	sadcdas	dsa@dfgklj.mvb	
2	USERNAME	PASSWORD1	EMAIL@ADRES.nl	
3	tomeuh	vL9UjBDrdp6uCksdNFLboulexMsAAAAAAAAAAAAAAAAAAA	DSD@asd.nl	
4	dingie	7MXhHQWRaOPqd4FITRMS2XN59sYAAAAA/AAAAAAAAAAAAAAAA	asd@asd.nl	
(New)				

Figuur 6: De code waaruit de hash wordt gegenereerd.

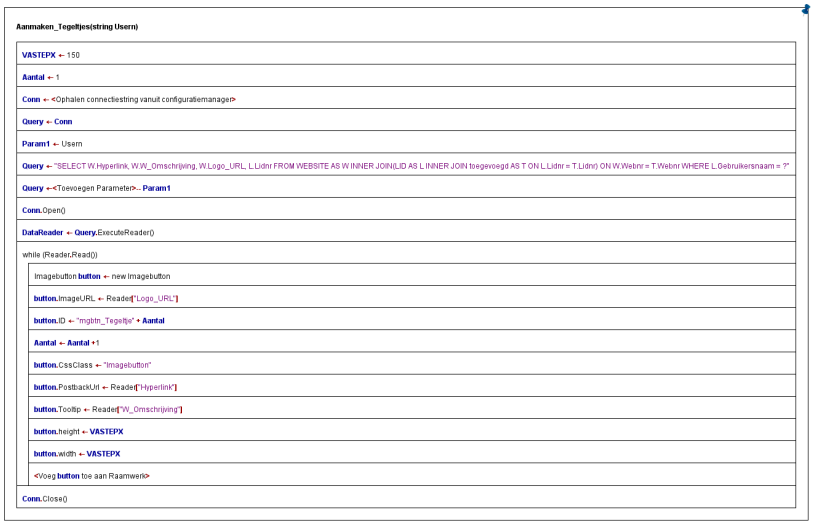
```css
/*Opmaak van alle hokjes met links*/
.Tegelwerk {
    display: grid;
    grid-template-columns: auto auto auto;
    grid-column-gap: 50px;
    grid-row-gap: 50px;
}

.tegeltje .imagebutton {
    background-color: rgba(59, 106, 187,0.5);
    height: 130px;
    width: 100%;
}



.tegeltje a {
    float: none;
    display: block;
    color: white;
    text-align: center;
    padding: 14px 20px;
    text-decoration: none;
}
```



Figuur 7: Het resultaat van de CSS opmaak op de tegels.



Figuur 8: Meerdere persoonlijke tegels



Figuur 9: PSD van de functie 'Aanmaken_Tegels'