Ledenadministratie

PHP en MySQL



Door: Daniel Treep

Studentnummer: 312840519

Datum: 20-02-2023

Inhoud

[Ontwikkelomgeving 2](#_Toc127782641)

[Belangrijk 3](#_Toc127782642)

[Database 3](#_Toc127782643)

[Families 3](#_Toc127782644)

[Leden 4](#_Toc127782645)

[Groepen 4](#_Toc127782646)

[Jaren 4](#_Toc127782647)

[Contributies 4](#_Toc127782648)

[Gebruikers 5](#_Toc127782649)

[Werking applicatie 5](#_Toc127782650)

[MVC Object georiënteerd 5](#_Toc127782651)

[Controller 5](#_Toc127782652)

[View 6](#_Toc127782653)

[Model 6](#_Toc127782654)

[Navigeren in de applicatie 6](#_Toc127782655)

[Inloggen, registreren en verwijderen 7](#_Toc127782656)

[CRUD operaties 7](#_Toc127782657)

[Pagina’s 8](#_Toc127782658)

[Familieoverzicht 8](#_Toc127782659)

[Ledenoverzicht 8](#_Toc127782660)

[Groepenoverzicht 8](#_Toc127782661)

[Contributies 9](#_Toc127782662)

[Jaaroverzicht 9](#_Toc127782663)

[Profiel 9](#_Toc127782664)

[Testverslag 9](#_Toc127782665)

[Verbeteringen 9](#_Toc127782666)

# Ontwikkelomgeving

Ik werk in Visual Studio Code en heb het boek gevolgd en AMPPS geïnstalleerd. Dit programma heeft vervolgens Apache, MySQL en PHP 7.4 geïnstalleerd. Mijn ontwikkelstack is dus WAMP.

Als ik wil werken aan de opdracht zet ik de Ampps app aan. Deze zet vervolgens de Apache server aan waardoor ik in de localhost kan werken met PHP en MySQL in de browser. Localhost laat de bestanden zien van de ‘www’ map uit de Ampps bestanden. Hier heb ik een map aangemaakt voor de ledenadministratie waarin ik vervolgens alle PHP bestanden heb gemaakt en opgeslagen. Zodra ik een bestand heb aangepast / geschreven herlaad ik de pagina in de browser en zijn de veranderingen te zien.

Verder heb ik phpmyadmin gebruikt voor het maken van de database en test ik de queries in Command Prompt nadat ik ben ingelogd in MySQL.

## Belangrijk

Het SQL creatiescript en database export bestand bevind zich in de sql map. Als dit bestand wordt uitgevoerd wordt er een nieuwe database gemaakt genaamd ledenadministratie. Vervolgens worden alle tabellen gemaakt en de waarden ingevoerd. Mocht u een andere naam willen kunt u de naam aanpassen op lijn 23 en 24 naar uw keuze.

De gegevens om connectie te maken met de database bevinden zich in model/pdoconnection. Hier kunt u uw eigen gegevens invoeren.

De CRUD operaties zijn alleen beschikbaar als admin is ingelogd. Het wachtwoord is ‘admin’.

# Database

Afbeelding met tekst

Automatisch gegenereerde beschrijvingEr zijn 6 tabellen in de database en het ziet er als volgt uit:

## Families

De familietabel bestaat uit de volgende kolommen:

* Family ID: Dit is de primary key en auto increment als er een nieuwe familie wordt toegevoegd. Deze kolom is verbonden met de family ID kolom bij de leden. Meerdere leden kunnen namelijk tot 1 familie behoren. De waarde bestaat uit cijfers en is daarom INT.
* Surname: Dit is de achternaam van de familie. Deze hoeft niet uniek te zijn want er kunnen meerdere families zijn met dezelfde achternaam. De waarde kan bestaan uit letters en is daarom VARCHAR.
* Adress: Op dit adres woont de familie. Deze is wel uniek aangezien er 1 familie op 1 adres woont. De waarde kan bestaan uit letters en cijfers en is daarom VARCHAR.

## Leden

De ledentabel bestaat uit de volgende kolommen:

* Member ID: Dit is de primary key en auto increment als er een nieuw lid wordt toegevoegd. Deze kolom is verbonden met contributies. Een contributie behoort tot een lid maar er kunnen ook meerdere contributies behoren tot 1 lid. De waarde bestaat uit cijfers en is daarom INT.
* Name: Dit is de naam van het lid. De waarde kan bestaan uit letters en is daarom VARCHAR.
* Birthday: De geboortedatum van een lid. De waarde bestaat uit een datum en is daarom date.
* Group ID: Deze kolom laat zien tot welke groep het lid behoort. Elke groep heeft een uniek ID en hier wordt opgeslagen welke bij het lid hoort. Een lid kan tot 1 groep behoren. De waarde bestaat uit cijfers en is daarom INT.
* Family ID: Deze kolom verwijst naar de familie tabel en laat zien bij welke familie het lid hoort. Een lid behoort tot 1 familie. De waarde bestaat uit cijfers en is daarom INT.

## Groepen

De groepentabel bestaat uit de volgende kolommen:

* Group ID: Dit is de primary key en auto increment als er een nieuwe groep wordt toegevoegd. Dit geeft de unieke ID aan van de groep. Deze kolom is verbonden met de group ID kolom bij de leden. Een groep kan bestaan uit meerdere leden. Ook is hij verbonden met contributies zodat het contributie bedrag bepaalt kan worden. Er kunnen meerdere contributies behoren tot 1 groep. De waarde bestaat uit cijfers en is daarom INT.
* Group name: De naam van de groep. De waarde kan bestaan uit letters en is daarom VARCHAR.
* Group description: De omschrijving van de groep. De waarde kan bestaan uit letters en is daarom VARCHAR.
* Discount: Dit laat de korting zien die de groep heeft. De waarde bestaat uit cijfers en is daarom INT.

## Jaren

De jarentabel bestaat uit 1 kolom, namelijk de boekjaren. De jaren zijn uniek en zijn daarom de primary key. De kolom is verbonden met de contributies. Elke contributie behoort tot 1 boekjaar maar er kunnen meerdere contributies bij 1 boekjaar horen. De waarde bestaat uit cijfers en is daarom INT.

## Contributies

De contributietabel bestaat uit de volgende kolommen:

* Contribution ID: Dit is de primary key en auto increment als er een nieuwe contributie wordt toegevoegd. Dit geeft de unieke ID aan van een betaalde contributie. De waarde bestaat uit cijfers en is daarom INT.
* Age: De leeftijd van het lid dat de contributie heeft betaald in dat boekjaar. De waarde bestaat uit cijfers en is daarom INT.
* Group ID: De waarde van de groep waarbij het lid op dat moment bij behoort. Deze kolom is daarom verbonden met de group ID van de groepen tabel. Een contributie behoort tot 1 groep maar 1 groep kan meerdere contributies hebben. De waarde bestaat uit cijfers en is daarom INT.
* Amount: Dit is de betaalde hoeveelheid op dat moment. Dit wordt bepaalt door de leeftijd van het lid, bij welke groep het lid hoorde op dat moment en hoeveel korting daar bij hoort. De waarde bestaat uit cijfers tot 2 achter de 0 en is daarom float.
* Year ID: Dit geeft aan in welk jaar de contributie is betaald. Deze is verbonden met de jaren tabel. Een contributie behoort altijd tot een jaar maar 1 jaar kan meerdere contributies hebben. De waarde bestaat uit cijfers en is daarom INT.
* Member ID: Dit laat zien voor welk lid de contributie is betaald. Deze kolom is verbonden met de ledentabel. Een contributie behoort altijd tot 1 lid maar 1 lid kan meerder contributies hebben. De waarde bestaat uit cijfers en is daarom INT.

## Gebruikers

De gebruikers tabel bestaat uit de volgende kolommen:

* User ID: Dit is de primary key en auto increment als er een nieuwe gebruiker wordt toegevoegd. Het geeft de unieke code aan van elke gebruiker. De waarde bestaat uit cijfers en is daarom INT.
* Username: Dit geeft de gebruikersnaam aan van de gebruiker. Hiermee logt hij in en moet uniek zijn. De waarde kan bestaan uit letters, cijfers en tekens en is daarom VARCHAR.
* Password: Dit geeft het wachtwoord aan van de gebruiker. Hiermee logt hij in. Het wachtwoord is beveiligd door een HASH. De waarde kan bestaan uit letters, cijfers en tekens en is daarom VARCHAR.

# Werking applicatie

## MVC Object georiënteerd

De applicatie is gebaseerd op de Model View Controller (MVC) architectuur. Elke onderdeel heeft zijn eigen taak. De Model is verantwoordelijk voor het creëren, lezen, updaten en deleten (CRUD) van de data uit de database en geeft niet om de vormgeving. De View is verantwoordelijk voor de presentatie van de data en de User Interface waarmee de gebruiker handelingen kan uitvoeren. Zodra een handeling is gedaan door de gebruiker wordt dit door de Controller opgevangen. Hier wordt bepaald welke actie er wordt ondernomen door functies aan te roepen van de Model en welke pagina of elementen de gebruiker te zien krijgt. De Controller is dus een link tussen de View en de Model.

### Controller

In deze applicatie wordt op de startpagina ‘ledenadministratie.php’ meteen een Controller object gemaakt en daar de invoke functie van aangeroepen. Hierbij is een PDO object als argument bijgevoegd. Dit PDO object is verantwoordelijk voor de connectie met de database en zonder zou er geen toegang zijn om data aan te passen of te lezen. Dit PDO object wordt vervolgens aan alle methodes van de Models als argument bijgevoegd zodat die een query kunnen uitvoeren op de database.

In de controller worden allereerst variabalen gedeclareerd en vervolgens geïnstantieerd bij de construct functie. De variabelen zijn dan instanties van de verschillende Model objecten. Hierna kan de controller een methode aanroepen op de gekozen Model object variabele.

De controller zorgt er vervolgens voor dat het begin van de HTML wordt getoond. Daarna welke actie er ondernomen moet worden door de POST array te lezen. Deze wordt geupdate als de gebruiker bepaalde buttons klikt. Dit kan zijn om een pagina of crud popup te laten zien, of om een operatie uit te voeren op de database.

Verder zijn er nog twee functies te vinden in het controller bestand. GetAge ontvangt twee parameters, namelijk twee datums. Vervolgens berekend de functie het verschil tussen de twee datum en retourneert die. Ik gebruik dit om de leeftijd te bepalen van een persoon.

GetGroup heeft 1 parameter, namelijk een leeftijd. Op basis van de leeftijd wordt een group ID geretourneerd. Zo deel ik iemand in een groep op basis van leeftijd.

### View

De view map bestaat uit de mappen includes en paginas. Daarnaast zijn er ook de start en eind bestanden te vinden van de HTML.

In de map paginas zijn de verschillende pagina’s te vinden die de gebruiker kan opvragen met de verschillende overzichten. In de map includes zijn de popups te vinden van de verschillende CRUD operaties van elke pagina.

In het HTML start bestand bevindt zich de head met daarin de link naar de CSS, het begin van de body met de header en het begin van main. In de header bevindt zich een plek om in te loggen of te registreren. In de main sectie bevindt zich de navigatie. Hier zijn linkjes te zien de verwijzen naar de verschillende pagina’s. Daarna begint de div waar de content wordt getoond op basis van de input van de gebruiker.

In het eind bestand is de HTML te vinden van de footer en de afsluitende body en html tags.

### Model

Ik heb ervoor gekozen om de model op te splitsen in meerdere models om het overzichtelijk te houden. Zo heeft elke pagina waarbij crud operaties moeten uitgevoerd kunnen worden een eigen model. Ook de login heeft een eigen model omdat het met de database interacteert.

Elke model heeft een paar methodes. Voor het lezen van de data gebruik ik get functies, voor het toevoegen van data: create functies, voor het updaten: update functies en voor het verwijderen: delete functies. De get functies retourneren alle opgevraagde data naar een variabele in de controller. Dit wordt vervolgens gelezen door de view en getoond op de pagina.

Een methode werkt meestal zo:

* De benodigde waarden worden gelezen uit de POST array.
* Ik formuleer een query voor wat ik wil doen of laten zien met eventueel de POST data.
* Ik voer de actie uit op de database en retourneer eventueel de resultaten array.

## Navigeren in de applicatie

Op de pagina is een navigatie met linkjes naar de verschillende pagina’s. Dit werkt als volgt:

* De gebruiker klikt op de link
* In de link wordt de waarde toegevoegd van de pagina aan de ‘page’ property in de GET array.
* De controller leest deze waarde en slaat die op in zijn eigen ‘page’ variabele. Als deze niet bestaat is het null en blijft de pagina leeg. Als de ‘page’ variabele overeenkomt met de link krijgt deze een CSS class toegewezen om te laten zien dat deze actief is.
* De controller ziet dat waarde overeenkomt met een van de cases in het switch statement.
* De controller voert de code uit van de case die overeenkomt. Hierin staat welke pagina er moet worden getoond.
* De pagina met inhoud verschijnt in de content div.

## Inloggen, registreren en verwijderen

Als een gebruiker op de startpagina komt is er een mogelijkheid om in te loggen. Dit gaat zo:

* De gebruiker typt zijn gebruikersnaam en wachtwoord in.
* De gebruiker klikt op login. Dit is een submit button met de naam ‘login’. Dit wordt toegevoegd aan de POST array door de post methode, samen met de gebruikersnaam en het wachtwoord.
* De controller leest dat de gebruiker wil inloggen uit de POST array.
* De controller roept de loginUser methode aan van de Login Model en geeft het PDO mee als argument om met de database te kunnen interacteren.
* De methode slaat de waardes op uit de gebruikersnaam en wachtwoord en zet eventuele html characters om in html veilige code.
* Er wordt een zoekopdracht gedaan om te kijken of de gebruikersnaam aanwezig is in de database.
* Is dit het geval wordt er gekeken of het wachtwoord overeenkomt. Zo ja, wordt er een sessie variabele aangemaakt met de gebruikersnaam. Dit wordt gebruikt om de gebruiker te begroeten.
* De pagina herlaad en retourneert de waarde ‘true’ naar de sessie variabele ‘loggedin’ in de controller. Als deze op true staat krijgt de gebruiker nieuwe opties in de header. Namelijk om naar zijn profiel te gaan of weer uit te loggen.
* Is de gebruikersnaam niet aanwezig of klopt het wachtwoord niet, wordt er een foutmelding getoond met de reden van de fout.

Heeft de gebruiker nog geen account dan kan hij zich registreren. De stappen zijn bijna hetzelfde. Er wordt gekeken of de gebruikersnaam aanwezig is, want die moet uniek zijn. Als dit het geval is wordt er een foutmelding getoond. Als de gebruikersnaam niet aanwezig is wordt het wachtwoord beveiligd met een hash en wordt de informatie opgeslagen in de database. Hierna kan de gebruiker inloggen met de geregistreerde data.

Op de profielpagina kan de gebruiker zijn account verwijderen door het wachtwoord in te typen en op de knop verwijderen te klikken. De methode deleteUser wordt aangeroepen door de controller. Er wordt geverifieerd of het wachtwoord klopt met het wachtwoord wat is gelinkt aan de sessie ‘username’ variabele en als dit klopt wordt het account verwijderd uit de database.

## CRUD operaties

**De CRUD operaties zijn alleen beschikbaar als de admin is ingelogd** zodat niet elke gebruiker de database kan aanpassen. Het wachtwoord is ‘**admin’**. Als de admin de data wilt aanpassen kan hij op de buttons klikken. Dit gaat als volgt:

* De admin klikt op een van de buttons boven de tabel: toevoegen, aanpassen of verwijderen.
* Elke button is een submit button met een unieke naam.
* Deze naam wordt aan de POST array toegevoegd als er op wordt geklikt door de post method van de form.
* De controller leest welke naam er is toegevoegd en kiest uit de IF statements welke CRUD popup er wordt getoond. Elke CRUD popup is anders omdat het andere data nodig heeft.
* Als de informatie is ingevoerd kan er op de submit button worden geklikt. Elk informatie blokje heeft een unieke naam en wordt aan de POST array toegevoegd.
* De controller leest de naam van de submit en bepaalt welke methode van welk Model moet worden aangeroepen op basis van een IF, IF ELSE statement.
* De gekozen methode wordt uitgevoerd.
* In de methode wordt eerst eventueel de ingevoerde informatie gelezen en opgeslagen in variabelen.
* Daarna wordt een query uitgevoerd op de database. Dit voegt toe, leest, update of verwijderd data uit de database.
* De methode kan eventueel het resultaat als array retourneren.
* Deze resultaten array wordt opgeslagen in een variabele in de controller. De variabelen bevinden zich voordat de pagina wordt getoond door de controller en zijn daarom beschikbaar om te lezen door de pagina.
* De pagina leest of er data is en met een loop wordt per rij de data op de pagina laten zien.

## Pagina’s

### Familieoverzicht

De eerste pagina is een overzicht met de families. Hierbij zijn de familienamen, wie bij welke familie hoort, het aantal en het adres te zien. Ook is er te zien hoeveel totale contributie er dat jaar voor de familieleden bij elkaar betaald zou moeten worden.

Er kunnen families worden toegevoegd door de naam en het adres op te geven. Als er een familie wordt toegevoegd krijgt deze een uniek familie nummer. Er zijn dan nog geen leden toegewezen aan die familie. Dit kan bij het ledenoverzicht.

Aanpassen kan door het familienummer in te voeren en de informatie in te voeren. Verwijderen kan door het familienummer in te voeren. Dit kan echter alleen als er geen leden zijn met die familienaam en er nog geen contributie voor is betaald. Anders is het al opgeslagen in de database.

### Ledenoverzicht

De tweede pagina is een overzicht met alle leden. Op deze pagina is het lidnummer, achternaam, adres en geboortedatum te zien. Ook is de groep en huidige contributie te zien als hij nog lid zou zijn op dit moment.

Om een lid toe te voegen is de voornaam nodig en de geboortedatum. Ook kan de familienaam geselecteerd worden uit de beschikbare families. Op basis van de leeftijd wordt een lid automatisch ingedeeld in een groep en wordt de juiste contributie berekend. Hetzelfde geldt voor het aanpassen van een lid. Dit kan alleen bij een bestaand lid en daarvoor is het lid nummer nodig.

### Groepenoverzicht

De derde pagina is een overzicht met de verschillende groepen waarbij een lid kan horen. De groepen zijn ingedeeld op basis van leeftijd en hebben elk een eigen korting op de contributie.

Een groep kan worden toegevoegd door de naam, omschrijving en korting in te voeren. Hetzelfde geld voor aanpassen. Voor het verwijderen hoeft alleen de groep geselecteerd te worden.

### Contributies

De vierde pagina is een overzicht van alle betaalde contributies. Elk jaar wordt er contributie betaald en daarom is er een boekjaar te zien. Er staat voor welk lid de contributie is betaald en in welke groep het lid zat in dat boekjaar op basis van leeftijd. Ook staat het bedrag aan contributie wat is betaald.

Om een contributie toe te voegen is alleen het lidnummer en het boekjaar nodig. Vervolgens wordt berekend in welke groep het lid zat op dat moment en wordt de overige data getoond. Hetzelfde geldt voor het aanpassen van een contributie. Hier is alleen het bestaande contributie nummer voor nodig, net als bij een contributie verwijderen.

### Jaaroverzicht

De vijfde pagina is een jaaroverzicht van alle boekjaren. Hier is te zien hoeveel leden er in dat jaar waren en hoeveel contributie er in totaal is betaald door die leden.

### Profiel

Als een gebruiker is ingelogd kan hij op het profiel icoontje klikken. Dan opent de profiel pagina. Hier kan een gebruiker zijn informatie opzoeken door het lidnummer in te voeren.

De informatie laat de naam, achternaam, adres, geboortedatum, groep, huidige contributie, aantal jaar lid en totaal betaalde contributie zien. Daarbij wordt de gezochte persoon opgeslagen in een sessie variabele zodat als de gebruiker terug komt de informatie meteen klaar staat. Ook kan een gebruiker op deze pagina zijn account verwijderen door het wachtwoord opnieuw in te voeren en op de button te klikken.

# Testverslag

Alle functies die in de app aanwezig zijn werken naar behoren. Het toevoegen, aanpassen en verwijderen van data gaat allemaal goed als de juiste data wordt ingevoerd. Ook kan een gebruiker zich registreren, inloggen, account verwijderen en informatie opzoeken.

## Verbeteringen

Als er onjuiste informatie ingevoerd wordt, bijvoorbeeld bij het verwijderen van een niet bestaand persoon, gebeurt er niks en zou er een foutmelding getoond kunnen worden.