Heilige Familie IEPER

# Geïntegreerde Proef – 6I

**-Stageverslag Page Computers**
**-Helpdesk: HTML5 – CSS3 – PhP – JavaScript – MySQL**

**Oliver Lanckriet**
**6 Informaticabeheer**
**2017 - 2018**

# Woord vooraf

Deze bundel omvat mijn stagedossier van Page Computers en zowat alle code die gebruikt wordt in de werking van mijn helpdesk gemaakt in PhP, HTML, CSS, JavaScript en MySQL. Deze GIP is een vertegenwoordiging van de kennis die ik heb opgedaan tijdens mijn 2 jaar Informaticabeheer.

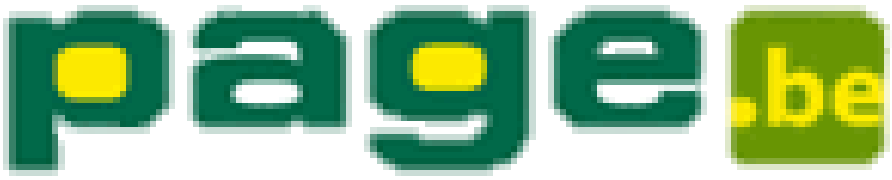
In het ontwikkelen en voorbereiden van mijn helpdesk is veel werk gekropen door verscheidene mensen. Hierdoor zou ik graag dhr. Staelens, dhr. De Wandel, dhr. Plets en mijn klasgenoten bedanken, zonder hen zou ik deze opdracht niet kunnen verwezenlijkt hebben.

Voor mijn stage wil ik dhr. Maes van Page Computers en collega's bedanken. Tijdens de twee weken die ik bij hen doorbracht heb ik veel geleerd over het werken in een informaticabedrijf en over hoe ik er tegenover sta.

Het laatste dankwoord gaat uit naar de jury die de tijd neemt om ons te evalueren, en ons een duwtje zullen geven richting de echte wereld.

# Inhoudstafel

# Stagedossier: Page Computers



**Inleiding**
In oktober heb ik stage gelopen bij de technische dienst van Page Computers, een computerwinkel te Poperinge die onderdeel is van Page Elektronica. Ik speelde assistent van mijn stagementor Jan en hielp hem bij de reparaties en het onderhoud van toestellen van onze klanten. Dit varieerde van het uitvoeren van virusscanners tot het vervangen van voedingen of problemen in het BIOS op te lossen. Ik leerde reparaties uitvoeren op desktops, laptops maar ook smartphones en printers. Het grootste deel van mijn stage besteedde ik achter de schermen, maar af en toe mocht ik ook klanten helpen over de toonbank.

## 1 Hard- en software

Tijdens mijn stage beschikte ik, samen met de werknemers van Page, over enkele HP-laptops die op de laatste versie van Windows 10 Pro draaiden, nl. de Fall Update. Al deze laptops hebben toegang tot de fileserver van het bedrijf waar op één volume de klantgegevens op staan die we bijhouden voor back-ups indien er iets fout zou lopen, en op een tweede volume de freeware die we gebruiken voor onze taken.

Deze freeware is bevat o.a. virusscanners voor malware, adware en spyware en enkele testprogramma's zoals CrystalDiskInfo om harde schijven te testen. Om verwarring tegen te gaan gebruikt deze fileserver een hoop verschillende netwerkadressen zodat werknemers met verschillende functies geen onnodige informatie te zien krijgen.

Toestellen aangesloten op de Wi-Fi krijgen standaard enkel netwerkadressen te zien die de technische dienst gebruikt, om specifiekere gegevens te vinden zoals bedrijfsdocumenten die op het netwerk staan zou men dus al goed overweg moeten kunnen met de netwerkadressen.

### 1.1 Servers- en netwerkapparatuur

De fileserver die Page gebruikt (fig. 1) staat beneden in de kelder en werkt op Windows Server 2008. Het bedrijf gebruikt twee switches, één in de productieafdeling en één in de computerwinkel om overal toegang tot het netwerk te hebben. Deze server kan worden bestuurd vanaf de computer van de netwerkbeheerder. De bekabeling van deze server is, na jaren gebruikt en af en toe aangepast te worden, een verse kom spaghetti geworden.

Vroeger had Page naar het schijnt een mailserver, maar die hebben ze een tijd geleden afgeschaft. Nu gebruiken de werknemers van Page Outlook om mails te lezen en te versturen.

Fig. 1: de server

### 1.2 Backups

Verder staat de server in RAID-5 om gebruik te maken van de handige pariteitscontrole, en worden back-ups van bedrijfsgegevens opgeslagen in de opslag van de CEO Luc Page die niet ver van de vestiging woont. Deze back-ups worden elke nacht gemaakt en om er voor te zorgen dat deze goed verlopen werkt men op het bedrijfsnetwerk met log-on-times zodat men geen wijzigingen maakt tijdens het back-uppen omdat bepaalde werknemers aan thuiswerken doen. Over de hardware van deze back-up installatie heb ik niets specifieks gehoord.

De netwerkbeheerder heeft ook een archief van bedrijfsdocumenten in digitale versie staan op zijn eigen opslag bij hem thuis. Deze back-ups bevinden zich buiten het fysieke bedrijf, dus als er bv. een brand zou zijn kan men nog de belangrijke en vrij actuele gegevens terugwinnen.

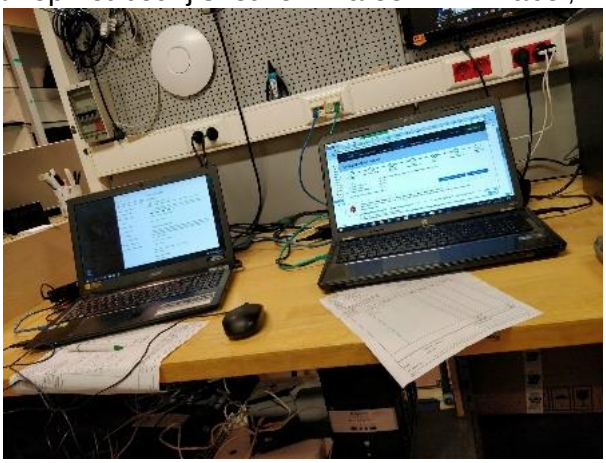
### 1.3 Softwarepaketten

De boekhouders bij Page werken voor facturatie met gegevens van producten, leveranciers, klanten enz. in een relationele databank, hiervoor gebruikt men het vooraf vermelde ISABEL. In de winkel maakt men voor onderhoud ook snel een servicerapport op d.m.v. de informatie omtrent de prijs van verschillende taken. Voor andere doeleinden heeft Page echter geen echte database.

### 1.4 Werkstations

Alle vaste werkstations en de stroomvoorziening van de laptops zijn verbonden met een UPS (Uninterruptible Power Supply) zodat men in het geval dat de stroom uitvalt toch nog eventjes hun pc kan gebruiken om alles op te slaan en correct af te sluiten. Met de UPS-en die men gebruikt krijgt men hier toch een halfuurtje voor, wat meer dan genoeg is. Dit heb ik één keer kunnen meemaken toen de stroom uit viel en de werkplaats enkel door de schermen van de op UPS aangesloten pc's verlicht werd.

Tijdens de reparaties sloten we onze toestellen aan op het bedrijfsnetwerk via een LAN-kabel, waarvan we een dozijn aansluitingen hebben in de werkplaats. Wi-Fi is ook beschikbaar als het toestel in kwestie geen netwerkpoorten heeft of als de reparatie te maken heeft met het vervangen van een defecte Wi-Fimodule. Dit draadloze netwerk is enkel in de personeelsruimte en in de computerwinkel beschikbaar voor tijdens de pauzes, of als de klanten ze nodig hebben. De Wi-Fi in de computerwinkel is enkel actief tijdens de openingsuren en de middagpauze.



# 2 Patrick, netwerkbeheerder bij Page

De informatie over het netwerk van Page haalde ik uit de uitleg die ik kreeg van Patrick, de netwerkbeheerder bij Page. Patrick staat in voor het onderhoud en de organisatie van het bedrijfsnetwerk en hij functioneert als ICT-er in de productieafdeling voor het geval er een probleem is met een computer. Ook al draait de server op een OS van negen jaar geleden, toch blijft Patrick mee met de tijd door zijn interesse in het vak en de nodige bijscholingen. Over de professionele contacten die Patrick heeft heb ik hem niet specifiek aangesproken, maar hij vermeldde tijdens onze gesprekken enkele voorvallen die hij gehad heeft met mede-netwerkbeheerders uit andere bedrijven omtrent nieuwe hard- en software, of het optimaal instellen van netwerken.

Patrick staat elke werkdag paraat om de werknemers van de productieafdeling te helpen met hun problemen. Hij zorgt er ook voor dat het netwerk vlotjes werkt zodat de werking van het bedrijf niet vertraagd wordt. Vanwege zijn lange dienst in het bedrijf is hij vertrouwd bij Luc Page en zit hij vaak op vergadering met de rest van het management. In het bedrijf is Patrick dus de helpdesk, al heeft Page geen online helpdesk waar men hun problemen kan melden zoals op onze GIP-site.

# 3 Toestellen en internetvoorziening

## 3.1 BYOD

Bij Page is het beleid over Bring Your Own Device erg eenvoudig: men mag zijn eigen toestel meenemen naar het werk zolang dit enkel voor professionele doeleinden gebruikt wordt. Aangezien er maar een beperkt aantal vaste lijnen zijn, is het handig voor bepaalde werknemers om via een mobiele telefoon bereikbaar te zijn. Sommige werknemers kiezen er ook voor om hun gegevens op de cloud te zetten, bv. met Office 365 of Google Drive, zodat ze hun documenten op eender welk toestel kunnen raadplegen. Werknemers hebben ook toegang tot software waarvoor het bedrijf licenties heeft afgesloten, zoals het boekhoudprogramma ISABEL en een standaard Officepakket. Page ontwikkelt zelf geen eigen software, maar mijn stagementor Jan heeft wel interesse om te leren programmeren en misschien een app'je te ontwikkelen voor het bedrijf.

## 3.2 Abonnementen

Bepaalde werknemers krijgen abonnementen van het bedrijf voor hun mobiele telefonie, zoals Bert, een technicus die in de computerwinkel werkt en vaak op stap gaat bij de klant zelf om de installatie van een product uit te voeren. Tijdens zijn werk kan het gebeuren dat hij naar zijn collega's in de computerwinkel moet bellen om te vragen naar benodigde informatie of om gegevens zoals garantie na te kijken dus is het een goeie zaak voor hem dat Page deze kosten dekt. Het abonnement van Bert geeft hem beschikking over genoeg belminuten en 4G om rond te komen tijdens zijn werk.

De ene keer dat ik mee ging met Bert om een internetconfiguratie op te zetten in een bed & breakfast hebben we naar elkaar gebeld omdat we elk aan de andere kant van het gebouw zaten, het was erg handig om toch met elkaar te kunnen spreken, ondanks de afstand.

Page Elektronica heeft zelf in de productieafdeling een internetabonnement van Telenet Business afgesloten voor hun zakelijke communicatie, terwijl Page Computers een eenvoudig particulier abonnement heeft bij de provider Proximus met een goeie downloadsnelheid en deftige Wi-Fi via het Proximus Internet & TV abonnement.

# 4 Web & media

## 4.1 Page.be, en je bent mee

De website van Page Computers (www.page.be) Is opgemaakt door de medewerkers van Page door middel van Joomla!, een CMS (Content Management System) om websites te ontwikkelen, een beetje zoals WordPress. De webshop van Page is, door hun partnering, gemaakt door Selexion. Dankzij deze partnering kan Page niet alleen makkelijk aan de beste producten raken, maar worden ze ook een verdeelpunt voor bestellingen gemaakt bij Selexion, wat zorgt voor meer volk in de winkel.

## 4.2 Hulp op afstand

Op de site kan men ook het programma dat de technische dienst gebruikt voor hulp op afstand, nl. TeamViewer, downloaden samen met een gedetailleerde gebruiksaanwijzing zodat iedereen het programma kan gebruiken. De website wordt geüpdatet door Björn, de verkoper van Page wanneer er bv. een nieuwe aanbieding is, of wanneer er nieuwe producten binnen zijn gekomen. Doordat men Joomla! gebruikt is het updaten van de site makkelijk en gebruiksvriendelijk.

Een online helpdesk zoals onze GIP-helpdesk heeft Page niet, maar tijdens de openingsuren van het bedrijf (van 9:30 tot 12:00 en van 13:30 tot 18:30, gesloten op maandagvoormiddag en zondag) is de computerwinkel telefonisch bereikbaar. Voor telefonische hulpsessies vraagt Page €10,00 voor 10 minuten, met een bijkomend tarief van €5,00 per bijkomende 5 minuten. Men mag ook natuurlijk zijn/haar toestel meenemen naar de winkel, zodat we ter plekke kunnen helpen of het toestel even bij ons houden voor onderhoud.

## 4.3 Sociale media

Via de Facebook-pagina van Page Computers (https://www.facebook.com/pagecomputers) vind je ook informatie over de computerwinkel en hun aanbiedingen. Op deze pagina heeft mentor Jan een foto gepubliceerd van de echograaf van het Jan Yperman ziekenhuis die we hersteld hebben (Fig. 3 & 4), een hoogtepuntje van mijn stage. Iemand heeft blijkbaar ook enkele foto's geplaatst van mezelf die op mijn laatste avond een virtual-reality bril uitprobeer.

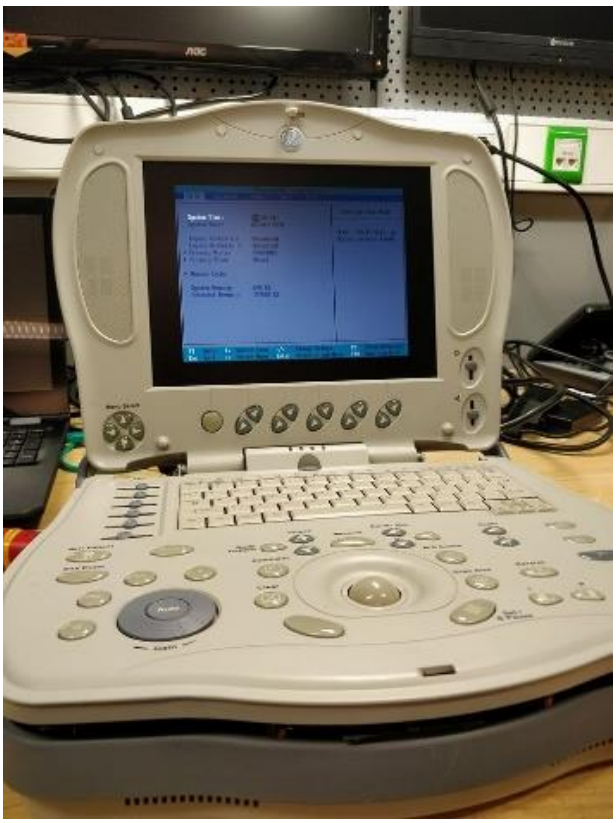



Fig. 3 & 4: De echograaf van het Jan Yperman Ziekenhuis

**Nawoord**

Al bij al heb ik veel geleerd tijdens mijn stage, vooral rond de vaste procedure die ze hebben om problemen te identificeren en zo op te lossen. Ik vind het jammer dat we slechts één keer stage hebben dit jaar, omdat ik ook graag eens zou stage gelopen hebben bij een softwarebedrijf.

# Eindwerk: Online Helpdesk

**Inleiding**
Als eindwerk kregen wij dit schooljaar de opdracht om een online helpdesk te ontwikkelen, voornamelijk in PhP. In dit deel van de bundel komt zowat alle code die bijdraagt tot het werken van de helpdesk. Om redundantie te vermijden komen bestanden die meermaals voorkomen in de mappenstructuur hier slechts één keer voor.

Qua thematiek en ontwerp heb ik mijn helpdesk vrij neutraal gehouden: ze is opgebouwd om op school gebruikt te worden, maar dit zou door enkele wijzigingen in het ticketformulier en de MySQL database veranderd kunnen worden om bij een bepaald bedrijf te passen.

Als kleurenpalet koos ik voor leigrijs en oker/goud op een gebroken witte achtergrond. Deze kleuren spraken me aan omdat ze neutraal en zacht zijn waardoor je geen storend gevoel krijgt tijdens het gebruiken van de helpdesk.

De helpdesk maakt gebruik van twee soorten gebruikers: nl. de gewone gebruikers (users) en administrators (admins).
Users krijgen toegang tot het melden van tickets, terwijl admins ze kunnen bewerken en verwijderen.
Terwijl iedereen zich kan registreren op deze helpdesk is het aanmaken van gebruikers met administratormachtigingen enkel mogelijk voor bestaande administratoren.

Bij het ontwikkelen en testen van deze helpdesk gebruikte ik twee gebruikers:
“Admin” als administrator, en “Oliver” als gebruiker. Beide profielen gebruiken het wachtwoord “Hfi1234”, een standaardwachtwoord die we gebruikten tijdens de lessen WEB & BNET.

# 1 Hoofdpagina's

Bij het surfen naar de website komen bezoekers standaard op de indexpagina terecht. Na het aandmelden komen users en admins op hun respectievelijke pagina's terecht.

## 1.1 Index.php

```php
<?php
// Code voor de indexpagina, gebruikt door niet-aangemelde gebruikers
error_reporting( E_ALL );
ini_set( "display_errors", 1);
include_once "models/Page_Data.class.php";
$pageData = new Page_Data();
// Via pageData nodige informatie (paginatitel, CSS & scripts) toevoegen
$pageData->title = "GIP Oliver Lanckriet - Helpdesk";
$pageData->addCSS("layout");
$pageData->addJS("backToTop");

//database connectie
$dbInfo = "mysql:host=localhost;dbname=gip";
$dbUser = "root";
$dbPassword = "";
$db = new PDO( $dbInfo, $dbUser, $dbPassword );
$db->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION );

//navigatie toevoegen
$pageData->content = include_once "views/navigation.php";

//Navigatie klik
$navigationIsClicked = isset( $_GET['page'] );
if ( $navigationIsClicked ) {
$controller = $_GET['page'];
} else {
$controller = "lijst";
}

$pageData->content .= include_once "controllers/$controller.php";
$page = include_once "views/page.php";
$page .= include_once"views/footer.php";
echo $page;
```

### 1.2 User.php

```php
<?php
// Code voor de standaard gebruikerspagina
// Niet toegankelijk voor onaangemelde gebruikers
// navigatie gebruikt bestanden uit de controllers/users/ map
error_reporting( E_ALL );
ini_set( "display_errors", 1 );
include_once "models/Page_Data.class.php";
$pageData = new Page_Data();
// Via pageData nodige informatie (paginatitel, CSS & scripts) toevoegen
$pageData->title = "GIP Oliver Lanckriet - Helpdesk";
$pageData->addCSS('layout');
$pageData->addJS("backToTop");
$pageData->addJS("popups");

//database connectie
$dbInfo = "mysql:host=localhost;dbname=gip";
$dbUser = "root";
$dbPassword = "";
$db = new PDO( $dbInfo, $dbUser, $dbPassword );
$db->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION );
include_once "models/Reg_User.class.php";
$user = new Reg_User();
//login-controller laden die het aanmeldformulier toont
$pageData->content = include_once "controllers/users/login.php";
//gebruikersmodule enkel tonen als de gebruiker aangemeld is
if( $user->isLoggedIn() ) {
	$pageData->content .= include "views/users/logout-form-html.php";
	$navigationIsClicked = isset( $_GET['page'] );
	if ($navigationIsClicked ) {
		$controller = $_GET['page'];
	} else {
		$pageData->content .=include "views/navigation.php";
		$controller = "lijst";
	}
	$pathToController = "controllers/users/$controller.php";
	$pageData->content .=include_once $pathToController;
}
$page = include_once "views/page.php";
$page .= include_once"views/footer.php";
echo $page;
```

### 1.3 Admin.php

```php
<?php
// Code voor de administratorpagina
// Enkel toegankelijk voor administrators
// Nagivatie gebruikt controllers uit de controllers/admin/ map
error_reporting( E_ALL );
ini_set( "display_errors", 1 );
include_once "models/Page_Data.class.php";
$pageData = new Page_Data();
// Via pageData nodige informatie (paginatitel, CSS & scripts) toevoegen
$pageData->title = "GIP Oliver Lanckriet - Helpdesk";
$pageData->addCSS('layout');
$pageData->addJS("backToTop");
$pageData->addJS("popups");

//database connectie
$dbInfo = "mysql:host=localhost;dbname=gip";
$dbUser = "root";
$dbPassword = "";
$db = new PDO( $dbInfo, $dbUser, $dbPassword );
$db->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION );
include_once "models/Admin_User.class.php";
$admin = new Admin_User();
//login-controller laden die het aanmeldformulier toont
$pageData->content = include_once "controllers/admin/adminlogin.php";
//adminmodule enkel tonen als de administrator aangemeld is
if( $admin->isLoggedIn() ) {
	$pageData->content .= include "views/admin/logout-form-html.php";
	$navigationIsClicked = isset( $_GET['page'] );
	if ($navigationIsClicked ) {
		$controller = $_GET['page'];
	} else {
		$pageData->content .=include "views/admin/navigation.php";
		$controller = "lijst";
	}
	$pathToController = "controllers/admin/$controller.php";
	$pageData->content .=include_once $pathToController;
}
$page = include_once "views/page.php";
$page .= include_once"views/admin/footer.php";
echo $page;
```

# 2 Page & PageData

Deze php-bestanden worden gebruikt als de “geraamtes” van de site. Page.php is een standaard HTML-pagina die gebruik maakt van Page_Data.class.php die op zijn beurt informatie krijgt van de hoofdpagina’s.

## 2.1 Page.php

```
<?php
// Blanco html-pagina gebruikt door Page_Data.class.php
return "
<!DOCTYPE html>
<html>
<head>
<!-- css & javascript -->
$pageData->css
$pageData->link
<!-- favicon by tupadesigns -->
<link rel='icon' href='img/olibeer.png' type='image/x-icon'/>
<link rel='shortcut icon' href='img/olibeer.png' type='image/x-icon'/>
<!-- titel -->
<title>$pageData->title</title>
</head>
<body>
<!-- alles in de body -->
$pageData->content
</body>
</html>";
```

## 2.2 Page_Data.class.php

```
<?php
// Data die op de webpagina moet komen
class Page_Data {
	public $title = "";
	public $content = "";
	public $css = "";
	public $link = "";
	public $embeddedStyle = "";

	public function addCSS($href) {
		$this->css .= "<link href='css/$href.css' rel='stylesheet' />";
	}

	public function addJS($href) {
		$this->link .= "<script src='js/$href.js'></script>";
	}
}
```

# 3 Views: Navigatiebalk & footer

Hoewel de footer op elke pagina dezelfde blijft, komt er na het aanmelden van de gebruiker extra informatie in de navigatiebalk. Bij onaangemelde gebruikers kan men enkel de indexpagina bekijken en aanmelden als user/admin. Users krijgen toegang tot het ticketsysteem, en admins krijgen toegang tot het beheer.

### 3.1 Navigation.php & logout-form-html.php (users & admins)

```
<?php
// Navigatiebalk bovenaan de webpagina
// Home verwijst naar indexpagina met lijst
// Aanmelden verwijst naar de aanmeldpagina (users / admins)
return "
<nav>
      <div class='nav'>
        <a href='index.php?'><img src='img/home.png' style='width:20px'
/></a>
            <a href='user.php?page=lijst'>Aanmelden</a>
            <a href='admin.php?page=lijst'>Administrators</a>
      </div>
</nav>
";
```

Voor users:

```
<?php
// Wordt weergegeven als navigatiebalk als de gebruiker ingelogd is
return "
<nav>
<div class='nav'>
      <form id='logout' method='post' action='index.php?page=logout'>
      <a href='user.php?page=lijst'><img src='img/home.png'
style='width:20px' /></a>
            <a href='user.php?page=ticket'>Probleem melden</a>
            <a href='user.php?page=management'>Probleem beheren</a>
            <input type='submit' id='logoutbtn' value='logout' name='logout'
/>
      </form>
</div>
</nav>
";
```

Voor admins:

```
<?php
// Wordt weergegeven als navigatiebalk als de gebruiker ingelogd is
return "
<nav>
      <div class='nav'>
      <form id='logout' method='post' action='index.php?page=logout'>
     <a href='admin.php?page=lijst'><img src='img/home.png'
style='width:20px' /></a>
            <a href='admin.php?page=admin-ticket'>Probleem melden</a>
            <a href='admin.php?page=management'>Problemen beheren</a>
            <a href='admin.php?page=users'>Nieuwe admin maken</a>

            <input type='submit' id='logoutbtn' value='logout' name='logout'
/>
      </form>
      </div>
</nav>
";
```

### 3.2 Footer.php

```
<?php
// De footer op elke pagina
return "
<footer>
      <center>No copyright - Oliver Lanckriet 2018</center>
</footer>
";
```

# 4 Views: lijst, tickets, aanmeldformulieren & registratie

In de views-map staan er drie bestanden centraal: list-entries-html.php, form.php & login-form-html.php. Deze bestanden worden het vaakst gebruik bij het dagelijks gebruik van de helpdesk.

### 4.1 List-entries-html.php

```
<?php
//Kijk of de lijst inhoud bevat
$entriesFound = isset( $entries );
if ( $entriesFound === false ) {
	trigger_error( 'views/list-entries-html.php needs $entries' );
}

//<ul> element maken
$entriesHTML = "";
$entriesHTML .= "<ul id='list' class='gip-entries'>";

//loop door alle records van de database
//onthoud elke rij tijdelijk als $entry
//$entry is een StdClass object met entry_id, title en intro
while ( $entry = $entries->fetchObject() ) {
	//<li> maken voor elke record
	$entriesHTML .= "
<li><h2>$entry->titel</h2><br />
<b>Beschrijving:</b><br /> $entry->tekst<br />
<b>Lokaal:</b> $entry->lokaal<br />
<b>Toestel:</b> $entry->pcnummer<br />
<b>Gemeld op:</b> $entry->datum_aangemaakt <b>door</b> $entry->melder</li>
<br />
";
}

//<ul> beëindigen
$entriesHTML .= "</ul>";
// back-to-topknop
$entriesHTML .= "<button onclick='backToTop()' id='button' title='Back to
top'>Top</button>";
return $entriesHTML;
```

### 4.2 Form.php

```
<?php
// HTML-Code van het formulier, bij tekstvakken placeholders aangeduid ter
verduidelijking, ook gemarkeerd als "required"
// -> Kan niet blanco gelaten worden (beschrijving optioneel indien
zelfverklarende titel)
return "
<div class='form'>
<form method='post' action='user.php?page=ticket'>
<input type='text' id='titel' name='title' placeholder='Titel...' required>
<input type='text' id='lokaalnr' name='lokaal' placeholder='Lokaal...'
required>
<input type='text' id='pcnummer' name='pcnummer' placeholder='PC-Nummer...'
required>
<textarea name='entry' placeholder='Beschrijf uw probleem...'></textarea>
<input type='text' id='melder' name='melder' placeholder='Gemeld door...'
required>
<input type='submit' onclick='ticket()' name='action' value='opslaan'>
</form>
</div>
";
```

### 4.3 Login-form-html.php

```
<?php
//Loginformulier voor gebruikers
return "
<div class='login'>
      <form method='post' action='user.php?page=lijst'>
      <p>Meld u aan om een probleem te melden</p>
      <input type='text' name='email' placeholder='Geef je naam in...'
required /><br />
      <input type='password' name='password' placeholder='Geef je wachtwoord
in...' required /><br />
      <input id='loginbtn' type='submit' value='login' name='log-in' />
      <p>Nog geen profiel? <a href='index.php?page=register'>registreer
nu!</a></p>
</form>
</div>
";
```

### 4.4 Loginerror.php

```
<?php
// error
return "
<p>Verkeerde gebruikersnaam/wachtwoord, probeer het opnieuw.</p>";
?>
```

### 4.5 New-user-form-html.php

```
<?php
// code voor het registratieformulier van gebruikers
if (isset ( $userFormMessage ) === false) {
      $userFormMessage = "";
}
return "
<div class='login'>
<form method='post' action='index.php?page=register'>
<p>Registreer u!</p>
<input type='text' name='email' placeholder='Geef uw naam in...'
required/><br />

<input type='password' name='password' placeholder='Geef uw wachtwoord in...'
required><br />

<input type='submit' id='loginbtn' value='registreren' name='new-user'/>
<p id='admin-form-message'>$userFormMessage</p>
</form>
";
```

### 4.6 New-admin-form-html.php

```
<?php
// Code voor de registratie van administrators
if (isset ( $adminFormMessage ) === false) {
      $adminFormMessage = "";
}
return "
<div class='login'>
<form method='post' action='admin.php?page=users'>
<p> Registreer een nieuwe admin</p>
<input type='text' name='email' placeholder='Geef uw naam in...'
required/><br />

<input type='password' name='password' placeholder='Geef uw wachtwoord in...'
required><br />

<input type='submit' id='loginbtn' value='registreren' name='new-admin'/>
<p id='admin-form-message'>$adminFormMessage</p>
</form>
";
```

# 5 Views: Management (admins)

Administratorgebruikers kunnen gebruik maken van de management & editor-pagina's om de helpdesk te beheren.

### 5.1 Editor-html.php

```
<?php
//Code voor de editor om tickets te wijzigen

//check if required data is available
$entryDataFound = isset( $entryData );
if( $entryDataFound === false ){
      //default values for an empty editor
      $entryData = new StdClass();
      $entryData->entry_id = 1;
      $entryData->titel = "";
      $entryData->lokaal= "";
      $entryData->pcnummer = "";
      $entryData->tekst = "";
      $entryData->message = "";
}

return "
<div class='form'>
      <form method='post' action='admin.php?page=editor'>
       <input type='hidden' name='id' value='$entryData->entry_id' />
<input type='text' id='titel' name='title' value='$entryData->titel'
required>

<input type='text' id='lokaalnr' name='lokaal' value='$entryData->lokaal'
required>

<input type='text' id='pcnummer' name='pcnummer' value='$entryData->pcnummer'
required>

<textarea name='entry'>$entryData->tekst</textarea>



<input type='submit' onclick='ticketUpdate()' name='action' value='opslaan'>
<p id='editor-message'>$entryData->message</p>
<input type='submit' onclick='ticketDelete()' name='action'
value='verwijderen'>
       </form>
</div>
";
```

### 5.2 Management-html.php

```php
<?php
//Kijk of de lijst inhoud bevat
$entriesFound = isset( $entries );
if ( $entriesFound === false ) {
	trigger_error( 'views/management-html.php needs $entries' );
}

//<ul> element maken
$entriesHTML = "<ul class='gip-entries'>";
$entriesHTML .= "<h2>Klik op een titel om de entry te bewerken</h2>";

//loop door alle records van de database
//onthoud elke rij tijdelijk als $entry
//$entry is een StdClass object met entry_id, title en intro
while ( $entry = $entries->fetchObject() ) {
	//<li> maken voor elke record
	$entriesHTML .= "
<li><a href='admin.php?page=editor&amp;id=$entry->entry_id' style='color:
black;'><h3>$entry->titel</h3></a></li>
";
}

//<ul> beëindigen
$entriesHTML .= "</ul>";
// back-to-topknop
$entriesHTML .= "<button onclick='backToTop()' id='button' title='Back to
top'>Top</button>";
return $entriesHTML;
```

# 6 Controllers: Lijst.php & ticket.php

Deze controllers worden gebruikt om de lijst van alle tickets en het ticketformulier weer te geven.

## 6.1 Lijst.php

```php
<?php
//Code van het maken van de lijst tickets vóór het inloggen
include_once "models/Lijst.class.php";
$entryTable = new Lijst($db);
$entries = $entryTable->getAllRecords();
$Output = include_once "views/list-entries-html.php";
return $Output;
```

## 6.2 Ticket.php

```php
<?php
// code voor het opgeven van tickets
include_once "models/Helpdesk_Entry_Table.class.php";
$entryTable = new Helpdesk_Entry_Table( $db );
// Data van het ticket opslaan
$editorSubmitted = isset( $_POST['action'] );
if ( $editorSubmitted ) {
	$buttonClicked = $_POST['action'];
	$insertNewEntry = ( $buttonClicked === 'opslaan' );
	if ( $insertNewEntry ) {
		// Data van het ticket
		$title = $_POST['title'];
		$lokaal = $_POST['lokaal'];
		$entry = $_POST['entry'];
		$pcnummer = $_POST['pcnummer'];
		$melder = $_POST['melder'];
		$entryTable->saveEntry( $title, $lokaal, $entry, $pcnummer,
$melder );
	}
}

$editorOutput = include_once "views/users/form.php";
return $editorOutput;
```

# 7 Controllers: Login.php, logout.php & register.php

Deze controllers worden gebruikt om in- en uit te loggen en gewone gebruikers te registreren.

## 7.1 Login.php

```php
<?php
//Code voor het aanmelden van een gebruiker
include_once "models/User_Table.class.php";
$loginFormSubmitted = isset( $_POST['log-in'] );
if( $loginFormSubmitted ) {

	$email = $_POST['email'];
	$password = $_POST['password'];
	//create an object for communicating with the database table
	$userTable = new User_Table( $db );
	try {
		//try to login user
		$userTable->checkCredentials( $email, $password );
		$user->login();
	} catch ( Exception $e ) {

	}
}
$loggingOut = isset ( $_POST['logout'] );
if ( $loggingOut ){
	$user->logout();
}
if ( $user->isLoggedIn() ) {
	$view = include_once "views/users/logout-form-html.php";
} else {
	$view = include_once "views/navigation.php";
	$view .= include_once "views/users/login-form-html.php";
	if( $loginFormSubmitted ) {
		$view .= include_once "views/admin/loginerror.php";
	}
}
return $view;
```

## 7.2 Logout.php

```php
<?php
//Code voor het uitloggen
session_start();
session_destroy();
header("Location: index.php"); //Herleiden naar homepage
?>
```

### 7.3 Register.php

```php
<?php
//Code om een gebruiker te registreren
include_once "models/User_Table.class.php";
//is form submitted?
$createNewUser = isset( $_POST['new-user'] );
//if it is...
if( $createNewUser ) {
	//grab form input
	$newUser = $_POST['email'];
	$newPassword = $_POST['password'];
	$userTable = new User_Table($db);
	try {
		//try to create a new user
		$userTable->create( $newUser, $newPassword );
		//tell user how it went
		$userFormMessage = "Nieuwe gebruiker $newEmail aangemaakt!";
	} catch ( Exception $e ) {
		//if operation failed, tell user what went wrong
		$userFormMessage = $e->getMessage();
	}
}
$newAdminForm = include_once "views/users/new-user-form-html.php";
return $newAdminForm;
```

# 8 Controllers: editor.php, management.php & adminregister.php

Deze controllers worden gebruikt om bestaande tickets aan te passen, de managementmogelijkheden van de administrator en het registreren van de administrator.

## 8.1 Editor.php

```php
<?php
//code voor de editor waar administrators meldingen kunnen aanpassen /
verwijderen
include_once "models/Helpdesk_Entry_Table.class.php";
$entryTable = new Helpdesk_Entry_Table( $db );
//was editor form submitted?
$editorSubmitted = isset( $_POST['action'] );
if ( $editorSubmitted ) {
	$buttonClicked = $_POST['action'];
	$save = ( $buttonClicked === 'opslaan' );
	$id = $_POST['id'];
	//was "save" button clicked
	$insertNewEntry = ( $save and $id === '0' );
	$deleteEntry = ( $buttonClicked === 'verwijderen' );
	$updateEntry = ( $save and $insertNewEntry === false );
	$title = $_POST['title'];
	$lokaal = $_POST['lokaal'];
	$pcnummer = $_POST['pcnummer'];
	$entry = $_POST['entry'];
	$id = $_POST['id'];
	if ( $insertNewEntry ) {
		//get title and entry data from editor form
		$title = $_POST['title'];
		$lokaal = $_POST['lokaal'];
		$pcnummer = $_POST['pcnummer'];
		$entry = $_POST['tekst'];
		//save the new entry
		$savedEntryId = $entryTable->saveEntry( $title, $lokaal,
$pcnummer, $entry);
	} else if ($updateEntry) {
		$entryTable->updateEntry($id, $title, $lokaal, $pcnummer,
$entry);
		$savedEntryId=$id;
	} else if( $deleteEntry ) {
		$entryTable->deleteEntry ($id);
		header("Location: admin.php?page=management");
	}
}
$entryRequested = isset( $_GET['id'] );
//create a new if-statement
if ( $entryRequested ) {
	$id = $_GET['id'];
	//load model of existing entry
	$entryData = $entryTable->getEntry( $id );
	$entryData->entry_id = $id;
	$entryData->message = "";
}
```

```
$entrySaved = isset( $savedEntryId );
if ( $entrySaved ) {
	$entryData = $entryTable->getEntry( $savedEntryId );
	//toon een bevestigingsboodschap
	$entryData->message = "Ticket upgedatet!";
}
$editorOutput = include_once "views/admin/editor-html.php";
return $editorOutput;
```

### 8.2 Management.php

```
<?php
// Code voor het maken van de lijst titeltjes van de managementpagina
include_once "models/Lijst.class.php";
$entryTable = new Lijst($db);
$entries = $entryTable->getAllRecords();
$Output = include_once "views/admin/management-html.php";
return $Output;

return $editorOutput;
```

### 8.3 Adminregister.php

```
<?php
//Code voor het registreren van administrators
//Kan enkel gedaan worden door een bestaande admin
include_once "models/Admin_Table.class.php";
//is form submitted?
$createNewAdmin = isset( $_POST['new-admin'] );
//if it is...
if( $createNewAdmin ) {
	//grab form input
	$newEmail = $_POST['email'];
	$newPassword = $_POST['password'];
	$adminTable = new Admin_Table($db);
	try {
		//try to create a new admin user
		$adminTable->create( $newEmail, $newPassword );
		//tell user how it went
		$adminFormMessage = "Nieuwe gebruiker $newEmail aangemaakt!";
	} catch ( Exception $e ) {
		//if operation failed, tell user what went wrong
		$adminFormMessage = $e->getMessage();
	}
}
$newAdminForm = include_once "views/admin/new-admin-form-html.php";
return $newAdminForm;
```

# 9 Models: Lijst.class.php & Helpdesk_Entry_Table.class.php

Deze models worden gebruikt om de lijst tickets weer te geven en om tickets op te slaan / wijzigen in de editor.

## 9.1 Lijst.class.php

```php
<?php
class Lijst
{
	private $db;

	public function __construct($dbConnection)
	{
		$this->db = $dbConnection;
	}

	// Alle tickets weergeven
	public function getAllRecords()
	{
		$sql = "SELECT entry_id, lokaal, titel, pcnummer, tekst,
datum_aangemaakt, melder FROM gip_entry order by entry_id desc";
		$statement = $this->db->prepare($sql);
		try {
			$statement->execute();
		} catch (Exception $e) {
			$exceptionMessage = "<p>You tried to run this sql: $sql <p>
			<p>Exception: $e</p>";
			trigger_error($exceptionMessage);
		}
		return $statement;
	}
}
```

### 9.2 Helpdesk_Entry_Table.class.php

```
<?php
//Klasse voor het toevoegen van een blog entry, aangepast van document uit de
les
class Helpdesk_Entry_Table {
	private $db;

	//Constructor
	public function __construct ( $db ) {
		$this->db = $db;
	}

	//Ticket opslaan in de MySQL databank
	public function saveEntry ( $title, $lokaal, $entry, $pcnummer,
$melder ) {
		$entrySQL = "INSERT INTO gip_entry ( titel, lokaal, tekst,
pcnummer, melder )
		VALUES ( '$title', '$lokaal', '$entry', '$pcnummer',
'$melder' )";
		$entryStatement = $this->db->prepare( $entrySQL );
		$formData = array($title, $lokaal, $entry, $pcnummer);
		try{
			$entryStatement->execute($formData);
		} catch (Exception $e){
			$msg = "<p>You tried to run this sql: $entrySQL<p>
			<p>Exception: $e</p>";
			trigger_error($msg);
		}
	}
	public function deleteEntry ( $id ) {
		$sql = "DELETE FROM gip_entry WHERE entry_id = ?";
		$statement = $this->db->prepare( $sql );
		$data = array( $id );
		try{
			$statement->execute( $data );
		} catch (Exception $e) {
			$exceptionMessage = "<p>You tried to run this sql: $sql <p>
					<p>Exception: $e</p>";
			trigger_error($exceptionMessage );
		}
	}
	public function updateEntry ( $id, $title, $lokaal,  $pcnummer, $entry)
{
		$sql = "UPDATE gip_entry
				SET titel = ?,
				lokaal = ?,
				pcnummer = ?,
				tekst = ?
				WHERE entry_id = ?";
		$data = array( $title, $lokaal, $pcnummer, $entry, $id);
		$statement = $this->db->prepare( $sql );
		try{
			$statement->execute( $data );
		} catch (Exception $e) {
			$exceptionMessage = "<p>You tried to run this sql: $sql
<p><p>Exception: $e</p>";
			trigger_error($exceptionMessage );
```

```
            }
            return $statement;
      }
      //Lijst van alle bijgehouden tickets tonen
      public function getAllEntries () {
            $sql = "SELECT entry_id, titel, lokaal, tekst, datum_aangemaakt,
melder
                        FROM gip_entry order by entry_id desc";
            $statement = $this->db->prepare( $sql );
            try {
                  $statement->execute();
            } catch ( Exception $e ) {
                  $exceptionMessage = "<p>You tried to run this sql: $sql <p>
<p>Exception: $e</p>";
                  trigger_error($exceptionMessage);
            }
            return $statement;
      }
      // Eén ticket volledig tonen
      public function getEntry( $id ) {
            $sql = "SELECT entry_id, titel, lokaal, pcnummer, tekst,
datum_aangemaakt
            FROM gip_entry
            WHERE entry_id = ?";
            $statement = $this->db->prepare( $sql );
            $data = array( $id );
            try{
                  $statement->execute( $data );
            } catch (Exception $e) {
                  $exceptionMessage = "<p>You tried to run this sql: $sql <p>
                  <p>Exception: $e</p>";
                  trigger_error($exceptionMessage );
            }
            $model = $statement->fetchObject();
            return $model;
      }

}
```

# 10 Models: Reg_User.class.php & User_Table.class.php

Deze models worden gebruikt voor het aan- & afmelden van gebruikers, en het registreren van de gebruikers. Gebruikers worden in de MySQL database opgeslagen in de “users” tabel.

## 10.1 Reg_User.class.php

```php
<?php
// Code voor het in- / uitloggen van gebruikers
class Reg_User {
	//declare a new method, a constructor
	public function __construct(){
		//start a session
		session_start();
	}
	//edit existing method
	public function isLoggedIn(){
		$sessionIsSet = isset( $_SESSION['logged_in'] );
		if ( $sessionIsSet ) {
			$out = $_SESSION['logged_in'];
		} else {
			$out = false;
		}
		return $out;
	}
	//edit existing method
	public function login () {
		//set session variable ['logged_in'] to true
		$_SESSION['logged_in'] = true;
	}
	//edit existing method
	public function logout () {
		//set session variable ['logged_in'] to false
		$_SESSION['logged_in'] = false;
		session_destroy();
	}
}
```

### 10.2 User_Table.class.php

```php
<?php
//Code voor de users tabel

class User_Table {
	private $db;

	public function __construct ( $db ) {
		$this->db = $db;
	}
	public function makeStatement ( $sql, $data ){
		$statement = $this->db->prepare( $sql );
		try {
			$statement->execute( $data );
		} catch (Exception $e) {
			$exceptionMessage = "<p>You tried to run this sql: $sql
<p><p>Exception: $e</p>";
			trigger_error($exceptionMessage);
		}
		return $statement;
	}
	public function create ( $email, $password ) {
		//check if e-mail is available
		$this->checkName( $email );
		//encrypt password with MD5
		$sql = "INSERT INTO users ( email, password )
				VALUES( ?, MD5(?) )";
		$data= array( $email, $password );
		$this->makeStatement( $sql, $data );
	}
	private function checkName ($email) {
		$sql = "SELECT email FROM users WHERE email = ?";
		$data = array( $email );
		$this->makeStatement( $sql, $data );
		$statement = $this->makeStatement( $sql, $data );
		//if a user with that e-mail is found in database
		if ( $statement->rowCount() === 1 ) {
			//throw an exception > do NOT create new admin user
			$e = new Exception("Error: '$email' is reeds in gebruik!");
			throw $e;
	}}

	public function checkCredentials ( $email, $password ){
		$sql = "SELECT email FROM users
				WHERE email = ? AND password = MD5(?)";
		$data = array($email, $password);
		$statement = $this->makeStatement( $sql, $data );
		if ( $statement->rowCount() === 1 ) {
			$out = true;
		} else {
			$loginProblem = new Exception( "login mislukt!" );
			throw $loginProblem;
		}
		return $out;}}
```

# 11 Admin_User.class.php & Admin_Table_Class.php

Deze models vullen de zelfde functie in als de models uit deel 10 maar dan voor de administrators. Nieuwe admins worden in de “admin” tabel gezet.

## 11.1 Admin_User.class.php

```php
<?php
//Code voor het in- / uitloggen van administrators
class Admin_User {
	//declare a new method, a constructor
	public function __construct(){
		//start a session
		session_start();
	}
	//edit existing method
	public function isLoggedIn(){
		$sessionIsSet = isset( $_SESSION['logged_in'] );
		if ( $sessionIsSet ) {
			$out = $_SESSION['logged_in'];
		} else {
			$out = false;
		}
		return $out;
	}
	//edit existing method
	public function login () {
		//set session variable ['logged_in'] to true
		$_SESSION['logged_in'] = true;
	}
	//edit existing method
	public function logout () {
		//set session variable ['logged_in'] to false
		$_SESSION['logged_in'] = false;
		session_destroy();
	}
}
```

### 11.2 Admin_Table.class.php

```php
<?php
//Code voor de admin tabel

class Admin_Table {
	private $db;

	public function __construct ( $db ) {
		$this->db = $db;
	}
	public function makeStatement ( $sql, $data ){
		$statement = $this->db->prepare( $sql );
		try {
			$statement->execute( $data );
		} catch (Exception $e) {
			$exceptionMessage = "<p>You tried to run this sql: $sql
<p><p>Exception: $e</p>";
			trigger_error($exceptionMessage);
		}
		return $statement;
	}

	// Maken van een nieuwe admin
	public function create ( $email, $password ) {
		//check if e-mail is available
		$this->checkEmail( $email );
		//encrypt password with MD5
		$sql = "INSERT INTO admin ( email, password )
				VALUES( ?, MD5(?) )";
		$data= array( $email, $password );
		$this->makeStatement( $sql, $data );
	}
	private function checkEmail ($email) {
		$sql = "SELECT email FROM admin WHERE email = ?";
		$data = array( $email );
		$this->makeStatement( $sql, $data );
		$statement = $this->makeStatement( $sql, $data );
		//if a user with that e-mail is found in database
		if ( $statement->rowCount() === 1 ) {
			//throw an exception > do NOT create new admin user
			$e = new Exception("Error: '$email' is reeds in gebruik!");
			throw $e;
		}
	}
	//partial code for models/admin/Admin_Table.class.php
	//declare new method in Admin_Table class
	public function checkCredentials ( $email, $password ){
		$sql = "SELECT email FROM admin
				WHERE email = ? AND password = MD5(?)";
		$data = array($email, $password);
		$statement = $this->makeStatement( $sql, $data );
		if ( $statement->rowCount() === 1 ) {
			$out = true;
		} else {
			$loginProblem = new Exception( "login mislukt!" );
			throw $loginProblem;
		}
```

```
            return $out;
        }
}
```

# 12 JavaScript

De helpdesk maakt gebruik van twee scripts: één in de lijstpagina, en een tweede bij het ticketformulier & de editor.

### 12.1 Back-To-Top-Knop (backToTop.js)

```
// Wanneer de gebruiker +20px naar beneden scrollt wordt de knop zichtbaar
window.onscroll = function() {scrollFunction()};

function scrollFunction() {
    if (document.body.scrollTop > 20 || document.documentElement.scrollTop >
20) {
        document.getElementById('button').style.display = 'block';
    } else {
        document.getElementById('button').style.display = 'none';
    }
}

// Wanneer de gebruiker op de knop klikt terug naar boven
function backToTop() {
    document.body.scrollTop = 0; // Safari
    document.documentElement.scrollTop = 0; // Chrome, Firefox, IE, Edge,
Opera...
}
```

Deze knop wordt zichtbaar nadat de gebruiker naar beneden scrollt op de lijstpagina om zo vlugger terug naar boven te geraken. Ik heb deze knop toegevoegd omdat ik ze al op meerdere sites heb gezien en altijd een handige functie vond.

### 12.2 Alerts (popups.js)

```
function ticket() {
    alert("Uw ticket werd succesvol geplaatst!");
}

function ticketUpdate() {
    alert("Uw ticket werd succesvol geüpdate!");
}
function ticketDelete() {
    alert("Uw ticket werd succesvol verwijderd!");
   }
```

Deze JavaScript-functies worden uitgevoerd in de editor bij het maken, updaten en verwijderen van tickets om wat meer feedback te geven aan de gebruiker zodat men zeker is dat het systeem zoals bedoeld werkt.

# 13 MySQL Database

De database van de site werkt met MySQL en omvat 3 tabellen: admin, gip_entry en users.

Admin:

| Column Name | Datatype | PK | NN | UQ | B | UN | ZF | AI | G | Default/Expression |
|---|---|---|---|---|---|---|---|---|---|---|
| admin_id | INT(11) | ☑ | ☑ | ☐ | ☐ | ☐ | ☐ | ☑ | ☐ | |
| email | TEXT | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | NULL |
| password | VARCHAR(32) | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | NULL |
| | | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | |

Gip_entry:

| Column Name | Datatype | PK | NN | UQ | B | UN | ZF | AI | G | Default/Expression |
|---|---|---|---|---|---|---|---|---|---|---|
| entry_id | INT(11) | ☑ | ☑ | ☐ | ☐ | ☐ | ☐ | ☑ | ☐ | |
| titel | VARCHAR(150) | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | NULL |
| lokaal | TEXT | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | NULL |
| pcnummer | TEXT | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | NULL |
| tekst | TEXT | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | NULL |
| datum_aangemaakt | TIMESTAMP | ☐ | ☑ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | CURRENT_TIMESTAMP |
| melder | VARCHAR(45) | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | NULL |
| | | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | |

Users:

| Column Name | Datatype | PK | NN | UQ | B | UN | ZF | AI | G | Default/Expression |
|---|---|---|---|---|---|---|---|---|---|---|
| id | INT(11) | ☑ | ☑ | ☐ | ☐ | ☐ | ☐ | ☑ | ☐ | |
| email | VARCHAR(45) | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | NULL |
| password | VARCHAR(45) | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | NULL |
| | | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | |

**Nawoord**

Nogmaals wil ik iedereen uit het voorwoord bedanken voor de steun bij het maken van deze GIP. Het was een hele uitdaging waarbij ik veel van mijn kennen en kunnen heb moeten toepassen.

Mijn stage bij Page Computers was zeker ook één van de hoogtepunten van dit schooljaar, al vond ik het wat jammer dat we slechts één keer stage mochten lopen: ik had immers graag nog eens stage gelopen bij een software- / webdesignbedrijf.

De stage en de GIP hebben me samen meer uitzicht gegeven naar de toekomst toe: graag zou ik willen verder studeren in een softwaregerichte richting zoals Software Development bij Howest.

**Bronnen**

De informatie die ik gebruikte bij het ontwikkelen van de helpdesk haalde ik vooral uit de lessen en cursussen van dhr. Staelens en dhr. De Wandel. Mijn klasgenoten hebben me ook in de goede richting geholpen. Aanvullende informatie haalde ik van het internet bij sites zoals w3schools.com.