

Werkprocessen – CMS – Pjotr Wisse – Floris van der Waals

In dit document staan alle werkprocessen die Floris en Pjotr hebben gemaakt tijdens het CMS project van 18-01-2021 tot 01-03-2021. Het eerste gedeelte zal bestaan uit de werkprocessen verricht door Floris, als laatste zullen de werkprocessen van Pjotr worden weergegeven. Dit document is gemaakt op 26-02-2021.

Bewijsmateriaal werkprocessen Floris

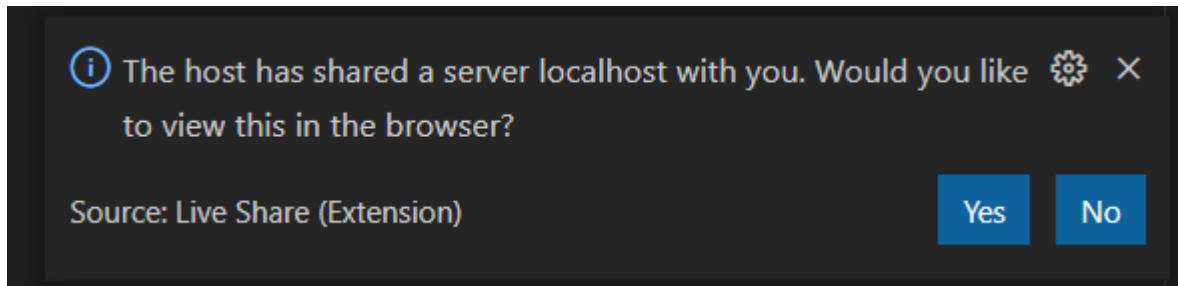
1. Stelt de opdracht vast: Voor dit werkproces heb ik het klanten gesprek gehouden. Zie hieronder de vragen die opgesteld zijn voor het klantengesprek met een korte omschrijving van het antwoord van de klant:
Vragen bij het klantengesprek CMS Portfoliogesprek
 - Welke stijl moet de portfoliowebsite krijgen?
Modern en gelicht(vormgeving student)
 - **(technische oefjes wanneer applicatie ontwerper is).**
 - **Een student voor applicatieontwerper.**
 - Kan het naast een student ook voor andere mensen bruikbaar zijn?
Alleen voor de student.
 - Hoe moet de website er visueel uit komen te zien(navigatie)?
Homepage, about me, CV en projectenpagina, Via menubalk, overal naartoe scrollen.
 - Wat moet er op de welkom pagina komen?
Meteen kunnen zien wiens portfolio het is en welke opleiding hij doet.
 - Wilt u nog extra onderdelen op de website dan in de omschrijving staat?
Link naar project is een vereiste.
 - Wilt u ook dat de extra projecten verwijderd kunnen worden omdat in de opdrachtomschrijving alleen toevoegen en aanpassen staat?
Alle functionaliteiten en ook verwijderen.
 - Wanneer wilt u het project tot uw beschikking hebben?
Na 6 weken tijd moet het product ingeleverd worden.

Daarbij hebben we nog gesproken over het principe van een navigatiebar en een scrollpagina. We hebben hiervoor gekozen om beide te implementeren.

We hebben niet met iets te maken gehad dat er onmogelijkheden waren voor dit project. We hadden samen nagedacht over het dit allemaal realistisch is in de tijd die we ervoor hebben.

2. Test het ontwikkelde product

De applicatie heb ik getest door het te bekijken in de browser.



Source: Live Share (Extension)

No

Toen ik langs de pagina's ging zag ik dat de footer nog niet netjes stond dus die hebben we dan aangepast. Dit had te maken met schermgrootte.



Morbi finibus nihil iusto, vel gravida lectus elementum ullamcorper. Quisque dapibus sollicitudin lincidunt. Nam rutrum sem sed arcu tempus sodales. Suspendisse ullamcorper eget mauris. Quis lobortis. Donec condimentum aliquam ipsum, quis accumsan massa placerat in. Suspendisse libero nulla, accumsan sed quam eget, maximus laculis erat.

Copyright © 2021 Niels Segaert

Op bootstrap heb ik dan gekeken en heb ik een fix hiervoor gevonden maar dan ziet het bij pjotr er weer niet goed uit. We hebben dus uiteindelijk besloten om het te presenteren op mijn computerscherm.

Voor de contact pagina zag ik dat de email functie nog niet helemaal werkt. Buiten deze dingen heb ik verder niks van fouten ontdekt.


Name

Email Address


Phone Number

Subject


Message



Heer Broekstraat 48




+31-06958737273



Segaar86@Gmail.com

Sorry Peter, it seems that my mail server is not responding.
Please try again later!



Send

Email Address

Phone Number

Subject

Message



Heer Broekstraat 48



+31-06958737273



Segaar86@Gmail.com

Sorry Peter, it seems that my mail server is not responding.
Please try again later!

Send

Fouten	Oplossingen
Email functie	Wat in de code aanpassen

Email functie

Oplossingen

Wat in de code aanpassen

3. Onderhoudt de applicatie

Tijdens het testprocedure heb ik gekeken of er aanpassingen moeten zijn voor de applicatie. Dit waren een aantal CSS punten, maar verder kwa functionaliteiten waren er geen punten waarvan wij dachten dat er iets nodig veranderd moest worden. Die punten heb ik dan aan de orde gebracht en die hebben we dan vervolgens aangepast.

Voor dit project hebben we gebruik gemaakt van JQuery libraries en het framework van Bootstrap. Hiervoor hebben we de nieuwste versie gebruikt en mocht er een nieuwe versie uitkomen dan zullen wij aanpassen naar die versie om het up-tot-date te houden.

4. Realiseert onderdelen van een product. Op het moment dat we het project willen gaan opleveren gaan we voor de oplevering bespreken dat we een aantal werkprocessen willen gaan aftekenen en die zo spoedig mogelijk naar de klant zullen opsturen zodat hij al bewust is dat dat eraan komt.

Zie hieronder delen van stukken script die we gemaakt hebben.

```
1  <?php
2  session_start();
3  require('database.php');
4  if (!isset($_SESSION["loggedin"])) {
5      header("Location: ../index.php");
6      exit();
7  }
8
9  // Insert into DATABASE
10 if(isset($_POST["title"], $_POST["text"], $_POST["subtext"], $_POST['Huidige_Afbeelding'])) {
11
12     $Huidig = $_POST['Huidige_Afbeelding'];
13     $Afbeelding = $_FILES['image'];
14     $Tijdelijk = $Afbeelding['tmp_name'];
15     $Afbeeldingnaam = $Afbeelding['name'];
16     $type = $Afbeelding['type'];
17     $map = "uploads/";
18     $unlink = "../";
19     $Toegestaan = array("image/jpg", "image/jpeg", "image/png", "image/gif");
20     $sql = "";
21
22     if (empty($Afbeelding) || $Afbeelding['size'] == 0) {
23         $sql = "UPDATE projects SET title=?, subtext=?, text=? WHERE id=?";
24         if ($stmt = $conn->prepare($sql)) {
25             $stmt->bind_param("sssi", $_POST['title'], $_POST['subtext'], $_POST['text'], $_GET['id']);
26             $stmt->execute();
27             // $stmt->close();
28             header("Location: ../admin/dashboard.php");
29         }
30     } else {
31         header('Location: ../admin/changepost.php?error=mysql');
32     }
33 }
34 elseif ($Afbeeldingnaam != $Huidig && in_array($type, $Toegestaan)) {
35     unlink($unlink.$Huidig);
36     move_uploaded_file($Tijdelijk, "$map.$Afbeeldingnaam");
37     $sql = "UPDATE projects SET title=?, subtext=?, text=?, headimage=? WHERE id=?";
38     $imagenew = $map.$Afbeeldingnaam;
39     if ($stmt = $conn->prepare($sql)) {
40         $stmt->bind_param("ssssi", $_POST['title'], $_POST['subtext'], $_POST['text'], $imagenew, $_GET['id']);
41         $stmt->execute();
42         // $stmt->close();
43         header("Location: ../admin/dashboard.php");
44     }
45     else {
46         header('Location: ../admin/changepost.php?error=mysql');
47     }
48 } else {
49     header('Location: ../admin/changepost.php?error=image_niet_geupload');
50 }
51 } else {
52     header('Location: ../admin/changepost.php?error=fields');
53 }
54 ?>
```

We hebben ook gebruik gemaakt van one page content. We hebben dus in aparte files die code gemaakt en dan via een require die op de home pagina gezet. Uiteindelijk kwam alles onder elkaar op de homepage. Zie hieronder een voorbeeld van de onepage code voor de biografie die middels een require op de home pagina is gezet.

```
1  <?php
2  session_start();
3  require('php/database.php');
4  $sql = "SELECT id,title,subtext,text,headimage FROM projects ORDER BY date DESC LIMIT 6;";
5  $result = $conn->query($sql);
6  ?>
7  <!doctype html>
8  <html lang="en">
9  <head>
10     <meta charset="utf-8">
11     <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
12     <?php require("components/style.php"); ?>
13     <title><?php echo $open;?> - Home</title>
14 </head>
15
16 <body>
17     <header>
18         <div class="container text-center">
19             <h1><?php echo $open;?></h1>
20             <p>Software Developer - Web Developer - Student</p>
21         </div>
22     </header>
23     <?php require("components/navbar.php"); ?>
24     <section id="article">
25         <div class="container">
26             <div class="row">
27                 <?php
28                 foreach ($result as $item) {
29                     $image = $item['headimage'];
30                     echo "
31                     <div class='column col-sm-12 col-md-6 col-lg-4'>
32                         <article class='card' style='background-image: url(";
33                         echo $image;
34                         echo ");' >
35                         <div class='card-body'>
36                             <h2>".$item['title']. "</h2>
37                             <p>".$item['subtext']. "</p>
38                             <p class='read'><a class='stretched-link' href='post.php?id=".$item['id']."'>Lees verder...</a></p>
39                         </div>
40                     </article>
41                     </div>
42                     ";
43                 }
44                 ?>
45             </div>
46         </div>
47     </section>
48     <?php
49     require("onpage/bio.php");
50     require("onpage/contact.php");
51     require("components/footer.php");
52     ?>
53     <!-- Optional JavaScript -->
54     <!-- jQuery first, then Popper.js, then Bootstrap JS -->
55     <?php require("components/scripts.php"); ?>
56     <script src="js/mail/jqBootstrapValidation.js"></script>
57     <script src="js/mail/contact_me.js"></script>
58 </body>
59
60 </html>
```

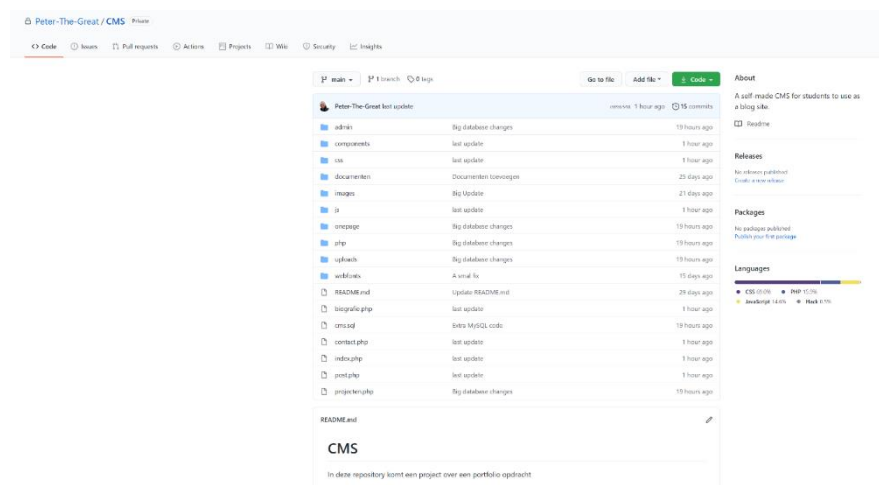
Zie hieronder hoe we voor de functionaliteiten prepared statements hebben gebruikt omdat die veiliger zijn dan sql queries en om sql-injectie te voorkomen

```
1  <?php
2  if($stmt = $conn->prepare("SELECT text FROM aboutme WHERE id = 1")) {
3      $stmt->execute();
4      $stmt->store_result();
5
6      if ($stmt->num_rows > 0) {
7          $stmt->bind_result($text);
8          $stmt->fetch();
9          $stmt->close();
10     }
11 }
12 ?>
13 <div class="container biografie-container text-center mt-5">
14     
15     <h2>Over Mij <?php echo $open;?></h2>
16 </div>
17 <section class="container" id="biografie">
18     <div class="row">
19         <span class="maxw"><?php echo $text;?></span>
20     </div>
21 </section>
```

Zie hieronder een voorbeeld van een stuk code met commentaar om aan te geven wat er gedaan wordt.

```
9 // Insert into DATABASE
10 if(isset($_POST["title"], $_POST["text"], $_POST["subtext"], $_POST['Huidige_Afbeelding'])) {
11
12     $Huidig = $_POST['Huidige_Afbeelding'];
13     $Afbeelding = $_FILES['image'];
14     $Tijdelijk = $Afbeelding['tmp_name'];
15     $Afbeeldingnaam = $Afbeelding['name'];
16     $type = $Afbeelding['type'];
17     $map = "uploads/";
18     $unlink = "../";
19     $Toegestaan = array("image/jpg", "image/jpeg", "image/png", "image/gif");
20     $sql = "";
21
22     if (empty($Afbeelding) || $Afbeelding['size'] == 0) {
23         $sql = "UPDATE projects SET title=?, subtext=?, text=? WHERE id=?";
24         if ($stmt = $conn->prepare($sql)) {
25             $stmt->bind_param("sssi", $_POST['title'], $_POST['subtext'], $_POST['text'], $_GET['id']);
26             $stmt->execute();
27             //$stmt->close();
28             header("Location: ../admin/dashboard.php");
29         }
30     }
```

De programmataal die we gebruikt hebben is visual studio code. We hadden gebruik gemaakt van Visual Live Share zodat we beiden konden bewerken in de code. Daarbij hebben we voor het opslaan van de code het op Github gezet. Verder hadden we de applicatie opgeslagen op een lokale server en die elke werksessie daarop gerunt. Zie hieronder hoe het project was opgeslagen in Github.



Voor mij waren nieuwe technieken die ik zeker weer wil gaan gebruiken dat je meerdere PHP bestanden maakt met functionaliteiten erin die je dan met een require weer gebruikt op pagina's die dat nodig hebben zoals een header. Daarbij ook het gebruik van prepared statements. Die zijn ook een stuk veiliger tegen SQL injectie. We hebben eigenlijk voor alle functionaliteiten zoals update, delete etc. prepared statements gebruikt.



















Bewijsmateriaal werkprocessen Pjotr

P1-K1-W2 – Beheren van Gegevens – Pjotr Wisse – CMS

In dit deel van het document wordt informatie gespecificeerd over inloggegevens, databasegegevens, aanpassingen, eisen en wijzigingen van de applicatie. Dit zal in uitermate detail worden omschreven. Er zal een complete omschrijving zijn van de database en file structuur.

De Filestructuur

We beginnen eerst met de file structuur. De file structuur is gebaseerd op een best wel basic file structuur.

 .git	16-2-2021 09:59	Bestandsmap	
 admin	8-2-2021 11:37	Bestandsmap	
 components	16-2-2021 08:29	Bestandsmap	
 css	1-2-2021 20:11	Bestandsmap	
 documenten	22-1-2021 13:05	Bestandsmap	
 images	1-2-2021 10:57	Bestandsmap	
 js	16-2-2021 08:32	Bestandsmap	
 onepage	15-2-2021 12:12	Bestandsmap	
 php	9-2-2021 08:37	Bestandsmap	
 uploads	16-2-2021 09:45	Bestandsmap	
 webfonts	1-2-2021 20:28	Bestandsmap	
 biografie.php	16-2-2021 08:39	PHP-bestand	2 kB
 cms.sql	16-2-2021 09:46	SQL-bestand	6 kB
 contact.php	16-2-2021 09:58	PHP-bestand	5 kB
 index.php	16-2-2021 08:40	PHP-bestand	2 kB
 post.php	16-2-2021 08:39	PHP-bestand	2 kB
 projecten.php	15-2-2021 11:46	PHP-bestand	2 kB
 README.md	18-1-2021 11:04	MD-bestand	1 kB

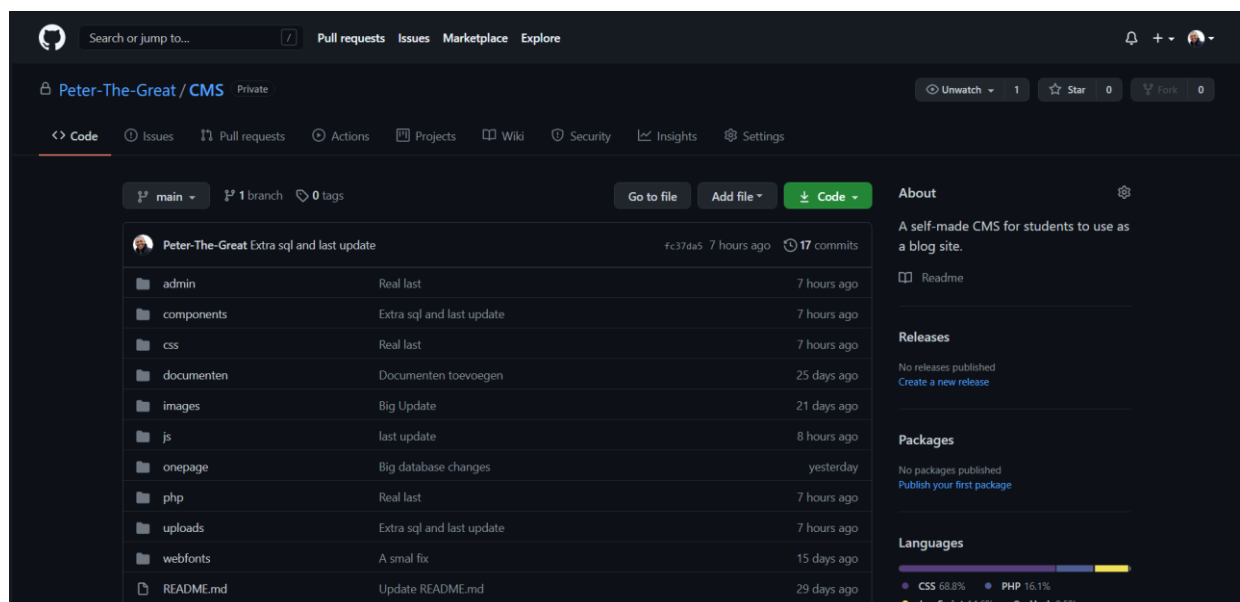
We zien hier in ieder geval alle PHP pagina's die openbaar zijn zoals de biografie-, contact-, index-, post- en projecten pagina. De JS en CSS folder zijn nog best wel simpel uit te leggen aan gezien daar de CSS en de javascript functionaliteit wordt behouden, dit zijn de front-end talen samen met de html in de PHP pagina's. De PHP folder daar zitten alle backend bestanden in die zorgen voor het aanmaken, wijzigen en verwijderen van post, maar ook de connectie met de database. Daarnaast wordt ook het inlog systeem hier behoudt zoals authenticatie proces en uitlog proces.

Components is een folder, waar bepaalde componenten in zitten die behoren bijna

elke pagina van de website, zoals een navbar, footer en links naar CSS en JS bestanden.

Admin is een folder waar alleen de administrator of de klant zelf toegang tot heeft. Hier zit een dashboard, post, en wijzig pagina's voor zijn bio en persoonlijke informatie. Om toegang te krijgen tot de pagina moet de gebruiker eerst inloggen met zijn of haar gegeven inlog gegevens.

De rest zoals onepage is bedoeld voor de homepage die voordoet als een onepage pagina, zodat een bezoeker gelijk een overzicht krijgt van de persoon zelf. Web fonts zijn voor web fonts. Images is een folder bedoelt voor standard images (wordt later misschien verwijderd) en uploads is een folder waar alle geüploade plaatjes worden opgeslagen. Als laatste, de .git folder is een folder bedoel voor onze git server, we hebben daarom ook het project op GitHub gezet voor back-up.

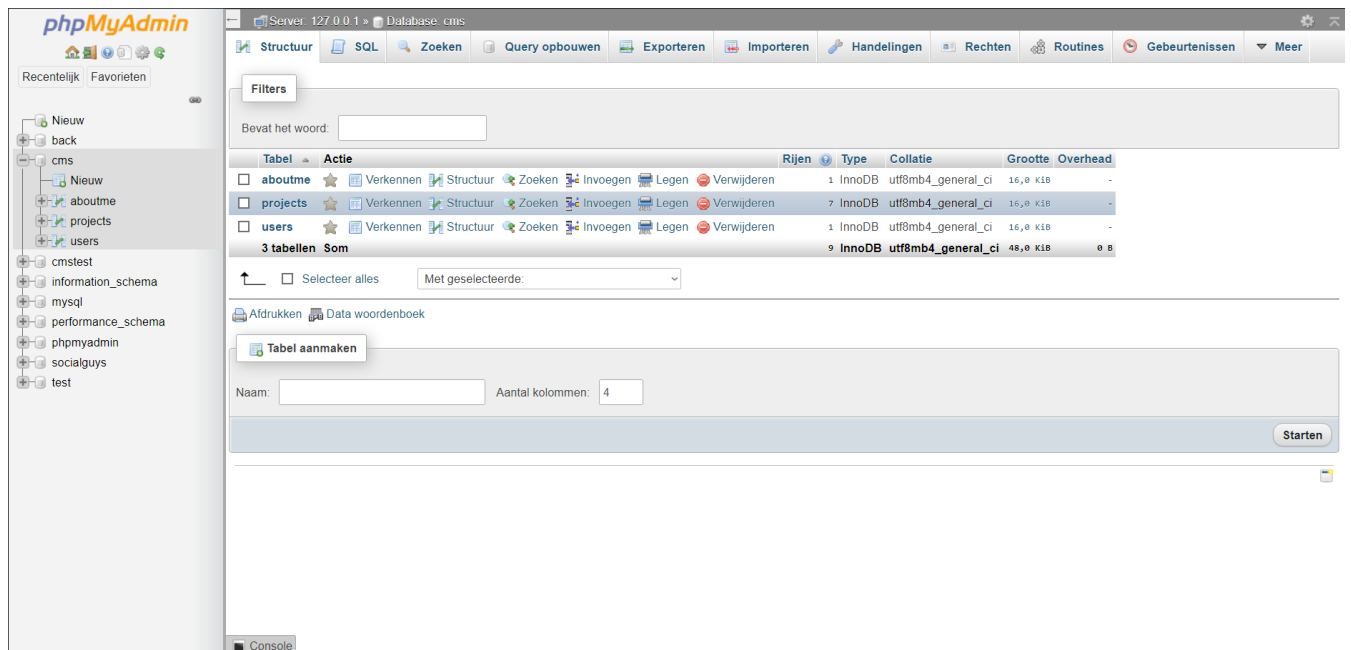


Hiernaast is er ook nog een kleine readme file toegevoegd voor GitHub.

De Database

De database is een hele simpele database die geen relaties nodig heeft, hierdoor kan er meer gefocuste worden op wat er kan worden toegevoegd in de tabellen.

Misschien dat we later relaties aanmaken, maar met hoe het project er nu uit ziet hebben we besloten om daar niet verder in te gaan. Hier is een overzicht van alle tabellen in de database:



We gebruiken Phpmyadmin als databasemanagement software voor een paar redenen.

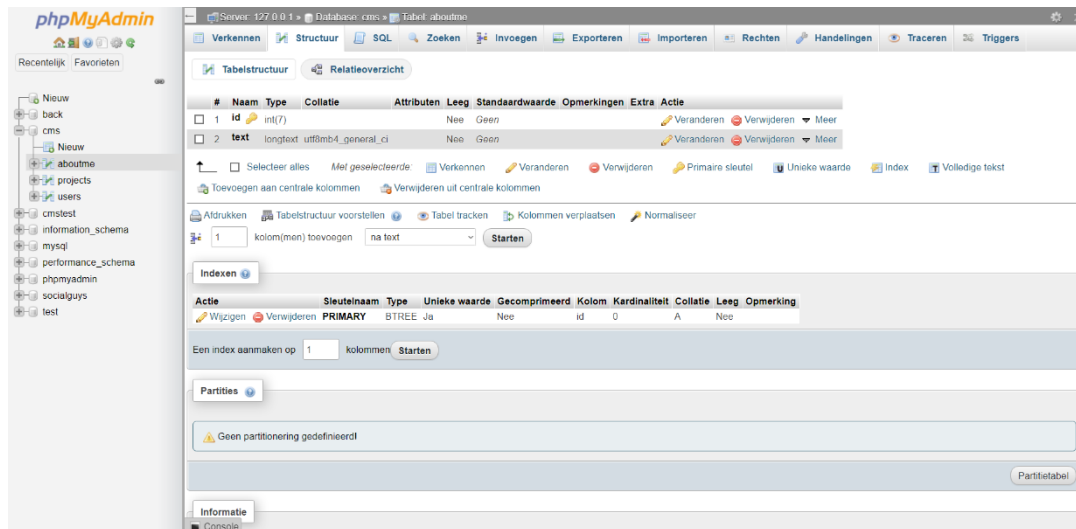
- Het is gratis te gebruiken, het zit in elke lamp of XAMPP server en is heel gemakkelijk om te leren hoe het werkt.
- Maakt gebruik van mysql, hierdoor kunnen we makkelijk met PHP samenwerken via de MYSQLi extension.
- Het is nog best flexibel, bijvoorbeeld als je een database wil exporteren naar een andere database en je kan data opslaan en weergeven waarmee iedereen kan me werken.

Phpmyadmin kan gebruikt worden in allemaal verschillende servers en de voor school gebruiken we ook phpmyadmin.

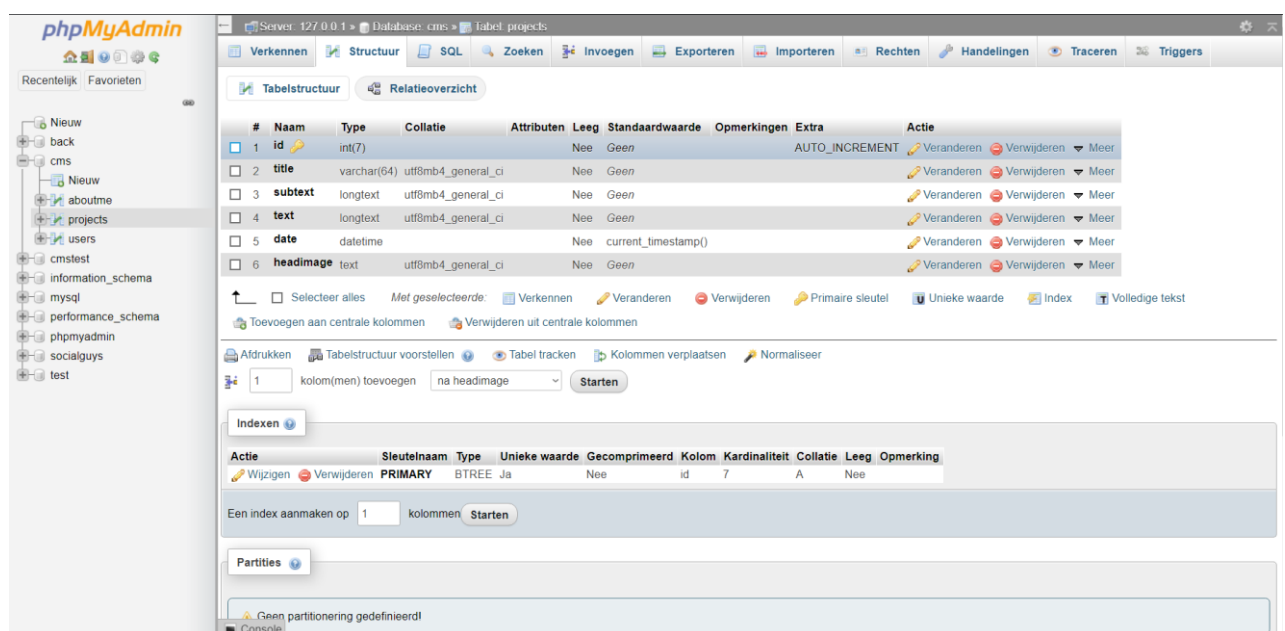
Terug naar de database, de database bestaat uit drie tabellen. Een aboutme-, projects- en users-tabel.

De aboutme tabel bestaat uit een simpel ID en een stuk tekst. Dit is de biografie van de gebruiker, hierin kunnen ze algemene informatie over zichzelf vertellen. Op de website als je inlogt en gaat naar biografie kun je gebruik maken van de tinymce api, hierin kan tekst en plaatjes ingezet worden. Deze data wordt in een blob van html tekst door verstuurd naar de database en opgeslagen. De ID is een int(7) en is de

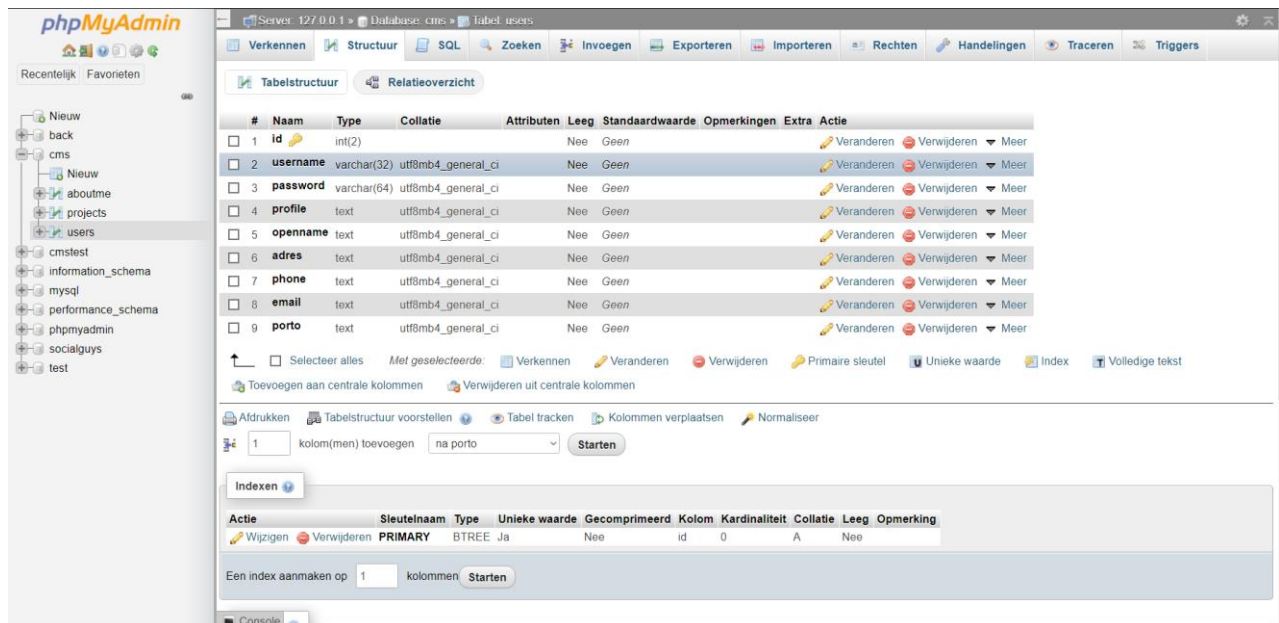
primary key van de tabel, de biotext wordt opgeslagen als een longtext, dit is makkelijk want we weten niet zeker hoeveel tekst en data de schrijver wil zetten in zijn/haar teksten.



De volgende is de project tabel dit is waar alle posts in worden opgeslagen. Dus de tekst, introductie tekst (subtekst), titel, thumbnail foto, een datum en een ID. Het ID is de primary key en wordt opgeslagen als een random uniek ID. Titel, subtekst en tekst zijn allemaal onderdelen voor de tekst op een post. De subtekst en de tekst worden allemaal op dezelfde manier opgeslagen als de biotext. Een longtext, gemaakt met de tinymce api. De datum wordt opgeslagen als een `current_timestamp()` in mysql, hierdoor kunnen we gemakkelijk de datum opslaan van wanneer de post werd gemaakt. De thumbnail van de post wordt opgeslagen als de naam van het bestand met een tekst type. De afbeelding zelf word in de upload folder opgeslagen dat doen we via PHP's `move_uploaded_file` functie.



Als laatste de user tabel. Dit is een tabel met een username, password, profiel, openname (niet username maar echte naam), adres, telefoon, email en portfolio.



#	Naam	Type	Collatie	Attributen	Leeg	Standaardwaarde	Opmerkingen	Extra	Actie
1	id	int(2)			Nee	Geen			Veranderen Verwijderen Meer
2	username	varchar(32)	utf8mb4_general_ci		Nee	Geen			Veranderen Verwijderen Meer
3	password	varchar(64)	utf8mb4_general_ci		Nee	Geen			Veranderen Verwijderen Meer
4	profile	text	utf8mb4_general_ci		Nee	Geen			Veranderen Verwijderen Meer
5	openname	text	utf8mb4_general_ci		Nee	Geen			Veranderen Verwijderen Meer
6	adres	text	utf8mb4_general_ci		Nee	Geen			Veranderen Verwijderen Meer
7	phone	text	utf8mb4_general_ci		Nee	Geen			Veranderen Verwijderen Meer
8	email	text	utf8mb4_general_ci		Nee	Geen			Veranderen Verwijderen Meer
9	porto	text	utf8mb4_general_ci		Nee	Geen			Veranderen Verwijderen Meer

Actie	Sleutelnaam	Type	Unieke waarde	Gecomprimeerd	Kolom	Kardinaliteit	Collatie	Leeg	Opmerking
Wijzigen Verwijderen	PRIMARY	BTREE	Ja	Nee	id	0	A	Nee	

Hier wordt een ID opgeslagen, voor nu als 1 simpel nummer. Je om in te loggen heb je een username en een wachtwoord, deze twee worden opgeslagen als een varchar en bij het wachtwoord wordt het opgeslagen met een sha1 encryptie methode. Een Profile is werkt in principe hetzelfde als een thumbnail, maar het is jou profiel foto, die kun je altijd veranderen. Openname is je publieke naam, meer hoeft ik niet te zeggen. Daarna heb je telefoon, email, adres en portfolio. De eerste drie zijn simpele tekst/varchar types die worden weer gegeven bij de contact pagina. Dat geldt ook voor de portfolio, maar de portfolio is eigenlijk een link die naar de portfolio linkt, we hebben nu nog geen optie om het te uploaden naar de webserver, dat zal later mogelijk zijn.

De database is zo gecreëerd dat alles netjes gestructureerd is en dat het snel in de database kan komen. We uploaden alle insert, select, update en delete functies via prepared statements, hiermee voorkomen wij SQL injecties en optimaliseren we de tijd en code om iets in de database toe te voegen. Als je meer wilt

Gegevens

Voor nu zijn de inloggegevens het volgende:

- Username: admin
- Wachtwoord: admin
- Echte naam: Niels Segaar
- Email: Segaar86@Gmail.com

- Portfolio: <https://84669.ict-lab.nl/Front2/Periode%201/Fluid%20Design/Portofolio/IMG/Pjotr%20Wisse%20English.pdf>
- Tel: +31-06958737273
- Adres: Heer Broekstraat 48

Software en talen gebruikt in een makkelijk lijstje:

- XAMPP: <https://www.apachefriends.org/index.html>
- Phpmyadmin: <https://www.phpmyadmin.net/>
- HTML, CSS, JS, PHP, SQL: <https://www.w3schools.com>
- Bootstrap: <https://getbootstrap.com>
- JQuery: <https://jquery.com>

B1-K3-W1 – Optimalisatie van het product – Pjotr Wisse – CMS

In dit deel van het document zal ik omschrijven hoe we ons product hebben geoptimaliseerd, met de verzoeken van de klant. We zullen een overzicht maken van hoe de applicatie in elkaar zit en welke talen en functies wij gebruiken om de applicatie zo veel mogelijk te optimaliseren.

Optimalisatie van applicaties

Bij een applicatie wil je graag dat alles vlot, snel en flexibel is. Niemand wil graag een slome website gebruiken en daarom is het ook belangrijk om je web applicatie te optimaliseren. Dit met meerdere manieren bereikt worden. Het gebruik maken van de juiste software, optimaliseren van programmeer talen zoals javascript, PHP en SQL of het ontwerp van de website gemakkelijker en toegankelijker te maken.

Optimalisatie van de applicatie met software

Ten eerste, moesten we beslissen, met welke software gaan we werken? We hebben besloten om met XAMPP te werken voor onze server, omdat XAMPP alle noodzakelijke componenten heeft voor een normale webserver. XAMPP bevat:

- Een Apache server
- PHP
- Perl
- Een Mysql database
- Phpmyadmin als databasemanagement programma (meer informatie bij beheert gegevens)
- En een mail server (nog niet volledig compleet)

Optimalisatie van de applicatie met editors

Ten tweede, met welke software/code editor gaan we de website maken? Het besluit was om Visual Studio Code te gebruiken voor een paar redenen:

- **Belangrijk:** Het is overal toegankelijk op elk apparaat
- Het heeft geïntegreerde git extensie en is makkelijk te combineren met GitHub
- Veel andere downloadbare extensies
- **Belangrijk:** De extensie Live Share een extensie die ervoor zorgt dat developers aan een gedeeld project kunnen werken in een sessie die wordt gehost op een lokale computer. Plus je kan ook je local server delen. Meer info: (<https://visualstudio.microsoft.com/services/live-share/>)

Het belangrijkste is dat Visual Studio code ook je code optimaliseert en het Visual is handig als development tool aangezien je veel functies en extensies kan gebruiken.

Optimalisatie van de applicatie met programmeer talen.

Als laatste hoe gaan we onze applicatie optimaliseren met de programmeer talen die we gaan gebruiken. In ieder geval gaan we gebruik maken van Javascript, PHP, HTML, CSS en SQL als de talen die we gaan gebruiken in onze applicatie. Hier zijn een paar optimalisatie manieren die wij hebben gebruikt in onze

Mysql is een heel vaak gebruikt door meerdere bedrijven omdat optimalisatie op mysql heel belangrijk is. In mysql heb je een paar opties om zelf je SQL code te optimaliseren. We hebben een paar van deze optie gebruikt.:

- Gebruik geen * in je select clause, hierdoor zorg je ervoor dat je minder data moet inladen dat onnodig is.
- Zorg in PHP voor prepared statements, dit voorkomt SQL injecties.
- Gebruik zo min mogelijk functies in je query, anders duurt de query langer.

In PHP hebben we ook een paar manier gevonden:

- Prepared statements, hiermee voorkom je niet alleen SQL injecties, het zorgt er ook voor dat je code sneller wordt uitgevoerd.
- Gebruik bij het checken zo min mogelijk if statements, het zorgt ervoor dat het PHP sneller wordt geëxecuteerd.
- Bij select functies kunne we ook prepared statements uitvoeren en dan een store result functie toevoegen zo kunnen we het resultaat aan variabelen toevoegen.
- Voor de veiligheid hebben we er ook voor gezorgd dat we de strip tags en htmlspecialchars functies gebruikt, omdat het snel is en het zorgt ervoor dat er geen scripts kunnen worden uitgevoerd of iets dergelijks.

```
if ($stmt = $conn->prepare("INSERT INTO projects (id, title, subtext, text, headimage) values (
    NULL,?, ?, ?, ?)")) {
    $stmt->bind_param("ssss", $titel, $_POST['subtext'], $_POST['text'], $afbeelding);
    $stmt->execute();
    header("Location: ../admin/dashboard.php");
}
```

Voorbeeld van een prepared statement.

Javascript heeft meerdere manieren om code te optimaliseren een van die opties is de library JQuery, hierdoor hoef je minder code te schrijven en is je bestand ook niet zo groot. Daarnaast is JQuery zo gemaakt dat de code al gelijk wordt geoptimaliseerd. We gebruiken minder javascript dan bijvoorbeeld PHP, maar dat is ter zaken.

Om HTML en CSS te optimaliseren hebben we een CSS framework gebruikt. Bootstrap, bootstrap heeft allemaal ingebouwde functies en stijlen dat we nauwelijks met CSS hoefde te werken. Bootstrap heeft ook ingebouwde JQuery tags die je kan gebruiken in je html pagina.

Hieronder zijn nog een paar voorbeelden van optimalisatie bij de site met plaatjes.

```

if($stmt = $conn->prepare("SELECT title,subtext,text,headimage FROM projects WHERE id = ?")) {
    $stmt->bind_param("i", $_GET["id"]);
    $stmt->execute();
    $stmt->store_result();

    if ($stmt->num_rows > 0) {
        $stmt->bind_result($title, $subtext, $text, $image);
        $stmt->fetch();
    }
}

```

Hier zie je dat we bij select geen * gebruiken.

`require('php/database.php');` Gebruik maken van require om gedeeltes van pagina's niet te herhalen.

```

<?php
// Check for empty fields
if(empty($_POST['name']) || empty($_POST['email']) || empty($_POST['phone']) || empty($_POST['subject']) || empty($_POST['message']) || !filter_var($_POST['email'], FILTER_VALIDATE_EMAIL)) {
    http_response_code(500);
    exit();
}

$name = strip_tags(htmlspecialchars($_POST['name']));
$email = strip_tags(htmlspecialchars($_POST['email']));
$phone = strip_tags(htmlspecialchars($_POST['phone']));
$head = strip_tags(htmlspecialchars($_POST['subject']));
$message = strip_tags(htmlspecialchars($_POST['message']));

// Create the email and send the message
$to = "pjotrwl5@gmail.com";
$subject = "Website Contact Form: $name, with the subject: $head";
$body = "You have received a new message from your website contact form.\n\n"."Here are the details:
\n\nName: $name\nEmail: $email\nPhone: $phone\n\nMessage:\n$message";
$header = "From: noreply@gmail.com\n"; // This is the email address the generated message will be from. We recommend using something like noreply@yourdomain.com.
$header .= "Reply-To: $email";
//mail($to, $subject, $body, $header);
if(!mail($to, $subject, $body, $header)){
    http_response_code(500);
}
?>

```

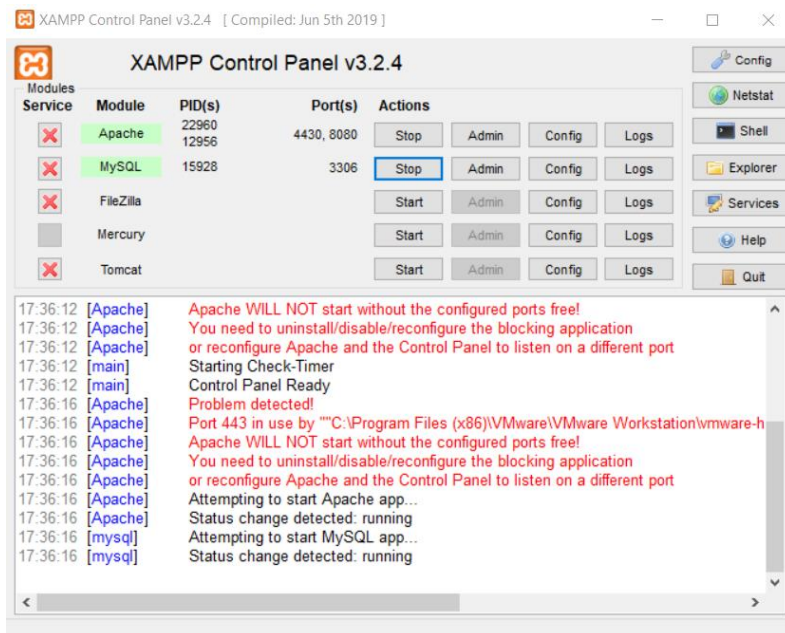
Dit is een mail verstuur script waarin strip tags wordt gebruikt met htmlspecialchars.

```

submitSuccess: function ($form, event) {
    event.preventDefault(); // prevent default submit
    // get values from FORM
    var name = $("input#name").val();
    var email = $("input#email").val();
    var phone = $("input#phone").val();
    var subject = $("input#subject").val();
    var message = $("textarea#message").val();
    var firstName = name; // For Success/Failure Message
    // Check for white space in name for Success/Failure Message
    if (firstName.indexOf(" ") >= 0) {
        firstName = name.split(" ").slice(0, -1).join("");
    }
    $this = $("#sendMessageButton");
    $this.prop("disabled", true); // Disable submit button
    $.ajax({
        url: "js/mail/contact_me.php",
        type: "POST",
        data: {
            name: name,
            email: email,
            phone: phone,
            subject: subject,
            message: message,
        },
        cache: false,
        success: function () {

```

We maken veel gebruik van JQuery en JQuery AJAX.



XAMPP is heel makkelijk om te gebruiken en is een nog best wel flexibele server applicatie die je kan gebruiken voor een webserver.

```
<script>
    tinyMCE.init({
        selector: '#subtext',
        height: '200',
        plugins: ['wordcount'],
        toolbar: 'undo redo | formatselect | ' +
            'bold italic ' +
            '| link unlink | bullist numlist outdent indent | ' +
            'removeformat',
        content_css: '//www.tiny.cloud/css/codepen.min.css'
    });
</script>
<script>
    tinyMCE.init({
        selector: '#text',
        height: '480',
        plugins: [
            'advlist autolink lists link image charmap print preview anchor',
            'searchreplace visualblocks code fullscreen',
            'insertdatetime media table paste code help wordcount'
        ],
        toolbar: 'undo redo | formatselect | ' +
            'bold italic forecolor backcolor | alignleft aligncenter ' +
            'alignright alignjustify | link unlink | bullist numlist outdent indent | ' +
            'removeformat | help',
        content_css: '//www.tiny.cloud/css/codepen.min.css'
    });
</script>
```

Tinymce om gemakkelijk tekst te creëren met javascript.

Dat was het einde van het document over hoe we onze website optimaliseren om gebruikers en de beheerders een snellere en fijnere ervaring te geven.

B1-K1-W4 - Realisatie van het product – Pjotr Wisse – CMS

In dit deel van het document zal ik omschrijven hoe de realisatie zal worden gedaan. Hier zal worden omschreven welke software, talen, editors en databases wij gaan gebruiken om zo de website op te bouwen.

Software die we gaan gebruiken

In deze opdracht hebben we niet zoveel software nodig om deze applicatie mogelijk te maken. Eerst wat we gaan we een webserver gebruiken, hiervoor gaan we XAMPP gebruiken, omdat XAMPP gratis is en het is makkelijkst te gebruiken voor simpele websites en servers. Het bevat een apache server, mysql support, PHP support en dat zorgt er ook voor dat we met de MySQLi extensie kunnen werken in PHP. Hierop zullen we vooral de applicatie en de database testen, zodra we klaar zijn gaan we het hosten op een van onze servers. Met plesk kunnen we ervoor zorgen dat de databasen en de website toegankelijk zijn via een domainnaam.



Voor de database gebruiken we phpmyadmin als databasemanagement software. Hierdoor kunnen gebruikers makkelijker in de gaten houden hoe de database eruit ziet en wat er allemaal op staat. Phpmyadmin ondersteund onze mariadb database die wij gaan gebruiken voor onze applicatie.

Programeer talen die we gaan gebruiken.

Met software zoals mysql en phpmyadmin kun je al verwachten dat we voor het grootste deel met PHP en SQL gaan werken. PHP en SQL zijn talen die makkelijk te gebruiken zijn als je een webapplicatie wilt maken met een database. PHP kan ervoor zorgen dat je query's en code veilig wordt verstuurd naar je mysql database, via checks, prepared statements en encryptie protocollen zoals sha1. SQL is ook handig

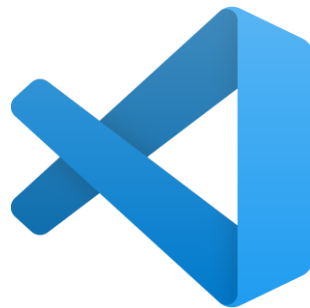
omdat structureel de taal makkelijk te begrijpen is en het is gestructureerd aan gegeven hoe je de data makkelijk kan krijgen. Naast PHP en SQL gaan we ook andere talen gebruiken zoals HTML, CSS en Javascript. Javascript is handig voor scripts bij bepaalde pagina's zoals de contact pagina en met HTML en CSS kan de site mooi gestileerd worden.



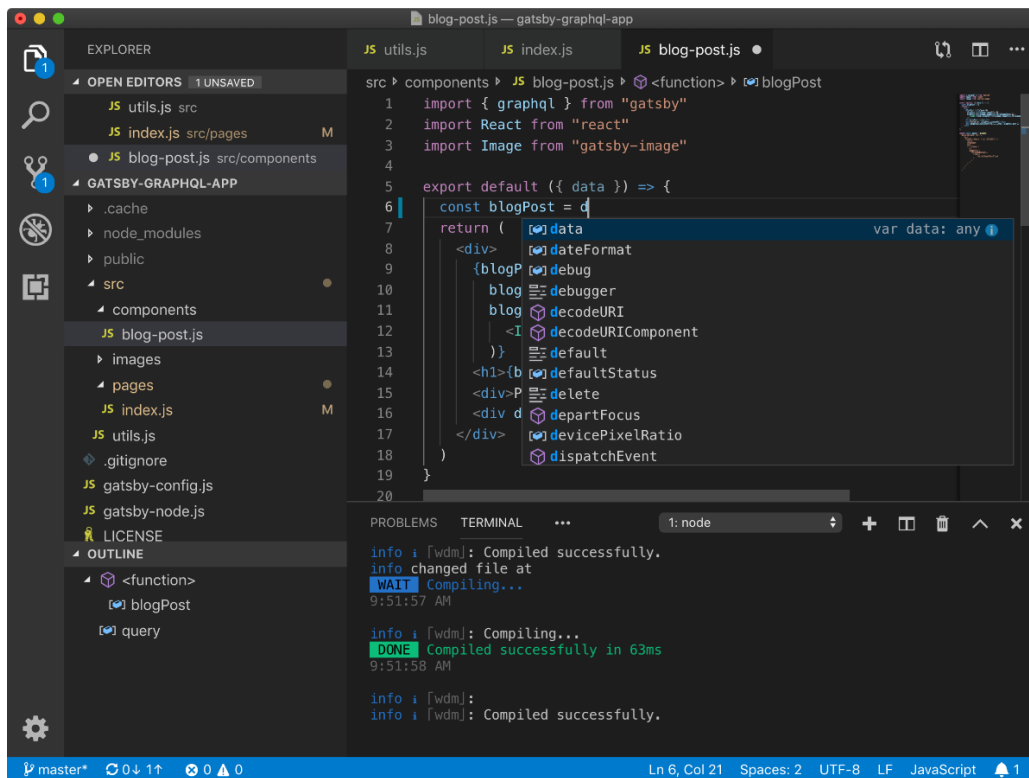
Dit zijn alle programmeer talen die wij gaan gebruiken in ons project als je een overzicht wilt zien van hoeveel we van deze talen gebruikt hebben kun je het op onze GitHub pagina bekijken: <https://github.com/Peter-The-Great/CMS>

Tools die we gaan gebruiken.

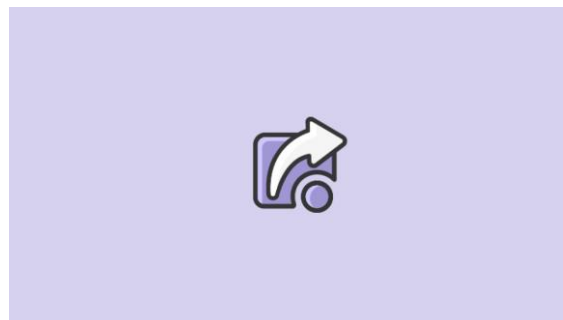
We hebben al gekeken naar welke belangrijke software we gaan gebruiken voor de database, maar in dit gedeelte gaan we bespreken met welke tools we de applicatie gaan bouwen en welke tools we zullen gebruiken.



In dit project zullen we gebruik maken van Visual Studio Code met visual studio code kunnen we een mooi overzicht creëren van ons project, visual studio code heeft ook een geïntegreerde GitHub en git ondersteuning (komt deels omdat het van Microsoft is visual en GitHub).



Voorbeeld van visual studio code in actie



Met visual kunnen we ook live sharen, dat is een extension waarmee developers aan een project kunnen werken die alleen maar op een computer hoeft te staan. Je deelt je project met andere via een sessie en iedereen kan code toevoegen, veranderen en verwijderen en ze kunnen je webserver bezoeken. Visual studio code is ook handig voor debuggen, syntax correctie en automatische suggesties om zo de code sneller te laten werken. Er zijn op visual studio code nog allemaal een boel andere extensie die we kunnen gebruiken om ons project beter te maken, maar die zijn niet zo belangrijk als deze.

Back-up en versiebeheer



Als het gaat om versiebeheer en back-up gebruiken we altijd onze vertrouwelijke git en GitHub services. GitHub geeft ons mogelijk om onze code op te slaan als back-up en als toegankelijkheid, zodat we er altijd toegang tot hebben. GitHub heeft een heel makkelijke manier repositories te maken en geeft je instructies om hoe je een remote server adres kan instellen via de git command line. Het is voor ons belangrijk en voor de klant belangrijk om altijd toegang te hebben tot de code op een veilige en toegankelijke manier.

Voor de database hadden we bijna een grootte fout gemaakt, alles op de database was kwijt geraakt, maar gelukkig was er wel een back-up gemaakt van de server zelf en alle informatie in de database alleen niet de database van de opdracht. Gelukkig konden we nog een test database overzetten naar de database en toen hadden we alles weer terug gezet naar hoe het hoorde. Om deze gebeurtenis is het voor ons dus heel belangrijk om altijd van onze database een back-up te maken. Meestal maken we op een hard kopie van de gehele XAMPP server of alleen een SQL bestand van de hele database.

```
22
23
24
25
26
27 -- tabelstructuur voor tabel 'aboutsme'
28
29
30 * CREATE TABLE 'aboutsme' (
31   'id' int(4) NOT NULL,
32   'text' longtext NOT NULL,
33 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
34
35
36 -- gegevens worden geladen voor tabel 'aboutsme'
37
38
39 INSERT INTO 'aboutsme' ('id','text') VALUES
40 (1, 'op mijn naam is Niek Segar en ik ben student bij het Grafisch Lyceum Rotterdam en ik doe de opleiding Mediatechnologie. Perilleusque sili amet ore l arc
41
42
43
44
45 -- tabelstructuur voor tabel 'projects'
46
47
48 * CREATE TABLE 'projects' (
49   'id' varchar(4) NOT NULL,
50   'title' varchar(55) NOT NULL,
51   'abstract' longtext NOT NULL,
52   'text' longtext NOT NULL,
53   'date' datetime NOT NULL DEFAULT current_timestamp(),
54   'headimage' text NOT NULL,
55 ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
56
57
58 -- gegevens worden geladen voor tabel 'projects'
59
60
```

Voorbeeld van SQL back-up van de tabellen in de database

In kort we hebben vooral GitHub gebruikt als versiebeheer en Back-up. Daarnaast gebruiken we ook SQL bestanden om al onze database informatie op te slaan, zodat we het later kunnen gebruiken en kunnen zetten op andere database.

Einde van dit document.