SYLLABUS

BOOTCAMP

SPIDER MONKEYS



Versie: Maki 1.0

Datum: 5-4-2022

Bewerkers: Tomi Ristic

Mark Goos

Susanne Mol

|  |
| --- |
| INHOUDSOPGAVE |

# 

[**INHOUDSOPGAVE**](#_cqx0jl5bmxfp) **3**

[**VOORAF**](#_elil6hgfnjt7) **5**

[OPSTARTEN](#_ocwhf3fr440p) 6

[Hardware](#_6uloopt18kb0) 6

[Installeren](#_8gann8onsag4) 6

[Licenties](#_eg6689qkpjii) 7

[Laracasts: Screencasts for the modern developer](#_ev92raamhotl) 7

[HUISREGELS](#_uykwsw2ag9vh) 8

[**OVER DE OPLEIDING**](#_ypajh49539ah) **9**

[WAAR WORD JE VOOR OPGELEID?](#_lrcw9w7vvupu) 9

[DE OPBOUW VAN DE OPLEIDING](#_fhnwyiy99r69) 10

[Leeruitkomsten per onderdeel](#_57dv5cpdf0lk) 10

[Planning van de opleiding](#_fivvphacaixc) 11

[**VOORAF**](#_eflsr6o5fop8) **13**

[OVER DE OPDRACHTEN](#_p6og9r237jsg) 13

[Vragen?](#_8p7fi9sbhwwl) 13

[Vergeet niet! Experimenteer!](#_55euowr427) 13

[BEGELEIDING](#_hk7s2yqwrmhc) 14

[Beschikbaarheid techcoach](#_psf29ycdo3wd) 14

[De stand-up](#_l19d6c1q3z9i) 14

[De stand-down](#_pblb5arffqpn) 14

[Samenwerken met je buddy](https://docs.google.com/document/d/1VGst1uzgIgNEmAFY6LdGDLdCPq_ac23xJbwPut2xxsM/edit#heading=h.anc24cel9co4) [14](https://docs.google.com/document/d/1VGst1uzgIgNEmAFY6LdGDLdCPq_ac23xJbwPut2xxsM/edit#heading=h.anc24cel9co4)

[**VOORAF**](#_j2wcmj7tdvj0) **16**

[HET DOEL VAN BLOK 1](#_k6p3rl324j7t) 16

[OVERZICHT LESSTOF VOOR BLOK 1](#_4i1dwp9lux7) 16

[DE OPDRACHTEN](#_icq9fbfz6msv) 16

[OVERZICHT LESSTOF](#_pqmvwfgbw4t8) 17

[**OPDRACHTEN BLOK 1**](#_t5li1xxjk0j7) **18**

[Reminder!](#_80t2bmitzlt1) 18

[Niveau 1 - Minimaal Benodigde Programmeerkennis](#_4sbsubulb4c7) 18

[PERSOONLIJK WERKDOCUMENT VOOR BLOK 1](#_2s337xqhaje) 18

[Opdracht: 1.1 - Introductie: Wat is PHP?](#_66lwtrwkug1q) 18

[Opdracht: 1.2 - Installing and using PHP](#_r25jd9p843e9) 19

[Feedbackmoment met de tech coach](https://docs.google.com/document/d/1VGst1uzgIgNEmAFY6LdGDLdCPq_ac23xJbwPut2xxsM/edit#heading=h.j2nb0pf6uiex) [19](https://docs.google.com/document/d/1VGst1uzgIgNEmAFY6LdGDLdCPq_ac23xJbwPut2xxsM/edit#heading=h.j2nb0pf6uiex)

[Opdracht: 1.3 - Syntax](#_skeuott325j3) 19

[Opdracht: 1.4 - Comments](#_xe0irg2lbfgf) 19

[Opdracht: 1.5 - Variables](#_8zwjgjw4hcl9) 19

[Opdracht: 1.6 - Echo en print Statements](#_4wk3ddmltff7) 20

[Feedbackmoment met de tech coach](https://docs.google.com/document/d/1VGst1uzgIgNEmAFY6LdGDLdCPq_ac23xJbwPut2xxsM/edit#heading=h.xgizfi9p7h) [20](https://docs.google.com/document/d/1VGst1uzgIgNEmAFY6LdGDLdCPq_ac23xJbwPut2xxsM/edit#heading=h.xgizfi9p7h)

[Opdracht: 1.7 - Data Types](#_bmcw4vyh0380) 20

[Opdracht: 1.8 - Constants](#_dmobtqbbzwth) 20

[Opdracht: 1.9 - Operators](#_arwwcvcqbkqc) 21

[Opdracht: 1.10 - If…Else…Elseif.](#_5iyb422wgr8u) 21

[Feedbackmoment met de tech coach](https://docs.google.com/document/d/1VGst1uzgIgNEmAFY6LdGDLdCPq_ac23xJbwPut2xxsM/edit#heading=h.cdxdalux1c3a) [21](https://docs.google.com/document/d/1VGst1uzgIgNEmAFY6LdGDLdCPq_ac23xJbwPut2xxsM/edit#heading=h.cdxdalux1c3a)

[Opdracht: 1.11 - Loops](#_63al1r9k64ow) 21

[Opdracht: 1.12 - Functions](#_xyfb2xjtnbf9) 21

[Opdracht: 1.13 - Arrays](#_dn3hnig75sny) 22

[Feedbackmoment met de tech coach](#_vofyb3tzcjtr) 22

[Niveau 2 - Exercising the Fundamentals in PHP.](#_99po5rqhd9st) 23

[Niveau 3 - Project Specifieke Kennis](#_pj010qdg3hv5) 23

[**VOORAF**](#_t5ug621bspzg) **25**

[HET DOEL VAN BLOK 2](#_z5jhs65i725z) 25

[OVERZICHT LESSTOF VOOR BLOK 2](#_mok81q5ihdxb) 25

[DE OPDRACHTEN](#_f27mm27m46d8) 25

[OVERZICHT LESSTOF](#_9jkp5q9mazy) 25

[**OPDRACHTEN BLOK 2**](#_dwk7cq1feyzj) **26**

[Reminder!](#_vjqmzhpwzi48) 26

[Niveau 1 - Minimaal Benodigde Programmeerkennis](#_u43mt0q0t615) 26

[PERSOONLIJK WERKDOCUMENT BLOK 2](#_wxo18q6cyqv3) 26

[Opdracht: 1.1 - WHAT IS OOP?](#_ltlrdf7fr3hb) 26

[Opdracht: 1.2 - Classes and Objects](#_70c4xf3cicg5) 27

[Feedbackmoment met de tech coach](#_m3dq9iap2nhn) 27

[Opdracht: 1.3 - Constructor](#_yktrqgn3nflf) 27

[Opdracht: 1.4 - Access Modifiers](#_m7mjk9i17daw) 28

[Opdracht: 1.5 - Inheritance](#_5h5sap7y30el) 28

[Feedbackmoment met de tech coach](#_5ebzso3t7z3e) 28

[Opdracht: 1.6 - Class Constants](#_xj3q3jqxvyo) 28

[Opdracht: 1.7 - Static Methods](#_8dk023n1of58) 29

[Opdracht: 1.8 - Static Properties](#_74tzcahvxayc) 29

[Feedbackmoment met de tech coach](#_w8ql1v2frd3v) 29

[Niveau 2 - Exercising the Fundamentals in PHP.](#_eh1pov3kggt0) 30

[Niveau 3 - Project Specifieke Kennis](#_fv0bwko3bhnt) 30

[**VOORAF**](#_8f004s78zltq) **32**

[HET DOEL VAN BLOK 3](#_fi4ach4ubsmf) 32

[OVERZICHT LESSTOF VOOR BLOK 3](#_49bnbepml48k) 33

[DE OPDRACHTEN](#_8x9ok94jxjee) 33

[OVERZICHT LESSTOF](#_xb5chy8iwu7v) 33

[**OPDRACHTEN BLOK 3**](#_yr0hw5692opw) **34**

[Reminder!](#_140c59mv2thq) 34

[Niveau 1 - Practicing SQL Statements](#_oz6zsq45nu4i) 34

[Niveau 2 - Designing your Database](#_rrn4l533cxt8) 36

[**VOORAF**](#_7fnrqd34jjkc) **39**

[HET DOEL VAN BLOK 4](#_a3jsn9ymo5h4) 39

[OVERZICHT LESSTOF VOOR BLOK 4](#_3upx9fr29d5b) 40

[DE OPDRACHTEN](#_4vhmuh5mcdd3) 40

[OVERZICHT LESSTOF](#_9t51uh4jtrfo) 40

[**OPDRACHTEN BLOK 4**](#_fq5ak9s6ypaj) **41**

[Reminder!](#_aq7dqxd0zh1) 41

[Niveau 1 - Fundamentals in Laravel](#_qx3j03rcm3uv) 41

[Focus](#_l0r47mn09gew) 41

[Project Fases](#_3o413hm60hr7) 41

[Inleidend](#_evh6842xs35i) 42

[Fase 1 - Setup your Project](#_qi8ygzma7lhz) 43

[Opdracht: 1.0 - Installation & Configuration](#_4wy30nnbi6tc) 43

[Fase 2 - The Basic Workflow](#_ty10z5otr4lv) 45

[Overzicht Ontwikkelstappen](#_24m0oleorf9a) 46

[Opdracht: 1.1 - Migrations](#_7lm40un9mdt0) 46

[Opdracht: 1.2 - Seeders](#_xa4bxxswvx6g) 49

[Opdracht: 1.4 - Models](#_gga6h47348z7) 53

[Opdracht: 1.3 - Factories (Optioneel)](#_vqzpauyuteh5) 53

[Opdracht: 1.5 - Routing](#_xwvga2nazn2c) 54

[Opdracht: 1.6 - Controllers](#_k7i69clv4l7x) 54

[Opdracht: 1.7 - Views](#_xfeo18v9fcfs) 55

[Fase 3 - CRUD actions.](#_if3f5aqk6q3f) 55

[Fase 4 - Model Relations](#_1acog4bj9bdf) 55

[**VOORAF**](#_y6bhkokruzhv) **57**

[HET DOEL VAN BLOK 5](#_4229gr9o01o7) 57

[OVERZICHT LESSTOF VOOR BLOK 5](#_p6xzpm3wfkyq) 58

[EINDOPDRACHT: WEBSHOP IN LARAVEL](#_vw1mc9g6waif) 58

[OVERZICHT LESSTOF](#_sajxovav7p6z) 58

[**EINDOPDRACHT:   
WEBSHOP IN LARAVEL**](#_ncdmpkczxn9d) **59**

[Opdracht 1.0 - Introductie](https://docs.google.com/document/d/1MWMENnFc_EVe7DQEa2qVO1k_6_8J7CZPugOpNVBNtlE/edit#heading=h.wn5au0qratjf) [59](https://docs.google.com/document/d/1MWMENnFc_EVe7DQEa2qVO1k_6_8J7CZPugOpNVBNtlE/edit#heading=h.wn5au0qratjf)

[Opdracht 1.1 - Research](#_i1m0jiviszmo) 59

[Opdracht 1.2 - Development](#_rgdqnz24ej8z) 60

|  |
| --- |
| VOORAF |

## **OPSTARTEN**

### **Hardware**

Intel i3 of vergelijkbare AMD processor

Mac gebruikers: Intel Generatie processor

8 Gb. RAM Geheugen

256 SSD schijf

Aanbevolen Hardware

Intel i5/i7 of AMD Ryzen 5 of M1 (Mac chip)

16 Gb. RAM geheugen

Linux of Mac OS

Gedurende de Base- en Bootcamp werken we met Linux of wanneer je een Mac hebt met Mac OS (beide gebaseerd op UNIX). We werken niet in Windows.

### **Installeren (Optioneel ga eerst door)**

Microsoft Visual Studio Code

https://code.visualstudio.com/download

PHP 8.\*

Afhankelijk van je OS: laat Google je helpen. Soms is PHP al geïnstalleerd in Linux en Mac

MySQL 8.\*

NodeJS en NPM https://nodejs.org/en/download/Homebrew https://brew.sh/index\_nl

Composer

<https://getcomposer.org/>  
  
NodeJS en NPM

<https://nodejs.org/en/download/>

phpMyAdmin

https://www.phpmyadmin.net/downloads/

GIT

geïnstalleerd <https://github.com/>

Help!

Heb je problemen met het downloaden van de programma’s of geen mogelijkheid tot het gebruik van een laptop met de juiste specificaties? Geef dit vooral aan we hebben leenlaptops beschikbaar. De eerste week van de bootcamp is geheel vrijblijvend. Mocht het programma toch niet helemaal bij je passen dan bieden we je de gelegenheid om te stoppen met de bootcamp.

### **Licenties**

#### **Laracasts: Screencasts for the modern developer**

Jullie krijgen via jullie codegorilla email accounts toegang tot [Laracasts](https://laracasts.com/). Hier kunnen allerlei screencasts over (web)development bekeken worden die jullie tijdens de bootcamp kunnen gebruiken als referentie. Aan het einde van de bootcamp stopt deze toegang dus doe er je voordeel mee maar geen paniek een groot deel van de Laracasts zijn ook gratis toegankelijk.

## 

## **HUISREGELS**

## 

## 

|  |
| --- |
| OVER DE OPLEIDING |

## **WAAR WORD JE VOOR OPGELEID?**

Bij CodeGorilla leer je in vier maanden de basis skills van full stack web development. Zo heb je snel zicht op een baan, waar je je verder kunt specialiseren in front end of back end development.

Programmeren, of webdevelopment, bestaat uit front end en back end development. Omdat je bij ons de basisbeginselen van beide leert, hieronder een korte samenvatting van de software waarmee je leert werken. Ook krijg je bij ons aanvullende trainingen in o.a. scrum (een veel gebruikte werkmethode) en verbeter je je persoonlijke vaardigheden.

* Front end development  
  Front end development houdt zich bezig met alles wat de gebruiker kan zien, dus de visuele aspecten. Je leert werken met HTML, CSS, JavaScript, VueJS, GIT version control, en (Cloud) deployment. [Meer lezen over het leren van front end development](https://codegorilla.nl/leer-programmeren/front-end-development).
* Back end development  
  Back end development omvat alle onzichtbare processen. Je leert werken met codetalen als PHP, Laravel, MySQL, Linux en Object Oriented Programming en Model View Controller. [Meer lezen over het leren van back end development](https://codegorilla.nl/leer-programmeren/back-end-development).
* Trainingen  
  Naast programmeren krijg je trainingen om te leren werken als programmeur. Denk aan agile- en scrum methodiek, project planning en timemanagement. [Lees hier meer over de trainingen](https://codegorilla.nl/updates/welke-21st-century-skills-leer-je).
* Persoonlijke vaardigheden  
  Om je zo snel mogelijk aan een baan te helpen, krijg je ook trainingen waarmee je essentiële vaardigheden als communiceren, presenteren, pitchen verbetert. [Lees hier meer over de trainingen](https://codegorilla.nl/updates/welke-21st-century-skills-leer-je).

## 

## **DE OPBOUW VAN DE OPLEIDING**

### **Leeruitkomsten per onderdeel**

|  |  |
| --- | --- |
| **Onderdeel** | **Leeruitkomsten** |
| Fundamentals in PHP | * Je kunt kan een ‘platte’ PHP file schrijven en variabelen m.b.v. var\_dump tonen en interpreteren * Je kunt kent de verschillende datatypen als integer, array, object, float, boolean etc * Je bouwt een MySQL PHP module om de data uit de database op te halen en deze data mee te geven aan view |
| Introduction to Object Oriented programming | * Je kunt een ‘platte’ PHP file herschrijven naar een class * Je begrijpt het verschil tussen public, protected en private methods en variabelen * Je bent in staat om een class uit te breiden (extenden) en je begrijp het nut hiervan * Je begrijpt Name Spacing en je kunt dit toepassen in je website * Je begrijpt wat een model is * Je vertaalt tabellen naar Models * Je schrijft Controllers om data via Models uit de database te halen en mee te geven aan Views (MVC) |
| Data Interaction with SQL and Database Management in MYSQL | * Je begrijp wat een relationele database is en vertaalt deze naar MySQL. * Je bouwt je database op uit tabellen met relaties tussen tabellen |
| Laravel als Full Stack Framework | * Je kunt zelfstandig een PHP framework opzetten met behulp van NPM en Composer * Je begrijpt het MVC model en je richt het PHP framework op deze wijze in. * Je kunt kent het verschil tussen ‘platte’ PHP files en anderzijds een structuur om met OOP (Object Oriented Programming) een framework op te zetten. * Behalve back-end ben je ook instaat om je website een aantrekkelijk uiterlijk te geven. * Je kunt m.b.v. Composer een Laravel project aanmaken en een versie van Laravel kiezen (doel en uitkomst) * Je kunt m.b.v. Composer packages installeren die de webshop nodig heeft * Je kunt m.b.v. NPM CSS en Javascript packages installeren * Je kunt packages die je met NPM hebt geïnstalleerd toepassen in het Laravel framework * Je begrijpt ‘webpack.mix.js’ om Javascript en CSS samen te voegen * Je begrijpt wat ‘artisan’ is en je maakt hier veelvuldig gebruik van * Je begrijpt hoe de .env file werkt binnen Laravel en kunt de configuratie keys interpreteren * Je begrijpt hoe je een database tabel kunt aanmaken m.b.v. artisan en wat migraties zijn * Je kunt database migraties aanmaken en vervolgens database tabellen aanmaken met diverse datatypen * Je begrijp wat de verschillen zijn tussen een Model, een Controller en een View * Je begrijpt routing en je weet hoe je een route in Laravel moet aanmaken * Je begrijpt dat een Model een ‘vertaling’ is van een databasetabel en hoe je de relaties tussen tabellen in Laravel (en PHP) moet schrijven * Je weet wat ‘blade’ templating is en hoe je deze zowel voor de voor- als de achterkant moet gebruiken |
| Laravel als API Backend | * Je weet waar versiebeheer voor wordt gebruikt en je kunt met Git werken. |

### **Planning van de opleiding**

Onderstaande is een globale planning, waarvan kan worden afgeweken. De totale duur van de Bootcamp is 12 weken.

**Niveau 1 = 1, Niveau 2 = 2, Niveau 3 = 3.**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Week Nr. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| Fundamentals in PHP. | x | x |  |  |  |  |  |  |  |  |  |  |
| Introduction to Object Oriented programming |  | x | x | x |  |  |  |  |  |  |  |  |
| Data Interaction with SQL and Databases Management in MYSQL |  |  |  | x | x | x |  |  |  |  |  |  |
| Laravel als Full Stack Framework |  |  |  |  |  |  | x | x | x |  |  |  |
| Laravel als backend api |  |  |  |  |  |  |  |  | x | x | x | x |

Progressie Overzicht Bootcamp

**Bootcamp Week 1 tot en met 6**

|  |  |  |  |
| --- | --- | --- | --- |
| Blok | Niveau 1 | Niveau 2 | Niveau 3 |
| Fundamentals in PHP. | Verplicht | Verplicht | Optioneel |
| Introduction to Object Oriented programming. | Verplicht | Verplicht | Optioneel |
| Data Interaction with SQL and Databases Management in MYSQL | Verplicht | Verplicht | Optioneel |

**Bootcamp Week 6 tot en met 12**.

|  |  |
| --- | --- |
| Project Start: Webshop in Laravel and Angular. | |
| Laravel als Full Stack Framework. | |
| Laravel als API Backend. | |
| Super tempo. | Je loopt echt hard voor als je na 4 weken al kan beginnen aan het leren van Laravel. Hier oogst je alle voordelen mee aan het einde want je kan meer oefenen. |
| Voldoende tempo. | Goed streven is om in week 6 te beginnen aan Laravel. Probeer dit voor elkaar te krijgen! Als het niet lukt is het niet het |
| Matig tempo. | Na 8 weken beginnen met Laravel is minimaal required zodat je de basics van het framework kan leren. Het combineren met Angular is dan niet heel realistisch. |

|  |
| --- |
| VOORAF |

## **OVER DE OPDRACHTEN**

Er gaan opdrachten via W3Schools uitgevoerd worden om je kennis te toetsen. Deze opdrachten kunnen uitgevoerd worden door de links in het document te bezoeken en de opdrachten daar uit te voeren. Deze opdrachten worden niet gecontroleerd.

### Vragen?

Mocht je er niet uitkomen klop aan bij jouw techcoach. Wanneer je behoefte hebt aan feedback, verdieping of merkt dat je bepaalde concepten lastig vindt, geef dit dan aan.

### Vergeet niet! Experimenteer!

Naast het toetsen van je kennis via de opdrachten op W3Schools is het nog belangrijker je de stof eigen te maken. Het is belangrijk dat je het “echte programmeren” vanaf scratch oefent. Na opdracht 1.2 van dit blok kun je op je eigen machine PHP files aanmaken en uitvoeren. Als je de exercises op W3Schools hebt uitgevoerd kun je in de overgebleven tijd zelf experimenteren met het combineren van de aangeleerde concepten om ze eigen te maken.

Je kunt met je techcoach bespreken wat handig is. Samen formuleren jullie dan maatwerk opdrachten die aansluiten bij jouw leerbehoefte en/of de leerbehoefte van de groep.

## 

## **BEGELEIDING**

## **Beschikbaarheid techcoach**

De techcoach voor de bootcamp is **maandag, dinsdag tussen 09:00 en 16:00** **aanwezig op locatie** en beschikbaar voor ondersteuning. Op **woensdag, donderdag en vrijdag werk je zelfstandig aan de bootcamp.**

De tech coach is op de vermelde dagen ook altijd **via Slack** bereikbaar. Op dagen dat de techcoach niet aanwezig is kun je bij je buddy terecht en je vragen noteren voor de volgende stand-up.

### **De stand-up**

**Maandag, dinsdag wordt om 09:00 uur de stand-up gehouden**. Het is de bedoeling dat je hierbij aanwezig bent. Tijdens de stand-up behandelt de aanwezige tech coach de volgende onderdelen:

1. Je geeft aan met welke opdracht je bezig bent.
2. Je geeft aan wat je van plan bent die dag te doen.
3. Je geeft aan hoe je de stof tot nu toe ervaart.
   * Rood: Ik loop vast of kom bijna niet verder.
   * Geel: Het is een uitdaging maar ik heb het gevoel dat ik verder kom.
   * Groen: Ik ervaar het als makkelijk en voorzie geen problemen.
4. Je geeft aan of er nog iets anders speelt dat niet behandeld is.

### **De stand-down**

**Maandag, dinsdag wordt aan het einde van de dag om 16:00 uur gezamenlijk afgesloten met een stand-down.**

Tijdens de stand-down behandelt de aanwezige coach de volgende onderdelen:

1. Je geeft of je gedurende de dag nog tegen problemen aangelopen bent en je ergens nog hulp bij nodig bent.
2. Je geeft aan of het plan wat je tijdens de stand-up hebt aangegeven gelukt is.
3. Je geeft aan of er nog overige dingen spelen die je graag behandelt zou willen hebben!

|  |
| --- |
| BLOK 1FUNDAMENTALSIN PHP |

## 

# 

|  |
| --- |
| VOORAF |

## **HET DOEL VAN BLOK 1**

Om aan het einde van de bootcamp een webshop te kunnen realiseren leer je in blok 1 de fundamentele principes van de programmeertaal PHP. Het is belangrijk om dit blok heel zorgvuldig aandacht te besteden aan de concepten die in de opdrachten voorbij komen. Deze zullen de basis vormen van je programmeerkennis in de toekomst.

## **OVERZICHT LESSTOF VOOR BLOK 1**

De volgende concepten moeten **minimaal** beheerst zijn na blok 1.

1. Introductie: Wat is PHP?
2. Installing and using PHP
3. Syntax
4. Comments
5. Variables
6. Echo & Print
7. Data Types
8. Constants
9. Operators
10. If…Else…Elseif.
11. Loops
12. Functions
13. Arrays

## **DE OPDRACHTEN**

De opdrachten dienen op de volgende manier uitgevoerd te worden:

|  |  |  |
| --- | --- | --- |
| Niveau 1 | Fundamentals in PHP. | Verplicht |
| Niveau 2 | Exercising the Fundamentals in PHP | Verplicht |
| Niveau 3 | Project Specifieke Kennis | Verplicht (Tenzij je achterloopt) |

## 

## **OVERZICHT LESSTOF**

In de opdrachten die in het volgend hoofdstuk beschreven zijn staan linkjes naar de lesstof in W3Schools. Deze lesstof is in tekst weergegeven. Als je er de voorkeur aan geeft om video’s te kijken over de lesstof, of als het concept nog niet duidelijk is nadat je de tekst hebt gelezen, dan kun je aanvullend video’s bekijken op Laracasts.

* [W3Schools: PHP →](https://www.w3schools.com/php/default.asp)
* [The PHP Practitioner Episode 1 t/m 10 →](https://laracasts.com/series/php-for-beginners)
* [PHP: The Right Way →](https://phptherightway.com/)

|  |
| --- |
| OPDRACHTEN BLOK 1 |

Hieronder staan de opdrachten en lesstof beschreven.

Het kijken van de video’s of lezen van de teksten is geen doel op zich.

## **Reminder!**

**Probeer je steeds te focussen op het echt snappen van de onderstaande concepten voordat je dieper op de stof ingaat of verder werkt anders kan dat heel verwarrend werken.** **Alle toekomstige kennis in het programmeren wordt voortgebouwd op deze fundamentele principes. Begrijp je dit niet voldoende zal het lastig zijn om toekomstige kennis op te doen.**

## **Niveau 1 - Fundamentals in PHP**

## **PERSOONLIJK WERKDOCUMENT VOOR BLOK 1**

De antwoorden op de opdrachten kan je verwerken in jouw eigen persoonlijke werkdocument. De tech coach zal in de aangegeven feedback momenten hier samen met je naar kijken.

|  |
| --- |
| [Bijlage Persoonlijk Werkdocument Blok 1 →](#_qqfh1fm4bwek) |

## **Opdracht: 1.1 - Introductie: Wat is PHP?**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools de [PHP Intro →](https://www.w3schools.com/php/php_intro.asp) |
| Opdracht 1.1\* | Leg in je eigen woorden uit wat PHP is. |
| Opdracht 1.2\* | Leg in je eigen woorden uit wat je kan doen met PHP. |
| Opdracht 1.3\* | Leg in je eigen woorden uit wat een PHP file is. |

## 

## **Opdracht: 1.2 - Installing and using PHP**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over het [installeren en gebruiken van PHP →](https://www.w3schools.com/php/php_install.asp)  Lees op PHP The Right Way over [getting started with PHP →](https://phptherightway.com/" \l "getting_started) |
| Opdracht 1.2.1\* | Leg in eigen woorden uit wat je nodig hebt om PHP zelf te draaien? |
| Opdracht 1.2.2\* | **Voer de volgende stappen uit:**  Download en Installeer Xampp op je machine met default settings en php 8.1.  Start je Xampp en zorg dat de Apache module draait (groen).  Click op de “Admin” knop in de Xampp app.  Click op de php info tab en controleer of php8.1 draait.  Click op de “Explorer” knop in de Xampp app en vindt de htdocs map.  Informatie: Uit deze map kan je met Xampp je php files runnen uitvoeren in de browser.  Maak in de htdocs map een “opdrachten” map aan.  Sla in de “opdrachten” map al je php files op!  Maak in de “opdrachten” map een index.php file aan.  Plaats in de file de volgende code: <?php print(“PHP Works!”); ?>  Bezoek in je browser localhost/opdrachten/index.php om het bestand te zien.  **Laat dit zien aan je techcoach voor controle.** |

## **Opdracht: 1.3 - Syntax**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over [syntax in PHP →](https://www.w3schools.com/php/php_syntax.asp) |
| Opdracht 1.3.1\* | Voer [Syntax Opdrachten 1 t/m 5 →](https://www.w3schools.com/php/exercise.asp?filename=exercise_syntax1) uit. |

## **Opdracht: 1.4 - Comments**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over [comments in PHP →](https://www.w3schools.com/php/php_comments.asp) |
| Opdracht 1.4.1\* | Leg in je eigen woorden uit waar comments voor bedoeld zijn? |

## **Opdracht: 1.5 - Variables**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over [variables and variables scope in PHP →](https://www.w3schools.com/php/php_variables.asp)  Bekijk [Episode 3: Variables →](https://laracasts.com/series/php-for-beginners/episodes/3) op Laracasts. |
| Opdracht 1.5.1\* | Voer [Variables Opdrachten 1 t/m 2 →](https://www.w3schools.com/php/exercise.asp?filename=exercise_variables1) uit. |

## **Opdracht: 1.6 - Echo en print Statements**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over [echo en print Statements in PHP →](https://www.w3schools.com/php/php_echo_print.asp) |
| Opdracht 1.6.1\* | Leg uit waar “echo” en “print” voor gebruikt worden? |
| Opdracht 1.6.2\* | Leg uit wat het verschil tussen “echo” en “print” is? |

## **Opdracht: 1.7 - Data Types**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over [PHP Data Types →](https://www.w3schools.com/php/php_datatypes.asp) |
| Opdracht 1.7.1\* | Leg uit wat een String is? |
| Opdracht 1.7.2\* | Leg uit wat een Integer is? |
| Opdracht 1.7.3\* | Leg uit wat een Float is? |
| Opdracht 1.7.4\* | Leg uit wat een Boolean is? |
| Opdracht 1.7.5\* | Leg uit wat een Array is? |
| Opdracht 1.7.6\* | Leg uit wat NULL is? \*optioneel\* |
| Opdracht 1.7.7\* | Leg uit wat een Object is? \*optioneel\* |

## **Opdracht: 1.8 - Constants**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over [Constants in PHP →](https://www.w3schools.com/php/php_constants.asp) |
| Opdracht 1.8.1\* | Leg uit wat constants zijn en hoe deze verschillen van variabelen? |

## **Opdracht: 1.9 - Operators**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over [Operators in PHP →](https://www.w3schools.com/php/php_operators.asp) |
| Opdracht 1.9.1\* | Voer [Operators Opdrachten 1 t/m 4 →](https://www.w3schools.com/php/exercise.asp?filename=exercise_operators1) uit. |

## **Opdracht: 1.10 - If…Else…Elseif.**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over [If…Else…Elseif in PHP →](https://www.w3schools.com/php/php_if_else.asp) |
| Opdracht 1.10.1\* | Voer [If… Else…Elseif Opdrachten 1 t/m 4 →](https://www.w3schools.com/php/exercise.asp?filename=exercise_ifelse1) uit. |

## **Opdracht: 1.11 - Loops**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over [Loops in PHP →](https://www.w3schools.com/php/php_looping.asp) |
| Opdracht 1.11.1\* | Leg uit waar loops voor gebruikt worden? |
| Opdracht 1.11.2\* | Leg uit welke verschillende soorten loops er zijn en wat hun functie is? |
| Opdracht 1.11.3\* | Voer [Loops Opdrachten 1 t/m 4 →](https://www.w3schools.com/php/exercise.asp?filename=exercise_loops4) uit. |

## **Opdracht: 1.12 - Functions**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over [Functions in PHP →](https://www.w3schools.com/php/php_functions.asp) |
| Opdracht 1.12.1\* | Leg uit welke twee soorten functies er bestaan en wat het verschil is? |
| Opdracht 1.12.1\* | Leg uit hoe de standaard syntax van een function in PHP eruit ziet? |
| Opdracht 1.12.1\* | Leg uit wat een argument is? |
| Opdracht 1.12.1\* | Leg uit hoe je een argument meegeeft aan een function? |
| Opdracht 1.12.1\* | Leg uit hoeveel argumenten je kan meegeven aan een functie? |
| Opdracht 1.12.1\* | Leg uit hoe je meerdere argumenten meegeeft aan een function? |
| Opdracht 1.12.1\* | Leg uit hoe je default argumenten meegeeft aan een function? |
| Opdracht 1.12.1\* | Leg uit hoe je een waarde returned? |
| Opdracht 1.12.1\* | Leg uit wat het nut is van een return? |
| Opdracht 1.12.1\* | Voer [Functions Opdrachten 1 t/m 4 →](https://www.w3schools.com/php/exercise.asp?filename=exercise_functions1) uit. |

## 

## **Opdracht: 1.13 - Arrays**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over [Arrays in PHP →](https://www.w3schools.com/php/php_arrays.asp) |
| Opdracht 1.13.1\* | Leg uit wat het nut is van een Array? |
| Opdracht 1.13.2\* | Leg uit welke soorten Arrays er bestaan en wat de verschillen zijn? |
| Opdracht 1.13.3\* | Leg uit op welke twee manieren je Arrays kan definiëren?  Tip: **Shortened Syntax.** |
| Opdracht 1.13.4\* | Leg uit wat je allemaal kan opslaan in een Array? |
| Opdracht 1.13.5\* | Voer [Arrays Opdrachten 1 t/m 4 →](https://www.w3schools.com/php/exercise.asp?filename=exercise_arrays1) uit. |

## **Feedbackmoment met de tech coach**

|  |
| --- |
| Als je deze opdrachten hebt uitgevoerd, stuur dan een bericht naar jouw tech coach om mee te kijken met de uitgevoerde opdrachten. Je gaat nu met je tech coach bespreken wat je verder nog wilt leren over de PHP fundamentals, samen maak je hiervoor een plan. |

## 

## **Niveau 2 - Exercising the Fundamentals in PHP.**

Voer de volgende oefenopdrachten uit in volgorde. In dit onderdeel gaan we de belangrijkste en meest voorkomende onderwerpen oefenen en herhalen namelijk **loops, functions** en **arrays.** Voer de opdrachten hieronder uit. Upload de antwoorden naar je eigen Github repository en deel de link met je techcoach.

## **Opdracht: 2.1 - Exercising Loops**

|  |  |
| --- | --- |
| Opdracht 2.1.1\* | Create a script that displays 1-2-3-4-5-6-7-8-9-10 on one line. There will be no hyphen(-) at starting and ending position. |
| Opdracht 2.1.2\* | Create a script using a for loop to add all the integers between 0 and 30 and display the total. |
| Opdracht 2.1.3\* | Create a script to construct the following pattern, using nested for loop  \*  \* \*  \* \* \*  \* \* \* \*  \* \* \* \* \* |
| Opdracht 2.1.4\* | Create a script to construct the following pattern, using a nested for loop. \* \* \*  \* \* \*  \* \* \* \*  \* \* \* \* \*  \* \* \* \* \*  \* \* \* \*  \* \* \*  \* \*  \* |
| Opdracht 2.1.5\* | Write a PHP script using nested for loop that creates a chess board as shown below. Use table width="270px" and take 30px as cell height and width. Example Output Below. |

## **Opdracht: 2.2 - Exercising Functions**

|  |  |
| --- | --- |
| Opdracht 2.2.1\* | Write a function to check whether a number is prime or not.   * Note: A prime number (or a prime) is a natural number greater than 1 that has no positive divisors other than 1 and itself. * Example Prime Numbers:     Optioneel mag je ook dit proberen na te maken als je verdieping zoekt! . |
| Opdracht 2.2.2\* | Write a function to reverse a string. |
| Opdracht 2.2.3\* | Write a PHP function that checks whether a string is all lowercase. |
| Opdracht 2.2.4\* | Write a function to sort an array. |
| Opdracht 2.2.5\* | Write a PHP function that checks whether a person is eligible to vote.  Condition   * Je moet een parameter **naam** en **leeftijd** kunnen als parameter kunnen passen aan je function. * Je moet **dynamisch** kunnen outputten met **echo** of de persoon oud genoeg is om te stemmen.   + Tomi je bent niet oud genoeg om te stemmen.   + De stem grens is 18 jaar. |

## **Opdracht: 2.3 - Exercising Arrays**

|  |  |
| --- | --- |
| Opdracht 2.3.1\* | $color = array('white', 'green', 'red', 'blue', 'black');  Write a script which will display the following string  "The memory of that scene for me is like a frame of film forever frozen at that moment: the red carpet, the green lawn, the white house, the leaden sky. The new president and his first lady. - Richard M. Nixon"  The words 'red', 'green' and 'white' will come from $color. |
| Opdracht 2.3.2\* | Input: $color = array('white', 'green', 'red'');  Write a PHP script which will display the colors in the following way :  Output :   * green * red * white   . |
| Opdracht 2.3.3\* | $ceu = array( "Italy"=>"Rome", "Luxembourg"=>"Luxembourg", "Belgium"=> "Brussels", "Denmark"=>"Copenhagen", "Finland"=>"Helsinki", "France" => "Paris", "Slovakia"=>"Bratislava", "Slovenia"=>"Ljubljana", "Germany" => "Berlin", "Greece" => "Athens", "Ireland"=>"Dublin", "Netherlands"=>"Amsterdam", "Portugal"=>"Lisbon", "Spain"=>"Madrid", "Sweden"=>"Stockholm", "United Kingdom"=>"London", "Cyprus"=>"Nicosia", "Lithuania"=>"Vilnius", "Czech Republic"=>"Prague", "Estonia"=>"Tallin", "Hungary"=>"Budapest", "Latvia"=>"Riga", "Malta"=>"Valetta", "Austria" => "Vienna", "Poland"=>"Warsaw") ;  Create a PHP script which displays the capital and country name from the above array $ceu. Sort the list by the capital of the country.  Sample Output :  The capital of Netherlands is Amsterdam  The capital of Greece is Athens  The capital of Germany is Berlin |
| Opdracht 2.3.4\* | Write a PHP script to calculate and display average temperature, five lowest and highest temperatures  Recorded temperatures : 78, 60, 62, 68, 71, 68, 73, 85, 66, 64, 76, 63, 81, 76, 73, 68, 72, 73, 75, 65, 74, 63, 67, 65, 64, 68, 73, 75, 79, 73  Expected Output :  Average Temperature is : 70.6  List of five lowest temperatures : 60, 62, 63, 63, 64,  List of five highest temperatures : 76, 78, 79, 81, 85, |
| Opdracht 2.3.5\* | Write a PHP function that checks whether a person is eligible to vote.  Condition   * Je moet een parameter **naam** en **leeftijd** kunnen als parameter kunnen passen aan je function. * Je moet **dynamisch** kunnen outputten met **echo** of de persoon oud genoeg is om te stemmen.   + Tomi je bent niet oud genoeg om te stemmen.   + De stem grens is 18 jaar. |
| Opdracht 2.6.5\* | Write a PHP script that displays the highest temperature that day and which city this was in.  $weeklyTemps = [  "Maandag" => [  "Groningen" => 25,  "Assen" => 12,  "Emmen" => 19  ],  "Dinsdag" => [  "Groningen" => 26,  "Assen" => 28,  "Emmen" => 19  ],  "Woensdag" => [  "Groningen" => 16,  "Assen" => 30,  "Emmen" => 35  ]  ];    **Example Output:** |

## **Niveau 3 - Diving Deeper**

Als je op dit punt bent aangekomen bespreek dan met je **tech coach** wat voor jou handig is om te doen.

Eenmaal op dit punt aangekomen kan je in Educative verder met oefenen. Hiervoor kun je de volgende course gebruiken. [Web Development in PHP](https://www.educative.io/path/web-development-php) →

Als leidraad voor je progressie het volgende:

1. Start bij Module 2 Learn PHP From Scratch en werk door tot Assessment 1. **Ga NIET verder met Object Oriented Programming.** Herhaal in module 2 de onderwerpen waar je op dit punt nog vraagtekens bij hebt voor extra oefening. Al deze onderwerpen zouden op dit punt te doen moeten zijn.
2. Als je klaar bent met Assement 1 ga door met Module 3. Het is goed als je tenminste tot en met Lesson 7 gedaan hebt.
3. Als het je lukt om door de hele module door te komen helemaal top! Ga ervoor!

**Ben je helemaal klaar bespreek dit met je techcoach!**

|  |
| --- |
| BLOK 2  INTRODUCTION TO OBJECT ORIENTED PROGRAMMING |

# 

# 

|  |
| --- |
| VOORAF |

## **HET DOEL VAN BLOK 2**

Nu je de fundamentals van het programmeren in PHP hebt geleerd gaan we aan de slag object oriented programming. Dit wordt in de praktijk gebruikt om complexe stukken abstract, modulair, behapbaar en overzichtelijk te houden. Als jullie uiteindelijk in het werkveld aan de slag gaan zal dit het grootste gedeelte van jullie tijd besteed worden aan deze methode van programmeren.

## **OVERZICHT LESSTOF VOOR BLOK 2**

De volgende concepten moeten **minimaal** beheerst zijn na blok 2.

1. What is OOP?
2. Classes/Objects
3. Constructor
4. Access Modifiers
5. Inheritance
6. Constants
7. Static Methods
8. Static Properties

## **DE OPDRACHTEN**

De opdrachten dienen op de volgende manier uitgevoerd te worden:

|  |  |  |
| --- | --- | --- |
| Niveau 1 | Minimaal Benodigde Kennis | Verplicht |
| Niveau 2 | Exercising the Fundamentals in Object Oriented PHP | Verplicht |
| Niveau 3 | Project Specifieke Kennis | Optioneel |

## **OVERZICHT LESSTOF**

Voor blok 2 wordt er wederom gebruik gemaakt van w3schools als basis en Laracasts als aanvullende lesstof.

* [W3Schools: PHP OOP (Object Oriented Programming) →](https://www.w3schools.com/php/php_oop_what_is.asp)
* [Laracasts: Object Oriented Principles in PHP →](https://www.w3schools.com/php/php_oop_what_is.asp)

|  |
| --- |
| OPDRACHTEN BLOK 2 |

Hieronder staan de opdrachten en lesstof beschreven.

Het kijken van de video’s of lezen van de teksten is geen doel op zich.

## **Reminder!**

**Probeer je steeds te focussen op het echt snappen van de onderstaande concepten voordat je dieper op de stof ingaat of verder werkt anders kan dat heel verwarrend werken.** **Alle toekomstige kennis in het programmeren wordt voortgebouwd op deze fundamentele principes. Begrijp je dit niet voldoende zal het lastig zijn om toekomstige kennis op te doen.**

## **Niveau 1 - Minimaal Benodigde Programmeerkennis**

## **PERSOONLIJK WERKDOCUMENT BLOK 2**

De antwoorden op de opdrachten kan je verwerken in jouw eigen persoonlijke werkdocument. De tech coach zal in de aangegeven feedback momenten hier samen met je naar kijken.

|  |
| --- |
| [Persoonlijke Werkbestand Blok 2 →](#_u0w6qbeiroxj) |

## **Opdracht: 1.1 - WHAT IS OOP?**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools de [PHP OOP Intro →](https://www.w3schools.com/php/php_oop_what_is.asp) |
| Opdracht\* | Leg in je eigen woorden uit wat het verschil is tussen flat PHP en Object Oriented Programming?  *OOP Alles is een object. - In PHP is niet alles een object... We hebben integers, floats, strings, arrays, resources, booleans en objects.* |
| Opdracht\* | Leg in je eigen woorden uit welke **vier voordelen** Object Oriented Programming heeft ten opzicht van flat PHP.  *Modulariteit voor eenvoudiger probleemoplossing*  *Hergebruik van code door overerving*  *Flexibiliteit door polymorfisme*  *Effectief oplossen van problemen* |
| Opdracht\* | Leg in je eigen woorden uit wat de twee hoofdonderdelen zijn van Object Oriented Programming?  ***Klassen*** *– Deze bepalen de staat, het gedrag en de identiteit van elk object. Klassen helpen ontwikkelaars de set eigenschappen te begrijpen die aan een object is gekoppeld of wordt gedeeld door een groep objecten.*  ***Objecten*** *–*  *Verzamelingen van gegevens en procedures die zijn gegroepeerd in afzonderlijke, herbruikbare entiteiten. Objecten hebben twee belangrijke kenmerken: data (beschrijft de toestand van een object en zijn unieke kenmerken) en gedrag (geeft aan hoe het samenwerkt met andere objecten).* |
| Opdracht\* | Leg in je eigen woorden uit wat het verschil is tussen een Class en een Object?  ***Klassen*** *– de definities voor het gegevensformaat en beschikbare procedures voor een bepaald type of objectklasse; kunnen zelf ook gegevens en procedures bevatten (ook wel klassemethoden genoemd), d.w.z. klassen bevatten de gegevensleden en lidfuncties*  ***Objecten*** *– instanties van klassen* |
| Opdracht\* | Leg in je eigen woorden uit wat er overgeërfd wordt van een Class als een object aangemaakt wordt?  *Overerving is de procedure waarbij een klasse de attributen en methoden van een andere klasse overerft. De klasse waarvan de eigenschappen en methoden worden geërfd, wordt de Parent-klasse genoemd. En de klasse die de eigenschappen van de bovenliggende klasse erft, is de klasse Child.* |
| Opdracht\* | Bedenk zelf een voorbeeld van een Class en bijbehorende Objecten anders dan afgebeeld op W3SCHOOLS.  <?php  class Employee  {    private $first\_name;    private $last\_name;    private $age;      public function \_\_construct($first\_name, $last\_name, $age)    {      $this->first\_name = $first\_name;      $this->last\_name = $last\_name;      $this->age = $age;    }      public function getFirstName()    {      return $this->first\_name;    }      public function getLastName()    {      return $this->last\_name;    }      public function getAge()    {      return $this->age;    }  }  $objEmployee = new Employee('Bob', 'Smith', 30);    echo $objEmployee->getFirstName(); // print 'Bob'  echo $objEmployee->getLastName(); // prints 'Smith'  echo $objEmployee->getAge(); // prints '30'  ?>  ?> |

## **Opdracht: 1.2 - Classes and Objects**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over het [Classes en Objects in PHP OOP →](https://www.w3schools.com/php/php_oop_classes_objects.asp) |
| Opdracht\* | Objecten erven dingen van elkaar over.  Leg uit in je eigen woorden uit wat het verschil is tussen objecten?  *Elk object heeft alle eigenschappen en methoden die in de klasse zijn gedefinieerd, maar ze hebben verschillende eigenschapswaarden.* |
| Opdracht\* | Wat is de meest basic Syntax definitie van een Class?  <?php class SimpleClass {     // property declaration     public $var = 'a default value';      // method declaration     public function displayVar() {         echo $this->var;     } } ?>  Uit welke **drie** onderdelen bestaat deze Syntax?  *Een klasse kan zijn eigen constanten, variabelen en functies .* |
| Opdracht\* | Hoe worden variables en functions in een Class genoemd?  *variabelen ("eigenschappen" genoemd) en functies ("methoden" genoemd).* |
| Opdracht\* | Uit welke twee verschillende soorten fundamentele onderdelen kan de body van een Class bestaan?  *Klasse is een set objecten die gemeenschappelijke kenmerken/gedrag en gemeenschappelijke eigenschappen/attributen deelt.* |
| Opdracht\* | Hoeveel objects kunnen er van een Class gemaakt worden?  *Theoretisch is er geen limiet.* |
| Opdracht\* | Welke keyword gebruik je om je object mee aan te maken?  *er is maar één manier (door een nieuw keyword te gebruiken), en de rest gebruikt intern een nieuw keyword.* |
| Opdracht\* | Waar wordt het “$this” keyword voor gebruikt?  *$this is een gereserveerd sleutelwoord in PHP dat verwijst naar het aanroepende object. Dit sleutelwoord is alleen van toepassing op interne methoden.* |
| Opdracht\* | Hoe kan je een property van binnen in de Class veranderen versus buiten de klasse?  *Een klasse bevat de methoden en eigenschappen die worden gedeeld door alle objecten die ervan worden gemaakt.*  *Hoewel de objecten dezelfde code delen, kunnen ze zich anders gedragen omdat er verschillende waarden aan kunnen worden toegewezen.* |
| Opdracht\* | Waar kun je de keyword **instanceOf** voor gebruiken?  *Het keyword instanceof wordt gebruikt om te controleren of een object tot een klasse behoort. De vergelijking retourneert true als het object een instantie van de klasse is, en false als dat niet het geval is.* |

## **Feedbackmoment met de tech coach**

|  |
| --- |
| Als je deze opdrachten hebt uitgevoerd, stuur dan een bericht naar jouw tech coach om mee te kijken met de uitgevoerde opdrachten. |

## **Opdracht: 1.3 - Constructor**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over [Constructor in PHP OOP →](https://www.w3schools.com/php/php_oop_constructor.asp) |
| Opdracht\* | Leg uit waar een Constructor voor gebruikt wordt en wat het voordeel is?  *Met een constructor kunt u de eigenschappen van een object initialiseren bij het maken van het object.* |
| Opdracht\* | Hoe ziet de basic syntax van de Constructor eruit?  *De construct-functie begint met twee underscores (\_\_)!* |

## **Opdracht: 1.4 - Access Modifiers**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over [Access Modifiers in PHP OOP →](https://www.w3schools.com/php/php_oop_access_modifiers.asp) |
| Opdracht\* | Welke drie Access Modifier bestaan er?  public  protected  private |
| Opdracht\* | Wat zijn hun verschillende rollen?  ***public*** *- de eigenschap of methode is overal toegankelijk. Dit is standaard*  ***protected****- de eigenschap of methode is toegankelijk binnen de klasse en door klassen die van die klasse zijn afgeleid*  ***private*** *- de eigenschap of methode is ALLEEN toegankelijk binnen de klasse* |
| Opdracht\* | Op welke twee onderdelen van een Class kan je Access Modifiers gebruiken?  ***public***  ***protected*** |
| Opdracht\* | Wat denk je dat het nut is van Access Modifiers in een “echte”  applicatie?  *Wijzigingen op toegangsniveau.* |

## **Opdracht: 1.5 - Inheritance**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over [Inheritance in PHP OOP →](https://www.w3schools.com/php/php_oop_inheritance.asp) |
| Opdracht\* | Wat is Inheritance?  *Inheritance in OOP = Wanneer een klasse is afgeleid van een andere klasse.* |
| Opdracht\* | Wat is het verschil tussen een Child Class en een Parent Class?  *De child klasse inherit alle openbare en beschermde eigenschappen en methoden van de parent klasse. Bovendien kan het zijn eigen eigenschappen en methoden hebben.* |
| Opdracht\* | Welke keyword wordt gebruikt om aan te geven dat er Inheritance plaatsvindt? Wat is de basis Syntax van Inheritance in zijn geheel?  Een inherited klasse wordt gedefinieerd met behulp van het keyword extends.  <?php  class A {     public int $prop; } class B extends A {     // Illegal: read-write -> readonly     public readonly int $prop; } ?> |
| Opdracht\* | Hoe kan je Inherited Methods Overridden? Wat is het nu hiervan denk je?  *public methods.* |
| Opdracht\* | Waar wordt de Final keyword voor gebruikt?  *Het keyword final voorkomt dat child klassen een methode of constante overschrijven door de definitie te laten voorafgaan door final. Als de klasse zelf definitief wordt gedefinieerd, kan deze niet worden verlengd.* |

## **Feedbackmoment met de tech coach**

|  |
| --- |
| Als je deze opdrachten hebt uitgevoerd, stuur dan een bericht naar jouw tech coach om mee te kijken met de uitgevoerde opdrachten. |

## **Opdracht: 1.6 - Class Constants**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools [Class Constants in PHP OOP →](https://www.w3schools.com/php/php_oop_constants.asp) |
| Opdracht\* | Wat is het nut van een Class Constant?  *Klasseconstanten kunnen handig zijn als u enkele constante gegevens binnen een klasse moet definiëren.* |
| Opdracht\* | Hoe declareer je een Class Constant?  *Een klasse Constant wordt binnen een klasse gedeclareerd met het keyword const.* |
| Opdracht\* | Hoe verschilt een Class Constant van een Class Property?  *Class constants are case-sensitive.* |
| Opdracht\* | Hoe access je een Class Constant van buiten de Class?  *door de naam van de* Class *te gebruiken, gevolgd door de scope-resolutie-operator (::) gevolgd door de naam van de Constant.* |
| Opdracht\* | Hoe access je een Class Constant van binnen de Class?  *door het* keyword *self te gebruiken, gevolgd door de scope-resolutie-operator (::) gevolgd door de* Constant *naam.* |

## **Opdracht: 1.7 - Static Methods**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over [Static Methods in PHP OOP →](https://www.w3schools.com/php/php_oop_static_methods.asp) |
| Opdracht\* | Wat is het verschil tussen een Static Method en een reguliere Class Method?  Static *betekent dat u niet hoeft te instantiëren (een objectreferentie declareren).* |
| Opdracht\* | Hoe ziet de Syntax van een Static Method eruit?  *Een* Static Method *is toegankelijk via een methode in dezelfde klasse met behulp van het* keyword *self en een dubbele punt (::):* |
| Opdracht\* | Hoe gebruik je de Static Method?  *Om een* Static Method *vanuit een onderliggende klasse aan te roepen, gebruikt u het* keyword *parent in de onderliggende klasse.* |

## **Opdracht: 1.8 - Static Properties**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over [Static Properties in PHP OOP →](https://www.w3schools.com/php/php_oop_static_properties.asp) |
| Opdracht\* | Wat is het verschil tussen een Static Property en een reguliere Property?  Static properties *kunnen direct worden aangeroepen - zonder een instantie van een* class *te maken.* |
| Opdracht\* | Hoe declareer je een Static Property?  with the static *keyword*. |
| Opdracht\* | Hoe gebruik je een Static Property **binnen** een Class?  *met behulp van het* self *keyword.* |
| Opdracht\* | Hoe gebruik je een Static Property **buiten** een Class?  *met behulp van het* parent *keyword*. |

## **Feedbackmoment met de tech coach**

|  |
| --- |
| Als je deze opdrachten hebt uitgevoerd, stuur dan een bericht naar jouw tech coach om mee te kijken met de uitgevoerde opdrachten. Je gaat nu met je tech coach bespreken wat je verder nog wilt leren over de PHP fundamentals, samen maak je hiervoor een plan. |

## 

## **Niveau 2 - Exercising the Fundamentals in PHP.**

De volgende exercises kunnen uitgevoerd worden om de basis concepten daadwerkelijk te oefenen.

|  |  |
| --- | --- |
| Opdracht\* | Voer de volgende [exercises →](#_vt5pvdrznv0) uit. |
| Opdracht\* | Kies een [eindopdracht →](#_8xk5adinwkgh) uit. |
| Optioneel | Wil je meer oefenen, vraag je tech coach om een opdracht. |

## **Niveau 3 - Project Specifieke Kennis**

De volgende onderwerpen zullen terugkomen bij het maken van de webshop. Het zijn ook cruciale onderdelen om te snappen voor het ontwikkelen van moderne web applicaties.

Dit onderdeel gaat niet getoetst worden

**Advanced Concepten die in Laravel terug gaan komen:**

1. PHP OOP - Namespaces
2. PHP OOP - Abstract Classes
3. PHP OOP - Interfaces
4. PHP OOP - Traits

**Further Reading?**

|  |
| --- |
| Check the [Object Oriented Laracast Course →](https://laracasts.com/series/object-oriented-principles-in-php) |

|  |
| --- |
| BLOK 3  DATA INTERACTION  WITH SQL AND  DATABASE MANAGEMENT  IN MYSQL. |

## 

# 

|  |
| --- |
| VOORAF |

## 

## **HET DOEL VAN BLOK 3**

In blok 2 hebben we een introductie gemaakt met Object Oriented Programming wat het makkelijker en efficiënter maakt om code met een heldere en overzichtelijke structuur te schrijven.

In de basis zijn alle opdrachten in de praktijk gebaseerd op het **managen**, **ophalen** en **laten zien** van data op een specifieke manier. Dit wordt gedaan door het gebruiken van de taal SQL (Structured Query Language) in een relationele database management systeem (o.a MYSQL).

In blok 3 gaan we aan de slag met het leren van de basics van SQL en gaan we als MYSQL als relationele database management systeem gebruiken omdat het open-source, gratis en wordt gebruikt door een flink aantal grote partijen in de praktijk zoals facebook, twitter, github en youtube.

## 

## **OVERZICHT LESSTOF VOOR BLOK 3**

De volgende concepten moeten **minimaal** beheerst zijn na blok 3.

1. Syntax
2. Select
3. Select Distinct
4. Where
5. And, Or, Not
6. Order By
7. Insert Into
8. Null Values
9. Update
10. Delete
11. Create DB
12. Drop DB
13. Create Table
14. Drop Table
15. Alter Table
16. Constraints
17. Not Null
18. Unique
19. Primary Key
20. Foreign Key
21. Check
22. Default
23. Index
24. Auto Increment
25. Data Types

## **DE OPDRACHTEN**

De opdrachten dienen op de volgende manier uitgevoerd te worden:

|  |  |  |
| --- | --- | --- |
| Niveau 1 | Minimale Fundamentals | Verplicht |
| Niveau 2 | Exercising SQL and MYSQL | Verplicht |
| Niveau 3 | Digging Deeper into SQL | Optioneel |

## **OVERZICHT LESSTOF**

Voor blok 2 wordt er wederom gebruik gemaakt van w3schools als basis en Laracasts als aanvullende lesstof.

* W3Schools: SQL Tutorial
* Laracasts: MySQL Database Design

|  |
| --- |
| OPDRACHTEN BLOK 3 |

Hieronder staan de opdrachten en lesstof beschreven.

Het kijken van de video’s of lezen van de teksten is geen doel op zich.

## **Reminder!**

**Probeer je steeds te focussen op het echt snappen van de onderstaande concepten voordat je dieper op de stof ingaat of verder werkt anders kan dat heel verwarrend werken.** **Alle toekomstige kennis in het programmeren wordt voortgebouwd op deze fundamentele principes. Begrijp je dit niet voldoende zal het lastig zijn om toekomstige kennis op te doen.**

## **Niveau 1 - Practicing SQL Statements**

Lees op W3Schools de volgende onderwerpen en oefen ze in [PHPmyadmin →](http://localhost/phpmyadmin/)

Deze opdracht zou je niet meer dan een dag moeten kosten om uit te voeren.

Het is belangrijk om alles een keer gedaan te hebben maar er niet te veel tijd aan kwijt te zijn. We gaan vanaf scratch oefenen in niveau 2.

|  |  |
| --- | --- |
| Opdracht\* | Je kunt [hier →](https://drive.google.com/file/d/1qJAPTQMfhX7S0fOORwAOfwrFfeNXup20/view?usp=sharing)  een dummy database downloaden om mee te oefenen.   1. Maak in PHPmyadmin een nieuwe database aan genaamd “testdb”. 2. Download de dummy database via de bovenstaande link. 3. Klik op de nieuw aangemaakte database en vervolgens op de importeer knop 4. Volg de instructies om de database te importeren. |
| Opdracht\* | Lees de volgende onderwerpen op W3Schools en oefen de statements in PHPmyadmin waar mogelijk. Klik op de nieuw aangemaakte database in PHPmyadmin. Via het “SQL” tabje kun je jouw statements testen op jouw dummy database.   * [SQL HOME](https://www.w3schools.com/sql/default.asp) * [SQL Intro](https://www.w3schools.com/sql/sql_intro.asp) * [SQL Syntax](https://www.w3schools.com/sql/sql_syntax.asp) * [SQL Select](https://www.w3schools.com/sql/sql_select.asp) * [SQL Select Distinct](https://www.w3schools.com/sql/sql_distinct.asp) * [SQL Where](https://www.w3schools.com/sql/sql_where.asp) * [SQL And, Or, Not](https://www.w3schools.com/sql/sql_and_or.asp) * [SQL Order By](https://www.w3schools.com/sql/sql_orderby.asp) * [SQL Insert Into](https://www.w3schools.com/sql/sql_insert.asp) * [SQL Null Values](https://www.w3schools.com/sql/sql_null_values.asp) * [SQL Update](https://www.w3schools.com/sql/sql_update.asp) * [SQL Delete](https://www.w3schools.com/sql/sql_delete.asp) * [SQL Select Top](https://www.w3schools.com/sql/sql_top.asp) * [SQL Min and Max](https://www.w3schools.com/sql/sql_min_max.asp) * [SQL Count, Avg, Sum](https://www.w3schools.com/sql/sql_count_avg_sum.asp) * [SQL Like](https://www.w3schools.com/sql/sql_like.asp) * [SQL Wildcards](https://www.w3schools.com/sql/sql_wildcards.asp)   Oefen de statements in PHPMYAdmin |
| Opdracht\* | Lees de volgende onderwerpen over het manipuleren van de database en de kolommen.   * [SQL Create DB](https://www.w3schools.com/sql/sql_create_db.asp) * [SQL Drop DB](https://www.w3schools.com/sql/sql_drop_db.asp) * [SQL Backup DB](https://www.w3schools.com/sql/sql_backup_db.asp) * [SQL Create Table](https://www.w3schools.com/sql/sql_create_table.asp) * [SQL Drop Table](https://www.w3schools.com/sql/sql_drop_table.asp) * [SQL Alter Table](https://www.w3schools.com/sql/sql_alter.asp) * [SQL Constraints](https://www.w3schools.com/sql/sql_constraints.asp) * [SQL Not Null](https://www.w3schools.com/sql/sql_notnull.asp) * [SQL Unique](https://www.w3schools.com/sql/sql_unique.asp) * [SQL Primary Key](https://www.w3schools.com/sql/sql_primarykey.asp) * [SQL Foreign Key](https://www.w3schools.com/sql/sql_foreignkey.asp) * [SQL Check](https://www.w3schools.com/sql/sql_check.asp) * [SQL Default](https://www.w3schools.com/sql/sql_default.asp) * [SQL Index](https://www.w3schools.com/sql/sql_create_index.asp) * [SQL Auto Increment](https://www.w3schools.com/sql/sql_autoincrement.asp) * [SQL Dates](https://www.w3schools.com/sql/sql_dates.asp) * [SQL Data Types](https://www.w3schools.com/sql/sql_datatypes.asp) |

## 

## **Niveau 2 - Designing your Database**

In niveau 2 gaan we in snelvlucht de basics van Database Design leren. Hiermee gaan we een database model maken voor een CV Applicatie die we uiteindelijk in Laravel ook daadwerkelijk gaan bouwen.

|  |  |
| --- | --- |
| Informatie | [Presentatie: Introduction to Database Design →](https://docs.google.com/presentation/d/1WSeTffptq04ZrfaUkGR8Y6ZSva8waE_0mHwWKpeYO-o/edit?usp=sharing)  Gebruik de presentatie als referentie voor het maken van de opdracht. Alle nodige kennis voor de opdracht kan je hierin terugvinden. |
| Informatie | [Laravel Database Design Best Practices →](https://webdevetc.com/blog/laravel-naming-conventions/#section_naming-database-tables-in-laravel)  [Github: Laravel Best Practices (Complete) →](https://github.com/alexeymezenin/laravel-best-practices#follow-laravel-naming-conventions)  Bekijk het overzicht van database design best practices.  Deze zijn gebaseerd op de best practices die gebruikt worden in de Laravel community. Dit zal je een solide start geven voor het structureren en benoemen van je tabellen en kolommen.  **Het wordt van je verwacht dat je deze hanteert bij het maken van de opdracht.** |
| Opdracht\* | Lees de[Projectbeschrijving: CV Applicatie →](#_r62q5mwamfkf)  We gaan het ontwikkelen van een datamodel oefenen zoals dit in de praktijk ook gedaan wordt aan de hand van de bovenstaande projectbeschrijving. |
| Opdracht\* | Maak een **Entity Relationship Diagram** voor je CV Applicatie op basis van de bovenstaande projectbeschrijving.  Aanpak:   1. Maak op basis van het kopje databeschrijving in de projectbeschrijving de tabellen en kolommen die je nodig bent voor je applicatie. Laat de relaties tussen de tabellen eerst achterwege. 2. Lees in de resources of de presentatie over welke soorten relaties bestaan en leg de relaties tussen je tabellen waar nodig. Vraag je techcoach om feedback als je twijfelt.   Resources:  [Voorbeeld Entity Relationship Diagram →](https://www.metabase.com/learn/images/table-relationships/sample-database-erd.png)  [Wat is een Entity Relationship Diagrams →](https://www.metabase.com/glossary/erd)  [Informatie over Table Relationships →](https://www.metabase.com/learn/databases/table-relationships)  [Je kunt deze app gebruiken om je ERD in uit te werken →](https://app.diagrams.net/)  Links in het diagrams menu kun je onder het kopje “Entity Relation” een tabel op je canvas slepen.  Je mag de opdracht ook op papier uitwerken.  **Deze moet goedgekeurd worden door je tech coach.** |
| Opdracht\* | Maak in PHPmyadmin volgens het opgestelde model de database en nodige kolommen aan. |
| Opdracht  (Optioneel) | Vul door het schrijven van SQL statements je aangemaakte database met records.  Requirements:   * Voeg tenminste 1 persoon toe aan jouw database. * Voeg tenminste 3 hobbies toe voor die persoon. |
| Opdracht  (Optioneel) | Schrijf een SQL statement die je toegevoegde persoon en zijn gerelateerde hobbies ophaalt. |

## 

|  |
| --- |
| BLOK 4  LARAVEL ALS  FULL STACK  FRAMEWORK. |

## 

# 

|  |
| --- |
| VOORAF |

## **HET DOEL VAN BLOK 4**

In blok 3 hebben we een aantal dingen geleerd. We hebben geleerd hoe je vanuit de projectbeschrijving van een klant een datamodel kan maken. Vervolgens hoe je met een relationele database management systeem een database, bijbehorende tabellen en kolommen op basis van het gemaakte datamodel volgens best practices in het veld aan kunt maken. Ten slotte hebben we gezien hoe interactie met de database zoals inserten van records met de database taal SQL gedaan kan worden.

In blok 4 gaan we onze eerste echte aanraking komen met een moderne manier van web development in het framework Laravel. We gaan Laravel gebruiken als Full Stack framework om een moderne applicatie te voorzien van zowel een front-end richting de gebruikers als een backend waar een admin nieuwe dingen in kan toevoegen.

We gaan de CV Applicatie waar we een datamodel van hebben gemaakt in het vorige blok uitwerken tot een volledig werkende applicatie. Het project zal opgedeeld worden in een aantal fases die je kan gebruiken als general guidelines in hoe je de projectontwikkeling aan kan pakken.

Het is belangrijk dat je een keer gezien hebt hoe interactie met de database werkt onder de kap. Laravel is een framework dat by design een boel functionaliteiten ingebouwd heeft om taken **makkelijker** uit te kunnen voeren. Onthoud dat wat er onder de kap van de motor is verstopt hetgeen is wat we in de vorige blokken geleerd hebben.

## 

## **OVERZICHT LESSTOF VOOR BLOK 4**

De volgende concepten moeten **minimaal** beheerst zijn na blok 4.

[Laravel Basics (Documentation) →](https://laravel.com/docs/9.x/)

1. Getting Started Section →
   * Installation
   * Configuration
   * Directory Structure
   * Starter Kits
2. The Basics Section →
   * Routing
   * Middleware
   * CSRF Protection
   * Controllers
   * Requests
   * Responses
   * Views
   * Blade Templates
   * URL Generation
   * Session
   * Validation
   * Error Handling
   * Logging

## **DE OPDRACHTEN**

De opdrachten dienen op de volgende manier uitgevoerd te worden:

|  |  |  |
| --- | --- | --- |
| Niveau 1 | Fundamentals in Laravel | Verplicht |

## **OVERZICHT LESSTOF**

Voor blok 4 gebruiken we voornamelijk de documentatie op de Laravel website en Laracasts als informatiebron.

* [Laracasts: Laravel 8 From Scratch →](https://laracasts.com/series/laravel-8-from-scratch)
* [Laravel Docs →](https://laravel.com/docs/9.x)

|  |
| --- |
| OPDRACHTEN BLOK 4 |

Hieronder staan de opdrachten en lesstof beschreven.

Het kijken van de video’s of lezen van de teksten is geen doel op zich.

## **Reminder!**

**Probeer je steeds te focussen op het echt snappen van de onderstaande concepten voordat je dieper op de stof ingaat of verder werkt anders kan dat heel verwarrend werken.** **Alle toekomstige kennis in het programmeren wordt voortgebouwd op deze fundamentele principes. Begrijp je dit niet voldoende zal het lastig zijn om toekomstige kennis op te doen.**

## **Niveau 1 - Fundamentals in Laravel**

De focus van dit niveau ligt op het snappen van de **fundamentals van het Laravel Framework.** We gaan deze fundamentals leren door het maken van een werkende applicatie voor een klant.

### Focus

De focus ligt op de volgende elementen:

* Het opzetten van een nieuw Laravel project.
* Het leren van de basic workflow door een klein stukje van je app te ontwikkelen en je eerste gegevens op een scherm te laten zien.
* Het manipuleren van gegevens door middel van CRUD (Create, Read, Update, Delete) acties in Laravel.
* Het leggen van relaties tussen modellen en deze gebruiken om gerelateerde gegevens op te kunnen halen en weer te kunnen geven.

### Project Fases

Hier even een kort overzicht van de fasen die we door gaan lopen om dit te realiseren:

* Fase 1: Project Setup
* Fase 2: The Basic Workflow
* Fase 3: CRUD Actions
* Fase 4: Model Relations

### 

### **Inleidend**

Hier even kort ter **herhaling** wat je moet weten voordat je begint met het daadwerkelijk ontwikkelen.

Alle projecten beginnen met een **projectbeschrijving.** Hierin wordt duidelijk ingekaderd wat de applicatie die je gaat bouwen moet gaan doen. Zo weet je precies wat je gaat bouwen voordat je begint en kom je niet in de knoei in een later stadium en je hele applicatie om moet gaan bouwen.

Naast een projectbeschrijving is het ook belangrijk om te weten HOE de applicatie eruit komt te zien. Door een simpel **wireframe** te maken maak je inzichtelijk voor jou en de klant wat er wordt weergegeven en op welke plek dit moet komen te staan.

Als je project geen databeschrijving heeft moet je uit de beschrijving en wireframe een datamodel maken en deze toetsen met je klant of deze volledig is. Als er een databeschrijving is gemaakt zoals in deze opdracht vertaal je deze naar een **datamodel** zoals we gedaan hebben in de vorige opdracht.

Dit geeft inzicht in hoe je gegevens aan elkaar relateren en je dus je applicatie in moet gaan richten en structureren voordat je ook maar een regel code geschreven hebt. Dat scheelt.

## 

## **Fase 1 - Setup your Project**

Alle projecten beginnen met het opzetten en configureren van een nieuw Laravel project. Om te kunnen ontwikkelen moeten we een nieuw project starten.

De volgende basisonderwerpen gaan we behandelen:

* Het **aanmaken** en **draaien** van een nieuw Laravel project op onze device.
* Hoe gebruiken we de .env file om de basisconfiguratie van ons project in te richten.
* Hoe ziet de mappenstructuur van een Laravel project in elkaar.
* Hoe kunnen we Artisan Commands gebruiken om generieke taken makkelijker kunnen maken en tijd te besparen.
* Het aanmaken van de database die we nodig zijn voor ons project.

## **Opdracht: 1.0 - Installation & Configuration**

|  |  |
| --- | --- |
| Opdracht\* | Laravel kan op twee manieren geïnstalleerd worden:   * Docker (complexer, alleen doen met ervaring) * Composer (simpel, aan te raden)   Als je eerder hebt gewerkt met technieken zoals docker en wsl kun je proberen om daarmee je project op te zetten.  **Voor de beginner!**  Bij twijfel gebruik de simpelste manier om Laravel te installeren. Via composer.  [Lees uitgebreide Laravel installatie instructies →](#_xdxlqojvwh29)  Je kunt de volgende pagina op de officiële documentatie ter referentie gebruiken: [Laravel Installation →](https://laravel.com/docs/9.x/installation) |
| Opdracht\* | Lees hier over hoe het [configureren](https://laravel.com/docs/9.x/configuration) van je project →  Lees het kopje **introduction** en **environment** configuration.   1. Zorg dat je weet op welke twee manieren Laravel geconfigureerd kan worden. 2. Voeg in je .env file de juiste database gegevens in zodat Laravel kan communiceren met je database. 3. Optioneel: Check of in je .env file je app key is toegevoegd. 4. Optioneel: Check of in je .env file je de debug value of true is. |
| Opdracht\* | Lees hier over hoe de [mappenstructuur van Laravel →](https://laravel.com/docs/9.x/structure)  Scan de introductie van deze pagina voor een overview van de directory structure.  Zorg ervoor dat je de volgende mappen kan vinden die ga je in de volgende opdrachten nodig hebben:   * Root/App/HTTP/Controllers * Root/App/Models * Database/Factories * Database/Migrations * Database/Seeders * Routes/Web * Public * Resource/Views |
| Opdracht\* | Lees hier over de functie van [Artisan Commands →](https://laravel.com/docs/9.x/artisan#introduction)  Bekijk de lijst van beschikbare artisan commands door volgende commando uit te voeren in je terminal:   * php artisan list   Optioneel: Bekijk de volgende link met [21 Artisan Make Commands For File Generation →](https://blog.quickadminpanel.com/list-of-21-artisan-make-commands-with-parameters/) |
| Opdracht \* | Maak in PHPmyadmin een nieuwe database aan met de naam “**app**”. Je mag natuurlijk ook een eigen naam kiezen voor je database. |
| Opdracht  (Optioneel) | Lees kort over [starter kits in Laravel →](https://laravel.com/docs/9.x/starter-kits)  Dit kan je in de toekomst helpen met een snellere start. We gaan hier huidig geen gebruik van maken. |

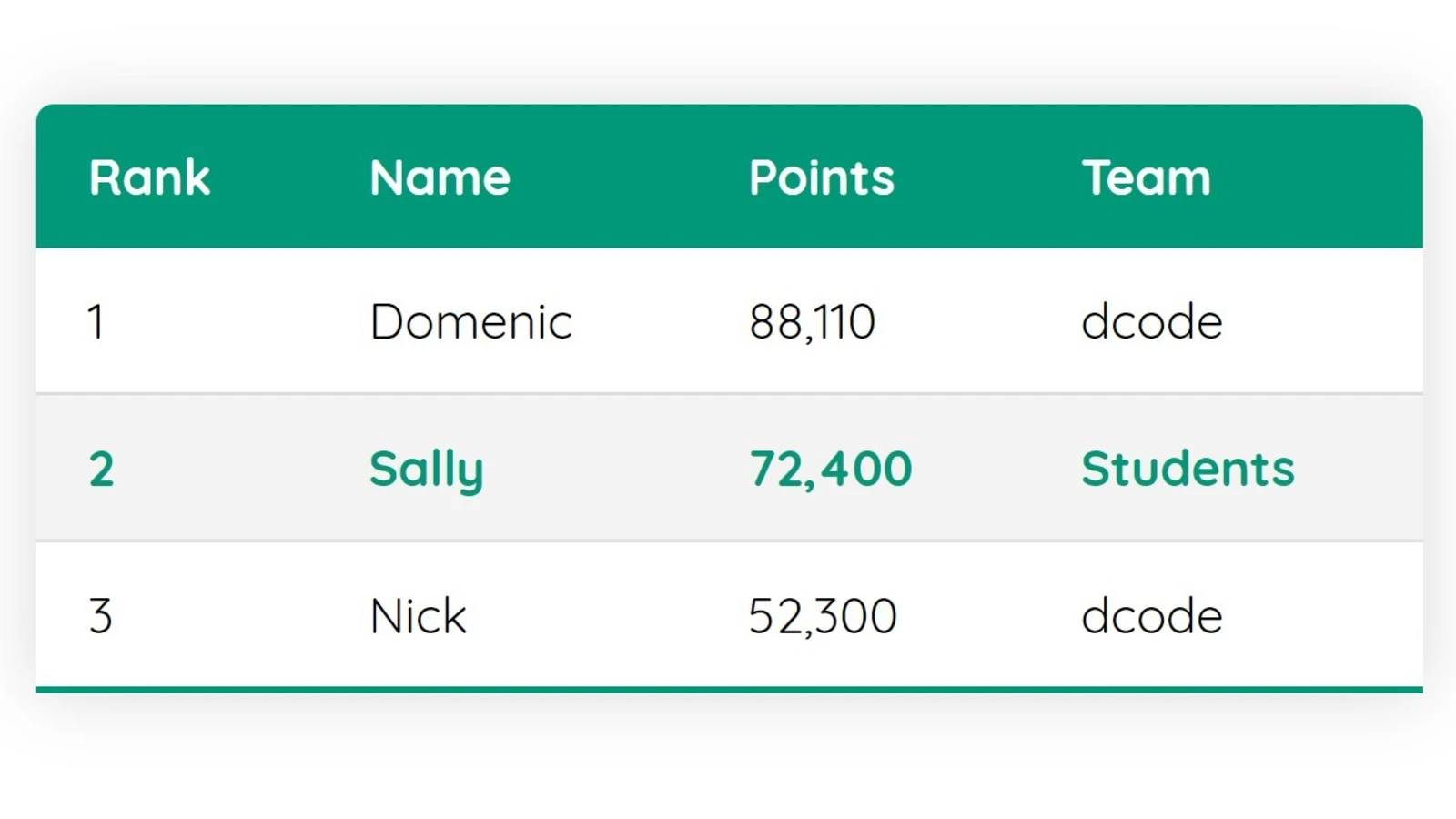
## 

## **Fase 2 - The Basic Workflow**

Het doel van deze fase is om aan het einde de complete workflow van Laravel op een simpele manier door te lopen. Aan het einde van deze fase kun je alle **personen** tonen in een simpel tabel overzicht.

Aan het einde van deze fase hebben we een scherm met de volgende output:

Een tabel met daarin een lijst van persoonsgegevens in een net overzicht.



## 

## **Overzicht Ontwikkelstappen**

Deze fase bestaat uit de volgende stappen:

1. Je tabellen opbouwen op basis van je datamodel met **Migrations**.
2. Dummy data toevoegen aan je database met **Seeds**.
3. Dummy data genereren door **Factories**.
4. Voor elke tabel in je database een **Model** aanmaken die gebruikt kan worden voor de interactie met die tabel vanuit je applicatie.
5. Entry points voor je applicatie maken (voor je gebruiker) door middel van **Routes**.
6. In je **Controller** je request handling logica schrijven die de nodige data ophaalt, format en doorgeeft aan de bijbehorende view.
7. In de **View** je data weergeven en stijlen naar wens.

## **Opdracht: 1.1 - Migrations**

We starten onze development workflow altijd bij het vertalen van ons datamodel naar migrations. Als het goed is heb je op dit punt in je ontwikkeling een **lege** database aangemaakt via phpmyadmin of tinker.

Migrations worden gebruikt om de structuur van je database te managen vanuit de code. Je hebt in de mysql opdrachten handmatig tabellen aan moeten maken in phpmyadmin. Dit is een tijdrovend proces en kan makkelijker. Elke migration is verantwoordelijk voor het maken van een **tabel** en de **kolommen** die nodig zijn voor die tabel.

In de migration files schrijf je een **definitie** van de tabellen die je nodig hebt voor je applicatie. Dit betekend **niet** dat het aanmaken en of aanpassen van de tabellen ook daadwerkelijk wordt uitgevoerd zodra je de file aan hebt gemaakt.

Je gebruikt Artisan commands om je migrations uit te voeren en zo je databasetabellen op en af te bouwen naar wens. Ook kun je via Artisan commands een nieuwe migration file in de map genereren.

In de volgende stappen gaan we daarmee oefenen.

|  |  |
| --- | --- |
| Pre Requisites | * Have an empty database created. * Have the necessary database credentials added to your .env file. |
| Informatie  (Extra) | Lees op de Laravel website wat de functie van een [**migrations →**](https://laravel.com/docs/9.x/migrations#introduction)is. |
| Stap 1 | Voer in je terminal de volgende commando uit:   * php artisan make:migration create\_persons\_table   Hiermee maak je een nieuwe migration file aan en gebruik je de juiste naamgeving. Deze file is huidig nog leeg maar zou voorgedefinieerde moeten zijn met een up() en down() method.  Expected Output:    Expected Output:  Je migration staat nu in de migrations map. |
| Stap 2 | De volgende stap in het proces is het definiëren van welke tabel en kolommen aangemaakt of afgebouwd moeten worden met het overnemen van je datamodel en een schema definitie maken in de up() method.  Hierin geef je aan wat er moet gebeuren als de migrate commands uitgevoerd moeten worden. Denk goed na over welke types kolommen je nodig bent en of deze leeg (nullable) mogen zijn of niet.  Expected Output: |
| Opdracht\*  (required) | Voer in je terminal de volgende commando uit:   * php artisan migrate   Expected output: |
| Opdracht  (Optioneel) | Voer in je terminal ook eens de volgende commando's uit:   * php artisan migrate:refresh (verwijderd al je tabellen en bouwt ze weer op) * php artisan migrate:rollback (verwijderd al je tabellen) |

## 

## **Opdracht: 1.2 - Seeders**

Het is lastig om je applicatie te testen als je niets hebt om te laten zien op je scherm. Het kost doorgaans ontzettend veel tijd om handmatig data toe te voegen aan je applicatie. Daar wil je als developer zo min mogelijk mee bezig zijn.

Seeders worden gebruikt om je app te voorzien van dummy data. In de vorige opdracht heb je jouw databasetabellen opgebouwd door middel van migrations. We gaan in deze opdracht de person's tabel vullen met dummy data zodat we dat uiteindelijk kunnen laten zien in een overzicht aan het einde van deze fase.

|  |  |
| --- | --- |
| Informatie | Lees op de Laravel website wat de functie van een [**seeder →**](https://laravel.com/docs/9.x/seeding#introduction)is. |
| Opdracht (required) | Leg via slack in eigen woorden uit wat de functie van een **seeder** is. |
| Opdracht  (required) | Voer in je terminal de volgende commando uit:   * php artisan make:seeder PersonSeeder   Hiermee maak je een nieuwe lege seeder file aan voor je persons table.  Expected Output: |
| Opdracht  (required) | Vul de seeder met **3 dummy records**.  De volgende stap is het vullen van je seeder file met dummy data zodat als je de seed commando uitvoert  Begin hiermee. Op de volgende manier kan je hardcoded data toevoegen.    Op de volgende manier kan je het semi randomized doen. Het scheelt wat typwerk maar geeft niet heel bruikbare data om mee te testen.    De onderstaande manier is de meest bruikbare manier om dummy data toe te voegen. Door het gebruik van de [Faker Library →](https://github.com/fzaninotto/Faker#installation) Deze kan menselijk leesbare dummy data voor je genereren.  Begin met het toevoegen van de bovenstaande code:    Gebruik de beschikbare functies en faker properties om dummy data te genereren. Bekijk op de github pagina welke mogelijkheden er allemaal zijn. |
| Opdracht  (required) | Call de **PersonSeeder** in je **DatabaseSeeder.php** file.    Door je seeder classes in een array mee te sturen als argument kan je meerder seeders tegelijk callen. |
| Opdracht  (required) | Voer in je terminal de volgende commando uit:   * php artisan db:seed   Laravel runt als je dit commando uitvoert de **run()** function in je **DatabaseSeeder.php** file. Zo kun je al je seeders tegelijk uitvoeren.  Bekijk als je het commando hebt uitgevoerd je database. Je zou nu dummy data terug moeten kunnen vinden.  Expected Output: |
| Opdracht  (Optioneel) | Voer in je terminal de volgende commando uit:   * php artisan migrate:fresh --seed   Bekijk hoe laravel al je migraties terugrolt en je database leeggooit. Vervolgens je tabellen weer opbouwt en je seeders uitvoert en zo je database weer vult. Dit is handig als je een schone lei wil hebben nadat je test data hebt toegevoegd aan je project.  Expected Output |

## 

## **Opdracht: 1.4 - Models**

Elke tabel in je database heeft een bijbehorende model class dat zorgt voor de interactie met die tabel vanuit je code. Je kunt niet alleen records uit je databasetabel halen maar je kunt ook nieuwe records inserten, updaten en deleten uit de tabel.

We gaan voor onze persons table een

|  |  |
| --- | --- |
| Informatie | Lees op de Laravel website wat de functie van een [**model →**](https://laravel.com/docs/9.x/eloquent#introduction)is.  **Belangrijk! Lees de sections:**   * Introduction * Table Names * Primary Keys * Timestamps |
| Opdracht\* | Leg via slack in eigen woorden uit wat de functie van een model is. |
| Opdracht\* | Voer de volgende commando uit in je terminal:   * php artisan make:model Person |

## **Opdracht: 1.3 - Factories (Optioneel)**

|  |  |
| --- | --- |
| Informatie | Lees op de Laravel website wat de functie van een [**Factory →**](https://laravel.com/docs/9.x/database-testing#introduction)is. |
| Opdracht | Leg via slack in eigen woorden uit wat de functie van een **Factory** is. |
| Opdracht | Gebruik de **php artisan make:factory** command om een **PersonFactory**  aan te maken. |
| Opdracht | Gebruik je factory om **ten minste 10 dummy records** voor je persons table mee aan te maken. |
| Opdracht | Gebruik de **php artisan migrate:fresh -- seed** command om je migrations opnieuw te runnen met je factories. |

## 

## **Opdracht: 1.5 - Routing**

|  |  |
| --- | --- |
| Informatie | Lees op de Laravel website wat de functie van een [**route →**](https://laravel.com/docs/9.x/routing#basic-routing)is.  Sections: **#Basic Routing** en **#The Default Route** Files. |
| Opdracht\* | Leg via slack in eigen woorden uit wat de functie van een **route** is. |
|  | Maak de index **route** (volgens conventie) voor je project die wijst naar je PersonController. |
| Optioneel | Lees hier voor meer achtergrondinformatie over de [Laravel Request Lifecycle →](https://laravel.com/docs/9.x/lifecycle) zodat je weet wat er onder de “motorkap” gebeurd als je dat wil. |

#### **Opdracht: 1.6 - Controllers**

|  |  |
| --- | --- |
| Informatie | Lees op de Laravel website wat de functie van een [**controller →**](https://laravel.com/docs/9.x/controllers#introduction)is.  Sections:  **#Introduction** en **#Basic Controllers.**  **Ga terug naar model sections en lees:**   * Retrieving Models |
| Opdracht\* | Leg via slack in eigen woorden uit wat de functie van een **controller** is. |
| Opdracht\* | Gebruik de **php artisan make:controller --resource** command om je **PersonController** mee aan te maken. De --resource flag maakt automatisch je controller functies aan. |
| Opdracht\* | Schrijf in je **index()** functie de logica die je nodig hebt om **alle** gegevens uit je **persons** table te halen. Gebruik hiervoor je **Person** model. |
| Opdracht\* | Pass deze data door naar je **person.index** view. (Als je dit runt zul je een error krijgen want deze view file bestaat nog niet. Dat doen we in de volgende oefening.) |

#### 

#### **Opdracht: 1.7 - Views**

|  |  |
| --- | --- |
| Informatie | Lees op de Laravel website wat de functie van een [**view →**](https://laravel.com/docs/9.x/eloquent#introduction)is. |
| Opdracht\* | Leg via slack in eigen woorden uit wat de functie van een **view** is. |
| Opdracht\* | Maak in je **resources → views** map een **person** map aan. Maak hierin ook een **index.blade.php** file aan. |
| Informatie | Lees op de Laravel website over [**Blade Templates & Functions →**](https://laravel.com/docs/9.x/blade) |
| Opdracht\* | Gebruik een **blade foreach loop** om je meegestuurde data te renderen op je pagina in een net [tabel →](https://www.w3schools.com/html/html_tables.asp) overzicht. |
| **Opdracht**  **(optioneel)** | Maak een **master.blade.php** view file aan en extend deze in je persons.index view file. |
| **Opdracht**  **(optioneel)** | Voeg een section toe voor styling en script partials in je master en extend deze sections in je persons.index view file. |

### **Fase 3 - CRUD actions.**

[Laravel CRUD Actions Tutorial →](https://www.itsolutionstuff.com/post/laravel-9-crud-application-tutorial-exampleexample.html)

Hier komt een beschrijving van de basic CRUD actions in Laravel op basis van de Controller Functions:

* show
* create
* store
* edit
* update
* delete

### **Fase 4 - Model Relations**

De basic workflow heb je nu gehad. Je zou nu voldoende informatie moeten hebben over hoe de basic workflow van Laravel in elkaar zit. De volgende stap is het toevoegen van meer complexiteit om geavanceerdere functionaliteit voor elkaar te krijgen. Dit gaan we doen aan de hand van model relations.

[Laravel Model Relationships Tutorial →](https://www.itsolutionstuff.com/post/laravel-eloquent-relationships-tutorial-from-scratchexample.html)

[Laravel Model Relationships Documentation →](https://laravel.com/docs/9.x/eloquent-relationships)

|  |
| --- |
| BLOK 5  LARAVEL ALS API BACKEND |

## 

# 

|  |
| --- |
| VOORAF |

## **HET DOEL VAN BLOK 5**

In blok 4 zijn we in aanraking gekomen met een moderne manier van web development. We hebben de basic workflow van Laravel leren kennen door het ontwikkelen van een CV applicatie. Laravel is in blok 4 hoofdzakelijk gebruikt als full stack framework. We hebben een voor- en achterkant ontwikkeld voor onze applicatie binnen het Laravel Framework.

In blok 5 gaan we aan de slag met ons eindproject. Het ontwikkelen van een webshop. Deze webshop zal aan het einde van de bootcamp gepresenteerd worden aan alle andere studenten en het team van Codegorilla als afsluiting van de bootcamp.

In dit blok krijg je ook de mogelijkheid om wat je hebt geleerd in de basecamp te verenigen met wat je hebt geleerd in de bootcamp. Het is de bedoeling om afsluitend kennis te maken met het ontwikkelen van een webshop waarin Laravel als API backend gebruikt zal worden. Angular zal ingezet worden om het front-end van je shop van je webshop mee te ontwikkelen.

We gaan gebruikmaken van dezelfde werkwijze als het ontwikkelen van de CV applicatie. Je voert dit blok uit in tweetallen. We gaan starten met het definiëren van een wireframe, functiebeschrijving en datamodel voor onze webshop. Daarna gaan we samen een github repository opzetten en hier gezamenlijk in ontwikkelen. Aan het einde van het blok gaan julie gezamenlijk je project presenteren.

## **OVERZICHT LESSTOF VOOR BLOK 5**

De volgende concepten moeten **minimaal** beheerst zijn na blok 5.

* Het ontwikkelen van een moderne web app van scratch in Laravel en Angular.
* Het leren samenwerken met collaborators in GIT
* Het leren samen ontwikkelen en verdelen van de ontwikkeling van een applicatie.
* Het plannen van de ontwikkeling van een applicatie.

## **EINDOPDRACHT: WEBSHOP IN LARAVEL**

In blok 5 gaan we bezig met het ontwikkelen van een webshop in Laravel & Angular volgens specificaties. Hierbij gaan we gebruik maken van de workflow die we de afgelopen maanden hebben geleerd.

## **OVERZICHT LESSTOF**

Voor blok 5 gebruiken we voornamelijk de documentatie op de Laravel website en Laracasts als informatiebron.

* [Laracasts: Laravel 8 From Scratch →](https://laracasts.com/series/laravel-8-from-scratch)
* [Laravel Docs →](https://laravel.com/docs/9.x)

|  |
| --- |
| EINDOPDRACHT:  WEBSHOP IN LARAVEL |

## 

## **Fase 1 - Project Research & Scope Definition**

|  |  |
| --- | --- |
| Informatie | Elk project begint met het doen van onderzoek en het bepalen van de scope van de opdracht die (voor een klant) gemaakt moet worden.  We doorlopen de volgende fasen:   1. Het lezen van de projectbeschrijving. 2. Het definiëren van de soort shop dat we gaan bouwen. 3. Het maken van een wireframe. 4. Het maken van een datamodel.   Hiermee hebben we aan het einde een duidelijk ingekaderd project die we kunnen ontwikkelen. |
| Opdracht 1 | **Oriënteer je op project door de projectbeschrijving te lezen:**     * [Eindopdracht: Webshop In Laravel →](#_8gquj1f1felh)   We beginnen met ons inlezen in de vraag van de klant door de projectbeschrijving te lezen. Hierin kan de vraag van de klant teruggevonden worden en dus ook wat er uiteindelijk ontwikkeld moet worden.  Het belangrijkste onderwerp is de **eisen (requirements)** en wensen **(nice-to-haves)** van de klant goed scherp te hebben. |
| Opdracht 2 | **Beslis samen met je partner wat voor soort webshop jullie willen ontwikkelen. Communiceer dit met je tech coach.**  Je tech coach kan je helpen knopen doorhakken als je hier moeite mee hebt. **Maak het niet te moeilijk voor jezelf.** Je kunt altijd uitbreiden als je tijd over hebt! |
| Opdracht 3 | **Ontwikkel samen een wireframe voor de webshop.**  [Voorbeeld Wireframe →](https://landing.moqups.com/img/content/wireframes/ecommerce-wireframe-1.png)  Je webshop heeft tot nu toe geen vorm. In de wireframe fase gaan we een vast vorm gegeven aan onze applicatie door het visueel maken van het skelet van onze applicatie in een wireframe.  Je klant spreekt namelijk geen “code” maar heeft wel een behoefte. Hoe kun je weten of de manier waarop jij de requirements van de klant denkt te gaan vervullen past bij wat ze nodig hebben.  Dit kan je polsen door middel van het samen bespreken van de wireframe. Het maken van een wireframe kost weinig tijd geeft ons de mogelijkheid om met de klant te overleggen of onze ideeën over de uitwerking van de eisen en wensen op de juiste manier terugkomen.  Je mag beginnen met het uitwerken van je wireframe op papier. Het is beter om je wireframe in [Draw.io →](https://app.diagrams.net/) uit te werken. Dit is eenvoudiger om te delen en bespreken met je klant en lijkt ook professioneler.  Helemaal mooi is om door middel van applicaties zoals [**Adobe XD** →](https://www.adobe.com/nl/products/xd.html) of [**Invision →**](https://www.invisionapp.com/) een clickable model te maken die je met je klant kan live kan doorstappen. |
| Opdracht 4 | **Ontwikkel samen een datamodel voor de webshop.**  Nu je samen met je klant duidelijk in kaart hebt gebracht hoe de applicatie eruit moet komen te zien (wireframe) kun je beginnen met het plannen van de ontwikkeling van je applicatie.  Op basis van de **requirements** en het **wireframe** ontwikkel je een datamodel voor je applicatie. **Je kan namelijk uit de requirements en de wireframe bedenken welke tabellen en kolommen er nodig zijn in je database.**  Je gaat vooraf nadenken en inzichtelijk maken hoe je data aan elkaar relateert. Dit helpt om structuur in je applicatie en ontwikkeling van je applicatie te krijgen zodat je achteraf niet grote onderdelen moet ombouwen omdat je iets over het hoofd ziet.  Mogelijke tabellen voor de webshop zijn:   * users * products * product\_stock * product\_categories * product\_properties * ratings * orders * order\_products * order\_status |
| Opdracht | Laat je ontwikkelde documenten goedkeuren door je tech coach. |

## **Fase 2 - Application Development**

|  |  |
| --- | --- |
| Informatie | In fase 1 hebben we alle kaders gedefinieerd die we nodig zijn om het ontwikkelen van een project te kunnen starten.  In fase 2 gaan we beginnen met de ontwikkeling van de applicatie. Dit doen we op de volgende manier:   1. Het opzetten van een GIT repository voor de code-base. 2. Het bespreken en evalueren van het ontwikkelplan (wekelijks). 3. Het ontwikkelen op basis van het ontwikkelplan. |
| Opdracht 1 | **Zet een Git Repository voor de codebase van je webshop.**  Tijdens de uitvoering van deze opdracht gaan jullie zoals in de praktijk gebruik maken van een git repository waar jullie in branches werken om je code-base te beheren.   * Maak een nieuwe github project voor de eindopdracht. * Voeg je teamgenoot toe als collaborateur. * Maak een development branch aan waar jullie je werk kunnen samenvoegen. * Maak een branch voor iedere collaborateur zodat jullie individueel je code kunnen uploaden. * Push **dagelijks** jouw werk naar je eigen branch. * **Wekelijks** mergen jullie je voortgang naar de development branch. * Je tech coach kijkt regelmatig mee met dit proces. |
| Opdracht 2 | **Bespreek je ontwikkelplan per week (sprint) met je klant. (tech coach)**  Als je een project ontwikkeld voor een klant zul je de klant op de hoogte moeten stellen van de voortgang van je project. Bepaal wat je op dit moment denkt dat een realistisch en haalbaar ontwikkeldoel is. Pas elke week je verwachting aan naarmate je meer leert over je ontwikkelsnelheid. |
| Opdracht 3 | **Start de ontwikkeling van je webshop.**  Succes! |

## 

## **Bijlagen**

### Blok 1 - Fundamentals in PHP

* Bijlage 1 - Persoonlijk Werkdocument
* Bijlage 2 - Exercises

### Blok 2 - Object Oriented Programming in PHP

* Bijlage 3 - Persoonlijk Werkdocument
* Bijlage 4 - Exercises
* Bijlage 5 - Eindopdracht

### Blok 3 - SQL & MYSQL

* Bijlage 6 - Het Project: CV Applicatie

### Blok 4 - Fundamentals in Laravel

* Bijlage 7 - Composer Installation Instructions

### Blok 5 - Laravel als API Backend

* Bijlage 8 - Projectbeschrijving: Webshop in Laravel

## **Blok 1 - Fundamentals in PHP**

### Bijlage 1 - Persoonlijk Werkdocument Blok 1

|  |  |
| --- | --- |
| Naam Student |  |
| Opleidingsgroep |  |

**Opdracht: 1.1 - Introductie: Wat is PHP?**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools de [PHP Intro →](https://www.w3schools.com/php/php_intro.asp) |
| Opdracht\* | Leg in je eigen woorden uit wat PHP is. |
| Opdracht\* | Leg in je eigen woorden uit wat je kan doen met PHP. |
| Opdracht\* | Leg in je eigen woorden uit wat een PHP file is. |

**Opdracht: 1.2 - Installing and using PHP**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over het [installeren en gebruiken van PHP →](https://www.w3schools.com/php/php_install.asp)  Lees op PHP The Right Way over [getting started with PHP →](https://phptherightway.com/#getting_started) |
| Opdracht\* | Leg in eigen woorden uit wat je nodig hebt om PHP zelf te draaien?  Download en Installeer Xampp op je machine met default settings en php 8.1.  Start je Xampp en zorg dat de Apache module draait (groen).  Click op de “Admin” knop in de Xampp app.  Click op de php info tab en controleer of php8.1 draait.  Click op de “Explorer” knop in de Xampp app en vindt de htdocs map.  Informatie: Uit deze map kan je met Xampp je php files runnen uitvoeren in de browser.  Maak in de htdocs map een “opdrachten” map aan.  Sla in de “opdrachten” map al je php files op!  Maak in de “opdrachten” map een index.php file aan.  Plaats in de file de volgende code: <?php print(“PHP Works!”); ?>  Bezoek in je browser localhost/opdrachten/index.php om het bestand te zien. |

**Opdracht: 1.3 - Syntax**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over [syntax in PHP →](https://www.w3schools.com/php/php_syntax.asp) |
| Opdracht\* | Voer [Syntax Opdrachten 1 t/m 5 →](https://www.w3schools.com/php/exercise.asp?filename=exercise_syntax1) uit. |

**Opdracht: 1.4 - Comments**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over [comments in PHP →](https://www.w3schools.com/php/php_comments.asp) |
| Opdracht\* | Leg in je eigen woorden uit waar comments voor bedoeld zijn? |
| Opdracht\* | Leg in je eigen woorden uit op welke verschillende manieren je comments kan gebruiken. |

**Opdracht: 1.5 - Variables**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over [variables and variables scope in PHP →](https://www.w3schools.com/php/php_variables.asp)  Bekijk [Episode 3: Variables →](https://laracasts.com/series/php-for-beginners/episodes/3) op Laracasts. |
| Opdracht\* | Voer [Syntax Opdrachten 1 t/m 2 →](https://www.w3schools.com/php/exercise.asp?filename=exercise_variables1) uit. |

**Opdracht: 1.6 - Echo en print Statements**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over [echo en print Statements in PHP →](https://www.w3schools.com/php/php_echo_print.asp) |
| Opdracht\* | Leg uit waar “echo” en “print” voor gebruikt worden?  Leg uit wat het verschil tussen “echo” en “print” is? |

**Opdracht: 1.7 - Data Types**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over [PHP Data Types →](https://www.w3schools.com/php/php_datatypes.asp) |
| Opdracht\* | Leg uit wat een String is?  Leg uit wat een Integer is?  Leg uit wat een Float is?  Leg uit wat een Boolean is?  Leg uit wat een Array is?  Leg uit wat NULL is? \*optioneel\*  Leg uit wat een Object is? \*optioneel\* |

**Opdracht: 1.8 - Constants**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over [Constants in PHP →](https://www.w3schools.com/php/php_constants.asp) |
| Opdracht\* | Leg uit wat constants zijn en hoe deze verschillen van variabelen? |

**Opdracht: 1.9 - Operators**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over [Operators in PHP →](https://www.w3schools.com/php/php_operators.asp) |
| Opdracht\* | Voer [Operators Opdrachten 1 t/m 4 →](https://www.w3schools.com/php/exercise.asp?filename=exercise_operators1) uit. |

**Opdracht: 1.10 - If…Else…Elseif.**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over [If…Else…Elseif in PHP →](https://www.w3schools.com/php/php_if_else.asp) |
| Opdracht\* | Voer [If… Else…Elseif Opdrachten 1 t/m 4 →](https://www.w3schools.com/php/exercise.asp?filename=exercise_ifelse1) uit. |

**Opdracht: 1.11 - Loops**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over [Loops in PHP →](https://www.w3schools.com/php/php_looping.asp) |
| Opdracht\* | Leg uit waar loops voor gebruikt worden?  Leg uit welke verschillende soorten loops er zijn en wat hun functie is?  Voer [Loops Opdrachten 1 t/m 4 →](https://www.w3schools.com/php/exercise.asp?filename=exercise_loops4) uit. |

**Opdracht: 1.12 - Functions**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over [Functions in PHP →](https://www.w3schools.com/php/php_functions.asp) |
| Opdracht\* | Leg uit welke twee soorten functies er bestaan en wat het verschil is?  Leg uit hoe de standaard syntax van een function in PHP eruit ziet?  Leg uit wat een argument is?  Leg uit hoe je een argument meegeeft aan een function?  Leg uit hoeveel argumenten je kan meegeven aan een functie?  Leg uit hoe je meerdere argumenten meegeeft aan een function?  Leg uit hoe je default argumenten meegeeft aan een function?  Leg uit hoe je een waarde returned?  Leg uit wat het nut is van een return?  Voer [Functions Opdrachten 1 t/m 4 →](https://www.w3schools.com/php/exercise.asp?filename=exercise_functions1) uit. |

**Opdracht: 1.13 - Arrays**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over [Arrays in PHP →](https://www.w3schools.com/php/php_arrays.asp) |
| Opdracht\* | Leg uit wat het nut is van een Array?  Leg uit welke soorten Arrays er bestaan en wat de verschillen zijn?  Leg uit op welke twee manieren je Arrays kan definiëren? Tip: Shortened Syntax.  Leg uit wat je allemaal kan opslaan in een Array?  Voer [Arrays Opdrachten 1 t/m 4 →](https://www.w3schools.com/php/exercise.asp?filename=exercise_arrays1) uit. |

### 

## **Blok 2 - Introduction to Object Oriented Programming in PHP**

### Bijlage 3 - Persoonlijk Werkdocument

|  |  |
| --- | --- |
| Naam Student |  |
| Opleidingsgroep |  |

#### **Opdracht: 1.1 - WHAT IS OOP?**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools de [PHP OOP Intro →](https://www.w3schools.com/php/php_oop_what_is.asp) |
| Opdracht\* | Leg in je eigen woorden uit wat het verschil is tussen flat PHP en Object Oriented Programming? |
| Opdracht\* | Leg in je eigen woorden uit welke **vier voordelen** Object Oriented Programming heeft ten opzicht van flat PHP. |
| Opdracht\* | Leg in je eigen woorden uit wat de twee hoofdonderdelen zijn van Object Oriented Programming? |
| Opdracht\* | Leg in je eigen woorden uit wat het verschil is tussen een Class en een Object?. |
| Opdracht\* | Leg in je eigen woorden uit wat er overgeërfd wordt van een Class als een object aangemaakt wordt? |
| Opdracht\* | Bedenk zelf een voorbeeld van een Class en bijbehorende Objecten anders dan afgebeeld op W3SCHOOLS. |

#### **Opdracht: 1.2 - Classes and Objects**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over het [Classes en Objects in PHP OOP →](https://www.w3schools.com/php/php_oop_classes_objects.asp) |
| Opdracht\* | Objecten erven dingen van elkaar over.  Leg uit in je eigen woorden uit wat het verschil is tussen objecten? |
| Opdracht\* | Wat is de meest basic Syntax definitie van een Class?  Uit welke **drie** onderdelen bestaat deze Syntax? |
| Opdracht\* | Hoe worden variables en functions in een Class genoemd? |
| Opdracht\* | Uit welke twee verschillende soorten fundamentele onderdelen kan de body van een Class bestaan? |
| Opdracht\* | Hoeveel objects kunnen er van een Class gemaakt worden? |
| Opdracht\* | Welke keyword gebruik je om je object mee aan te maken? |
| Opdracht\* | Waar wordt het “$this” keyword voor gebruikt? |
| Opdracht\* | Hoe kan je een property van binnen in de Class veranderen versus buiten de klasse? |
| Opdracht\* | Waar kun je de keyword **instanceOf** voor gebruiken? |

#### **Feedbackmoment met de tech coach**

|  |
| --- |
| Als je deze opdrachten hebt uitgevoerd, stuur dan een bericht naar jouw tech coach om mee te kijken met de uitgevoerde opdrachten. |

#### **Opdracht: 1.3 - Constructor**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over [Constructor in PHP OOP →](https://www.w3schools.com/php/php_oop_constructor.asp) |
| Opdracht\* | Leg uit waar een Constructor voor gebruikt wordt en wat het voordeel is? |
| Opdracht\* | Hoe ziet de basic syntax van de Constructor eruit? |

#### **Opdracht: 1.4 - Access Modifiers**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over [Access Modifiers in PHP OOP →](https://www.w3schools.com/php/php_oop_access_modifiers.asp) |
| Opdracht\* | Welke drie Access Modifier bestaan er? |
| Opdracht\* | Wat zijn hun verschillende rollen? |
| Opdracht\* | Op welke twee onderdelen van een Class kan je Access Modifiers gebruiken? |
| Opdracht\* | Wat denk je dat het nut is van Access Modifiers in een “echte” applicatie? |

#### **Opdracht: 1.5 - Inheritance**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over [Inheritance in PHP OOP →](https://www.w3schools.com/php/php_oop_inheritance.asp) |
| Opdracht\* | Wat is Inheritance? |
| Opdracht\* | Wat is het verschil tussen een Child Class en een Parent Class? |
| Opdracht\* | Welke keyword wordt gebruikt om aan te geven dat er Inheritence plaatsvindt? Wat is de basis Syntax van Inhertience in zijn geheel? |
| Opdracht\* | Hoe kan je Inherited Methods Overridden? Wat is het nu hiervan denk je? |
| Opdracht\* | Waar wordt de Final keyword voor gebruikt? |

#### **Opdracht: 1.6 - Class Constants**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools [Class Constants in PHP OOP →](https://www.w3schools.com/php/php_oop_constants.asp) |
| Opdracht\* | Wat is het nut van een Class Constant? |
| Opdracht\* | Hoe declareer je een Class Constant? |
| Opdracht\* | Hoe verschilt een Class Constant van een Class Property? |
| Opdracht\* | Hoe access je een Class Constant van buiten de Class? |
| Opdracht\* | Hoe access je een Class Constant van binnen de Class? |

#### **Feedbackmoment met de tech coach**

|  |
| --- |
| Als je deze opdrachten hebt uitgevoerd, stuur dan een bericht naar jouw tech coach om mee te kijken met de uitgevoerde opdrachten. |

#### **Opdracht: 1.7 - Static Methods**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over [Static Methods in PHP OOP →](https://www.w3schools.com/php/php_oop_static_methods.asp) |
| Opdracht\* | Wat is het verschil tussen een Static Method en een reguliere Class Method? |
| Opdracht\* | Hoe ziet de Syntax van een Static Method eruit? |
| Opdracht\* | Hoe gebruik je de Static Method? |

#### **Opdracht: 1.8 - Static Properties**

|  |  |
| --- | --- |
| Informatie | Lees op W3Schools over [Static Properties in PHP OOP →](https://www.w3schools.com/php/php_oop_static_properties.asp) |
| Opdracht\* | Wat is het verschil tussen een Static Property en een reguliere Property? |
| Opdracht\* | Hoe declareer je een Static Property? |
| Opdracht\* | Hoe gebruik je een Static Property **binnen** een Class? |
| Opdracht\* | Hoe gebruik je een Static Property **buiten** een Class? |

#### **Feedbackmoment met de tech coach**

|  |
| --- |
| Als je deze opdrachten hebt uitgevoerd, stuur dan een bericht naar jouw tech coach om mee te kijken met de uitgevoerde opdrachten. Je gaat nu met je tech coach bespreken wat je verder nog wilt leren over de PHP fundamentals, samen maak je hiervoor een plan. |

### Bijlage 4 - Exercises

Hier een aantal oefeningen zodat je in de praktijk een aantal praktische classes kan schrijven.

#### **Basic Class Exercises**

De onderstaande opdrachten zijn bedoeld om kennis te maken met het schrijven van OOP code.

#### **Opdracht 1**

Write a simple PHP class which **automatically** displays the following string when **initialized**.

**Output**: 'MyClass class has been initialized!'

#### **Opdracht 2**

Write a simple PHP class which displays an introductory message like "Hello All, I am Scott", where "Scott" is an argument value of the method within the class.

#### **Opdracht 3**

Write a PHP calculator class which will accept two values as arguments, then add them, subtract them, multiply them together, or divide them on request.

For example :

$mycalc = new MyCalculator( 12, 6);

echo $mycalc- > add(); // Displays 18

echo $mycalc- > multiply(); // Displays 72

### Bijlage 5 - Eindopdracht

Overleg met je tech coach wat op dit moment voor jouw situatie handig is om te doen.

Dit kan zijn het maken van een van de complexere eindopdrachten die [hieronder](#_evhv1mjdf1h8) beschreven staan. Afhangend van de tijd kan er ook besloten worden om eerst door te gaan met het volgende blok.

**BELANGRIJK!**

**Via de volgende link is een template app voor je opgesteld.** [**Download**](https://drive.google.com/drive/folders/1GFja-8OEiqzVr-111_6iLoLfaXODc7YG?usp=sharing) **deze naar je eigen omgeving en maak hierin de gekozen eindopdracht.**

**Als je in je applicatie gegevens moet invoeren mag je dit hardcoded doen. Dit hoeft dus NIET via een formulier te submitten.**

**We zijn vooral aan het oefenen met het aan elkaar knopen en gebruiken van OOP.**

### 

#### **Opdracht 1 - Taxikosten**

Een taxibedrijf heeft een aantal taxi‟s. De volgende tarieven worden gehanteerd:

* Per gereden km € 0,50.
* Daarboven per gereden minuut € 0,17.
* Van vrijdagavond 22.00 uur tot maandagochtend 7.00 uur geldt een toeslag van 15% (het begintijdstip is hiervoor bepalend).

Er dient een programma te worden ontwikkeld waarmee voor iedere taxirit de begin-en eindtijd (formaat hhmm), evenals de weekdag en de afstand kunnen worden ingevoerd. Voor elke rit moet het verschuldigde bedrag worden berekenden getoond.

Verder moeten getoond kunnen worden:

* cumulatief de totale inkomsten van het taxibedrijf.
* De gemiddelde afstanden over alle gereden ritten.
* De langste ritafstand.

#### **Opdracht 2 - Ouderbijdrage**

Een basisschool berekent de ouderbijdrage als volgt:

* Voor een kind jonger dan 6 wordt€ 38,= gerekend
* voor een kind jonger dan 10 (en niet jonger dan 6) € 50,=
* voor een kind vanaf 10 jaar € 65,=.

Er dient een programma te worden ontwikkeld waarmee voor ieder kind de geboortedatum (formaat ddmmjjjj) kan worden ingevoerd. Voor elk kind moet de verschuldigde ouderbijdrage worden berekend en getoond. Verder moet getoond worden:

* cumulatief de totale bijdrage
* de leeftijd van de jongste leerling
* het aantal kinderen in iedere categorie

#### **Opdracht 3 - Autoverhuur**

Een autoverhuurbedrijf bezit een aantal auto‟s. Deze auto‟s kunnen worden verhuurd. De volgende tarieven worden daarbij gehanteerd: Per gereden km € 0,18 (voor elke dag dat de auto verhuurd is zijn de eerste 100 km vrij). Daarnaast een bedrag van € 50,=per dag.

Er dient een programma te worden ontwikkeld waarmee voor iedere auto de begin-en einddatum (formaat ddmmjjjj) evenals de gereden afstand kunnen worden ingevoerd.

Voor elke verhuurperiode moet cumulatief de verschuldigde kosten worden berekend en getoond. Verder moeten de totale inkomsten, de gemiddelde afstand en de langste huurperiode worden getoond.

#### **Opdracht 4 - Camping**

Bij een camping komen reserveringen binnen. De volgende tarieven worden door de camping gehanteerd: Per plaats € 15,=. Daarbij per persoon € 2,50 en voor de auto € 3,=. Van 11 juli t/m 15 augustus geldt een toeslag van 25%.

Er dient een programma te worden ontwikkeld waarmee voor iedere reservering de begin-en einddatum (formaat ddmmjjjj), evenals het aantal personen en wel of geen auto kunnen worden ingevoerd. Voor elke reservering moeten de verschuldigde kosten worden berekend getoond. Verder moeten cumulatief de totale inkomsten, het gemiddeld aantal dagen waarvoor gereserveerd wordt en het totaal aantal personen worden getoond.

#### **Opdracht 5 - Schaatskampioenschap**

Aan een schaatskampioenschap doet een aantal schaatsers mee. Achtereenvolgens worden de volgende afstanden verreden: 500 meter, 5000 meter, 1500 meter en 10000 meter. Tijden worden geregistreerd tot in honderdsten van seconden nauwkeurig. De tijden behaald op de verschillende afstanden worden omgerekend naar punten door elke tijd te herleiden tot een 500 meter tijd. De rijder met het laagste puntentotaal mag zich kampioen noemen.

Er dient een programma te worden ontwikkeld waarmee voor elke deelnemer achtereenvolgens de naam en de tijden (formaat mmsshh) kunnen worden ingevoerd. Van iedere rijder moet vervolgens het puntentotaal worden getoond en de tevens wie de winnaar is.

#### **Opdracht 6 - Containerverhuur**

Bij een containerverhuurbedrijf worden containers verhuurd. Daarbij worden de volgende tarieven gehanteerd: de huur bedraagt € 40,= per m3 per dag. Het afvoeren van een bak kost € 60,= als het volume van de bak 2 m3 of minder bedraagt, anders € 125,=.

Er dient een programma te worden ontwikkeld waarmee voor iedere container, het volume van de bak, evenals de begin-en einddatum (formaat ddmmjjjj) van de huurperiode kunnen worden ingevoerd. Voor iedere verhuurde container moeten de verschuldigde kosten worden berekend en getoond. Daarnaast moeten cumulatief de totale inkomsten, het gemiddelde volume en de langste huurperiode worden getoond.

### 

#### **Opdracht 7 - Vakantiedagen**

Het aantal vakantiedagen van werknemers van een bedrijf wordt als volgt berekend: werknemers van afdeling 1 hebben 24 basis vakantiedagen, werknemers van andere afdelingen hebben 20 basis vakantiedagen. Daarbij krijgen werknemers ouder dan 55 jaar 5 extra vakantiedagen en werknemers die meer dan 10 jaar in dienst zijn 3 extra vakantiedagen.

Er dient een programma te worden ontwikkeld waarmee voor iedere werknemer de geboortedatum en datum van indiensttreding (formaat ddmmjjjj) en een viercijferig werknemersnummer kunnen worden ingevoerd. Het eerste cijfer van het werknemersnummer komt overeen met de afdeling. Voor elke werknemer moet het aantal vakantiedagen worden bepaald en getoond. Verder moeten cumulatief het totale aantal vakantiedagen, het gemiddeld aantal dienstjaren en de oudste werknemer worden getoond.

#### **Opdracht 8 - Contributie**

Bij het bepalen van de jaarlijkse contributie voor een sportvereniging gelden de volgende regels: Seniorleden betalen € 150,=, juniorleden € 75,=. Men is senior lid als men 18 jaar of ouder is. Spelende leden betalen bovendien nog € 45,= bondscontributie.Voor leden die langer dan 7 jaar lid zijn, wordt op de clubcontributie 5% korting in rekening gebracht.

Er dient een programma te worden ontwikkeld waarmee voor elk lid de naam,geboortedatum en datum van ingang lidmaatschap (formaat ddmmjjjj) kunnen worden ingevoerd, evenals of het lid wel of niet spelend lid is. Voor elk lid moet de contributie berekend en getoond worden. Verder moet cumulatief de totale contributie, het gemiddeld aantal lidmaatschap jaren en het jongste lid worden getoond.

#### **Opdracht 9 - Catering**

Een bedrijf verzorgt de catering van personen. Voor de periode aangegeven met een begindatum en een einddatum wordt de administratie voor een aantal klanten verwerkt. Een klant kan op een dag een voorgerecht (3 euro) en of een hoofdgerecht (5 euro)en of een nagerecht (2 euro) gebruiken.Voor een klant wordt opgeslagen de naam, aantal voorgerechten, aantal hoofdgerechten en aantal gerechten die de klant heeft gehad.

Uiteindelijk wil het bedrijf de volgende informatie gepresenteerd hebben:

Het totaal omgezette bedrag; De gemiddelde omzet per dag; De naam van de klant die voor het hoogste bedrag heeft geconsumeerd; De naam van de klanten die wel voorgerechten maar nooit een nagerecht hebben gehad; De namen van de klanten die voor een hoger bedrag dan het gemiddelde per klant hebben geconsumeerd.

### 

#### **Opdracht 10 - Bootverhuur**

Een bedrijf verhuurt speedboten. Voor Één bepaalde dag wordt elk van de tien aanwezige speedboten hoogstens 1 keer verhuurd. Gevraagd wordt een programma om de administratie van één dag bij te houden. Per boot worden de volgende gegevens vastgelegd: gewicht, aantal pk motorvermogen, lengte en bootnummer en de huurprijs per uur. Verder wordt afhankelijk van de verbruikte hoeveelheid brandstof een verhuur toeslag van 5 euro per liter berekend. Voor verhuurde boten wordt vastgelegd aanvangstijd verhuur, eindtijd verhuur,aantal liters verbruikte brandstof en opgelopen schade. Aan het einde van de dag wil de eigenaar de volgende gegevens gepresenteerd hebben: De totale omzet over alle boten; De totale verhuur tijd over alle boten; Het nummer van de boot met het hoogste brandstofverbruik per minuut tezamen met dit verbruik; Het percentage van de verhuurde boten dat schade heeft opgelopen; De boot die het kortst is verhuurd.

## **Blok 3 - DATA INTERACTION WITH SQL AND DATABASE MANAGEMENT IN MYSQL.**

### Bijlage 6 - Het Project: CV Applicatie

#### **Projectbeschrijving**

Het doel van deze opdracht is het ontwikkelen van een applicatie waarmee CV’s getoond kunnen worden. Het is de bedoeling dat voor meerdere personen een CV kan worden getoond op het scherm. Een gebruiker moet geselecteerd kunnen worden en vervolgens moet zijn of haar CV getoond worden op het scherm.

#### **Databeschrijving**

Een CV per persoon wordt opgebouwd uit de volgende data:

|  |  |
| --- | --- |
| **Persoonsgegevens** | * Voornaam * Achternaam * Woonadres * Telefoonnummer * Email * Geboortedatum * Nationaliteit * Linkedin Profiel |
| Welke **opleidingen** die persoon gevolgd heeft. | * Schoolnaam * Opleidingsnaam * Beschrijving van de opleiding * Startdatum * Einddatum |
| Welke **talen** de persoon beheerst. | * de taal * Niveau van taal aangegeven tussen 1 en 5. |
| Welke **werkervaring** die persoon heeft gehad. | * functie * bedrijf * startdatum * einddatum * functiebeschrijving |
| Welke **vaardigheden** die persoon bezit. | * de vaardigheid * niveau van vaardigheid aangegeven tussen 1 en 5. |
| Welke **hobby's** die persoon heeft. | * Naam van hobby. * Beschrijving van de hobby. |
| **Optioneel: Wat jij zelf nog wil toevoegen.** | |

#### **Ontwikkelfases voor het Project**

We zouden als klant graag de volgende fasen zien in het project:

* We zouden graag een CV op het scherm willen zien.
* Ten tweede zou het top zijn om net overzicht te hebben van alle personen die klant zijn bij ons.
* We zouden ten slotte ook graag de mogelijkheid hebben om een CV van een persoon in te zien via dit overzicht.
* Daarnaast zouden we graag nieuwe personen en cv’s willen kunnen toevoegen en editten.
* Optioneel: We zouden graag flexibel meerdere layouts willen hebben voor onze CV’s.

### 

#### **Voorbeeld van een CV**

Onderstaand hebben we een mooi voorbeeld gevonden van hoe een CV eruit zou moeten zien. Hij is niet helemaal volledig op de afbeelding maar het geeft een goede indicatie van de netheid.



**Einde Projectbeschrijving**

#### **Relevante Informatie**

Hieronder nog relevante informatie voor je als student.

#### **Inleidend**

Hier even kort wat je moet weten voordat je begint met het daadwerkelijk ontwikkelen.

Alle projecten beginnen met een **projectbeschrijving.** Hierin wordt duidelijk ingekaderd wat de applicatie die je gaat bouwen moet gaan doen. Zo weet je precies wat je gaat bouwen voordat je begint en kom je niet in de knoei in een later stadium.

Naast een projectbeschrijving is het ook belangrijk om te weten HOE de applicatie eruit komt. Door een simpel **wireframe** te maken maak je inzichtelijk wat er wordt weergegeven en op welke plek dit moet komen te staan.

Uit de wireframe kan je afleiden welke data je uiteindelijk moet gaan opslaan. Je kijkt naar wat er wordt weergegeven en maakt op basis hiervan **een datamodel** voor je applicatie. Dit geeft je inzicht in hoe je data aan elkaar relateert en geeft je de mogelijkheid om na te denken over hoe je jouw applicatie gaat structureren voordat je ook maar een regel code geschreven hebt.

#### **Datamodel Uitwerken**

Begin met het maken van een **lijst van entiteiten (tabellen)** die je haalt uit de projectbeschrijving. Je kunt ze spotten in de projectbeschrijving door de uitgewerkte databeschrijving te bestuderen.

Als deze er niet is moet je dit doen door je af te vragen welke entiteiten (Dit worden je tabellen e.g Cars, Orders, Users, etc) kan ik spotten waarvan ik een of meerdere eigenschappen (kolommen) moet opslaan.

Je kunt een begin maken met het tekenen van je datamodel door de relaties tussen de users te verbinden met alle andere tabellen.

Tip! Elke tabel, behalve de users table, heeft een user\_id veld, dit wordt een 1 op (n) relatie genoemd:

* 1 user heeft meerdere educations
* 1 user heeft meerdere jobs
* 1 user heeft meerder
* e … etc.

### Relevante Bronnen

<https://laracasts.com/series/mysql-database-design>

<https://www.w3schools.com/php/php_mysql_intro.asp>

[https://www.mysqltutorial.org/mysql-data-types.aspx](https://www.w3schools.com/php/php_mysql_intro.asp)

## **Blok 4 - Fundamentals in Laravel**

### Bijlage 7 - Composer Installation Instructions

#### **Step 1 - install PHP (Currently v8.1.6)**

* go to <https://windows.php.net/download/>
* download the x64 or x86 non thread safe php maps
* extract the contents of the map in a folder called code/php/php-8.1.6

#### **Step 2 - Install composer**

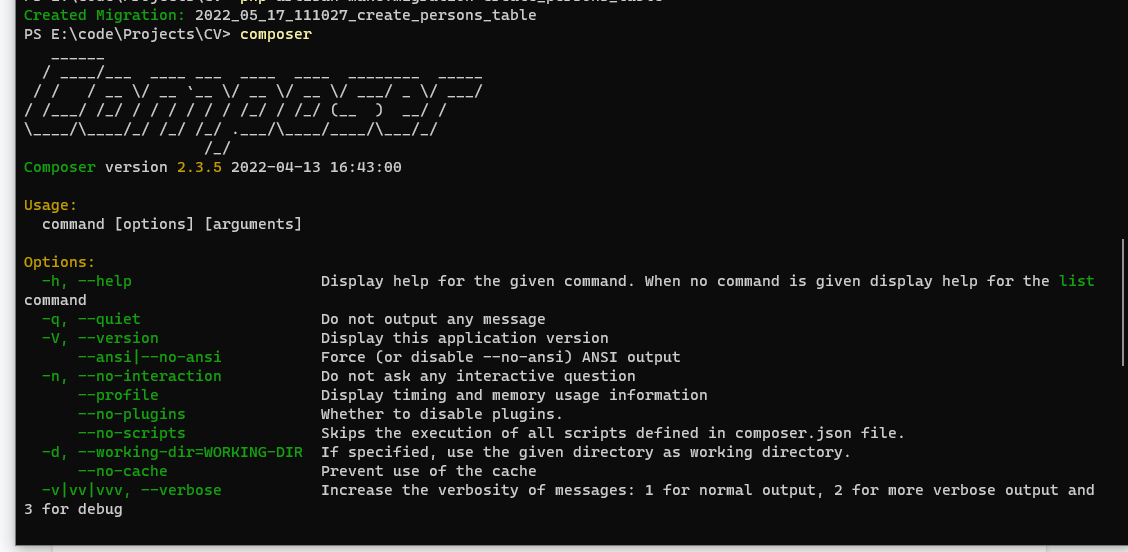
* go [here →](https://letmegooglethat.com/?q=composer) and click on the download button.
* Download the installer for your machine and run it.
* Click next until you are prompted to reference your php installation.
* Navigate to the folder created in step 1 and select the php.exe
* Once installation is complete restart restart terminal.
* Check if composer and php are correctly installed by executing the following commands: composer -v and php -v. You should see the following [expected output →](#_qcjb23vdd356) Otherwise contact your tech coach.

#### **Step 3 - Installing Laravel**

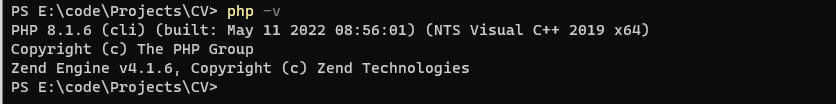
* install the global Laravel installer by running the following command:
  + composer global require laravel/installer
* navigate to the folder where you want to install your new laravel project with the terminal.
  + Hint: I use a code/projects folder to store all project files.
* create a new app with the command:
  + laravel new app
  + If an error occurred contact your techcoach.
* Possible Errors:
  + Enable fileinfo extension inside your php.ini. You can find the php.ini file inside the folder created in step 1. Locatie the php.ini file inside this folder. Open it and search for ext\_fileinfo and uncomment it. Save and try again.
* Navigate using the terminal into the just created project folder.
* Run php artisan serve to host your project.
* You can now access your project by typing localhost:8000 inside your browser. If you see the [laravel start screen as output →](https://docs.google.com/document/d/1MWMENnFc_EVe7DQEa2qVO1k_6_8J7CZPugOpNVBNtlE/edit#heading=h.32mf1t7nq6bz) when visiting the url you are set up correctly!

#### **Expected Output**

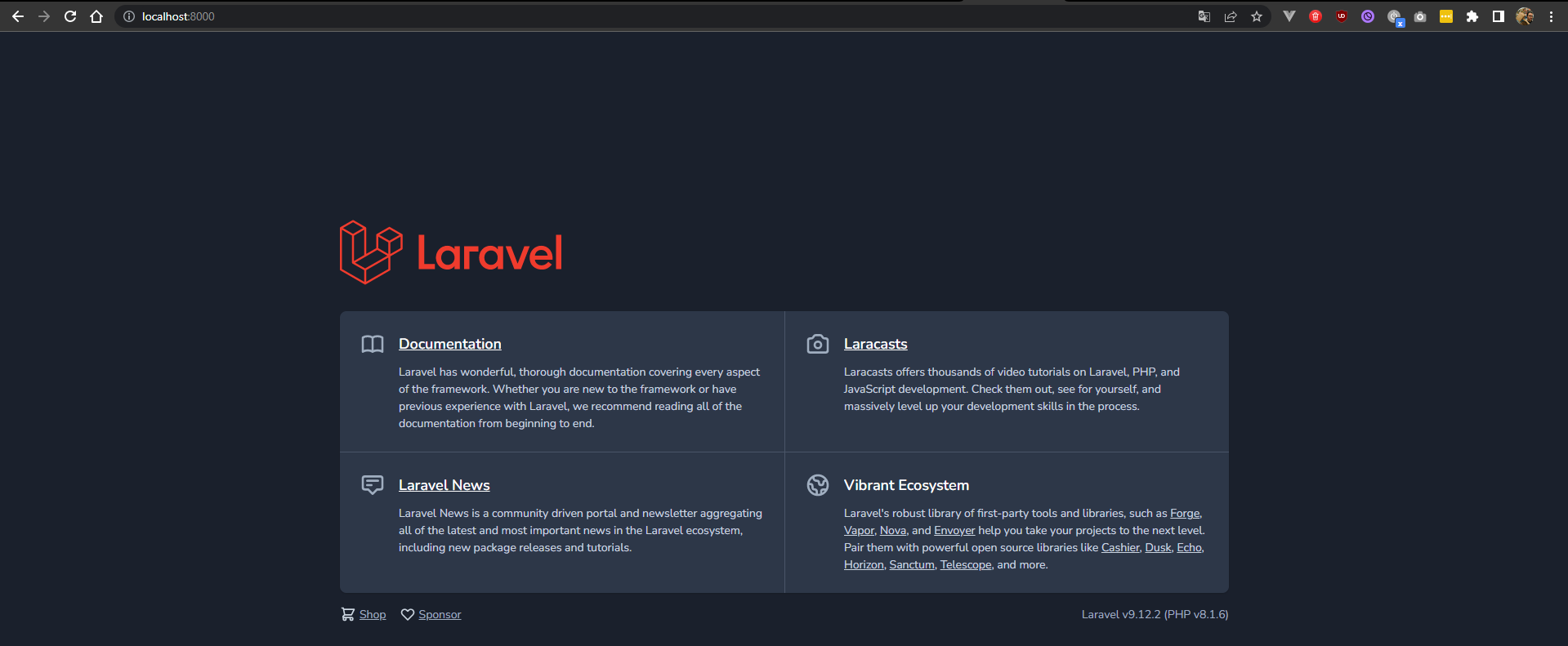
**Composer Installation**



PHP installation



Laravel Start Screen



## **Blok 5 - Laravel als Backend API**

### Bijlage 8 - Projectbeschrijving: Webshop in Laravel

#### **Inleiding**

We gaan alle kennis die we de afgelopen maanden hebben opgedaan combineren om een webshop te ontwikkelen in Laravel en Angular.

Het doel van de webshop is om een aantrekkelijk overzicht van producten met meerdere eigenschappen te kunnen tonen aan een bezoeker die ze vervolgens kan bestellen.

#### **Soort Webshop**

Je bent vrij in het kiezen van wat voor soort webshop je wilt gaan ontwikkelen. Kies als product iets waar je interesse in hebt, bijvoorbeeld:

* een gameshop
* een webshop waar je boeken verkoopt of LP’s
* of misschien een food order shop
* etc

Uit eigen ervaring weet je hoe een webshop er globaal uitziet; kijk daarom eens goed naar je favoriete webshop - misschien Coolblue? - en kijk naar de functionaliteiten van deze webshop. Kijk met name eens goed naar de filters die een webshop gebruikt waardoor je een beeld krijgt hoe de database van een webshop in grote lijnen is opgebouwd.

#### **Samenwerken**

De opdracht wordt uitgevoerd worden in tweetallen. Zorg ervoor dat je elkaar aanvult. Dit is je kans om je te specialiseren in wat je interessant vindt. Wil je meer leren over backend programmeren ga dan een samenwerking aan met een front-ender en vice versa. Maakt het je niet uit verdeel dan de taken eerlijk tussen jullie beide.

#### **Techniek**

Uiteindelijk is het de bedoeling om een webshop te ontwikkelen in **Laravel & Angular**. We beginnen onze ontwikkeling in het **Laravel** framework. Bespreek met je tech coach wat voor jouw groep een handig moment is om **Angular** in te gaan zetten.

## 

#### **Doel**

Het doel van deze opdracht is om aan het einde van de bootcamp je webshop te presenteren aan je medestudenten en het Codegorilla opleidingsteam. De webshop zal gepresenteerd worden op een scherm. Bereid dit goed voor.

#### **Functionaliteiten (Vereisten)**

Je webshop heeft de volgende vereisten opgedeeld in voor- en achterkant, must haves en- nice to haves.

#### **Voorkant Webshop (Voor de bezoeker / klant)**

|  |  |
| --- | --- |
| **Must Have** | Je moet een net overzicht van producten kunnen tonen aan de bezoeker van de webshop. Je moet minimaal 1 (soort) product hebben met meerdere eigenschappen (properties). Denk aan aan kleur, gewicht, afmetingen etc. etc. Deze properties sla je op in een 1 op n relatie. |
| **Must Have** | Je webshop moet beschikken over filter functionaliteit zodat de klant eenvoudiger zijn keuze kan filteren. Denk aan tags, een search functionaliteit of verdeling in categorieën. |
| **Must Have** | Een bezoeker moet een of meerdere producten tegelijkertijd kunnen bestellen. |
| **Must Have** | De te bestellen producten moeten in een winkelwagentje verzameld worden. Deze moet op elke pagina van je website beschikbaar en zichtbaar zijn voor de klant. De uitdaging ligt hier in de functionaliteit om het winkelwagentje steeds up to date te houden. |
| **Must Have** | De bezoeker moet vanuit het winkelwagentje zijn of haar order af kunnen ronden. |
| Nice to Have | Je bezoeker moet een kortingscode kunnen toevoegen bij het bestellen. |
| Nice to Have | Producten moeten met korting voor bepaalde tijd getoond kunnen worden. |
| Nice to Have | Je bezoeker moet een account aan kunnen maken zodat makkelijk opnieuw besteld kan worden. |
| Nice to Have | \*Verzin zelf een aantal toffe features\* |

#### **Achterkant Webshop (Beheerders Dashboard)**

|  |  |
| --- | --- |
| **Must Have** | Bouw CRUD functionaliteiten zodat de webshop dynamisch beheerd kan worden. Denk aan het toevoegen van producten, categorieën etc. |
| **Must Have** | Ontwikkel authenticatie functionaliteit. CRUD functionaliteiten moeten alleen uitgevoerd kunnen worden door ingelogde gebruikers. |
| Nice to Have | Toon relevante statistieken zoals totaal aantal bestelde producten, totale omzet, omzet per klant, nieuwe bestellingen etc. |
| Nice to Have | Toon een overzicht van orders en/of facturen. |
| Nice to Have | Toon een overzicht van alle klanten. |
| Nice to Have | \*Verzin zelf een aantal toffe features\* |

## 

#### **Tips**

Concentreer je eerst op de front-end; de voorkant die de klant ziet en waar alle

functionaliteit in zit waardoor een klant de mogelijkheid heeft om een bestelling te

plaatsen.

Zoek op internet op ‘data model webshop’ naar afbeeldingen/voorbeelden om een

indicatie te krijgen hoe een gemiddeld data model van een webshop eruit ziet.

Houdt het simpel want een maand werken aan een opdracht lijkt heel lang, maar je

zult merken dat je deze maand echt nodig hebt.

Wanneer je het data model hebt uitgeschreven, begin dat met het aanmaken van alle

migrations en seeders. Je hebt data nodig om aan de front-end van de webshop te

werken, maar ook voor de back-end om CRUD actions toe te passen.

Neem hier uitgebreid de tijd voor want als je de database, models en controllers

eenmaal hebt staan, dan heb je daar gedurende het bouwen aan je project profijt

van. Voor vrijwel elke tabel heb je een model en een controller nodig; maak deze meteen

aan nadat je een migration hebt geschreven.

Prijzen en BTW-tarieven van producten kunnen in de toekomst wijzigen, houd hier

rekening mee wanneer je data opslaat in b.v. de order\_products table (de producten

die de klant heeft besteld). Tenslotte moet b.v. na een jaar de factuur (invoices) nog

steeds de prijzen en BTW-tarieven uit het verleden tonen.

#### **Documentatie (Extra)**

De beste documentatie vind je op <https://laravel.com/docs> en https://stackoverflow.com.

Er zijn heel veel handige en nuttige packages die je kan gebruiken, bijvoorbeeld:

* spatie/laravel-permission (composer)
* uxweb/sweet-alert (composer)
* @fortawesome/fontawesome-free (NPM) sweetalert2

#### **Aandachtspunten**

* Verdeel je code netjes in mappen en in kleine, leesbare files
* Maak gebruik van meerdere CSS files i.p.v. een 1 groot CSS bestand, tenzij je maar weinig CSS extra gaat gebruiken
* In PHP gebruik je zoveel mogelijk camelCase namen voor variabelen, maar in Laravel zie je vooral files die met een hoofdletter beginnen, bijvoorbeeld UserController.php. Houd je aan de standaard van Laravel.
* Schrijf nette en consistente code. Denk hierbij aan indents, regels tussen codeblokken, comments wanneer nodig, ruim code die commented is op wanneer je deze code niet meer gebruikt etc. etc.
* Foutmelding(en) in de console van de browser
* De opzet van je database en het gebruik van datatypen
* Het gebruik van Models en Controllers
* Het gebruik van blades en hoe je data naar een blade stuurt
* We kijken met name ook naar de packages die je hebt geïnstalleerd en dat je deze met Composer en NPM gebruikt (Laravel mix)
* Welke standaard functionaliteit van Laravel heb je gebruikt
* CRUD actions
* Technieken
  + PHP/Laravel MVC framework
  + MySQL
  + HTML, CSS, eventueel Javascript
  + Git
  + Composer
  + NPM
  + Routing (GET, POST, PUT/PATCH en DELETE)

## 

## 