**MBO Rijnland**

Sjabloon Ontwerp

Volgens de ontwikkel concepten UML zijn de hierna volgende producten

uitgewerkt:

* Functioneel ontwerp
* Technisch ontwerp
* Bouw ontwerp

Dit document is opgesteld op basis van de richtlijnen die zijn opgesteld voor het afnemen van MBO examens ontwerp en bouwen.

In dit document is ook opgenomen het itereren conform Agile/Scrum

MBO Rijnland Gouda

Versie : 0.1

Datum : 12 mei 2023

Auteur : Roan van Dam

**Inhoud**

[1 Inleiding 2](#_Toc21590)

[2 Opdrachtomschrijving 3](#_Toc20690)

[2.1.1 Opdrachtomschrijving 3](#_Toc5475)

[2.1.2 Probleemstelling 3](#_Toc31380)

[2.1.3 Oplossing 3](#_Toc20353)

[2.2 Afstemmen met betrokkenen 3](#_Toc11181)

[2.3 Maak een overzicht User Stories en een product Backlog 4](#_Toc3329)

[3 Planning en begroting 5](#_Toc13159)

[4 Functioneel ontwerp 7](#_Toc7489)

[4.1 Uitvoeren iteraties 7](#_Toc21504)

[4.2 Context Klassen diagram (domein analyse) 8](#_Toc28815)

[4.3 Use case diagram met beschrijving (domein analyse) 9](#_Toc16236)

[4.4 Database Ontwerp (ERD) 10](#_Toc28777)

[4.4.1 Conceptueel model (ERD) 11](#_Toc8995)

[4.4.2 Normaliseren 12](#_Toc7824)

[4.4.3 Implementatie model 13](#_Toc871)

[4.4.4 Relationeel Representatie model 13](#_Toc22972)

[5 Technisch ontwerp 15](#_Toc27040)

[5.1 Inleiding 15](#_Toc23803)

[5.2 Wireframes 15](#_Toc18118)

[5.3 Mock-Ups 16](#_Toc30464)

[5.4 Keuze framework 16](#_Toc12785)

[6 Bouw 17](#_Toc8428)

[6.1 Architectuur Client/Server 17](#_Toc11395)

[6.2 Bouw ontwerp modules 17](#_Toc6229)

[6.3 Logische module beschrijving 18](#_Toc8331)

[6.4 Fysieke module beschrijving 18](#_Toc32410)

[7 Versie beheer 18](#_Toc13132)

[8 Testen 18](#_Toc20421)

[9 Opleveren (implementeren) 18](#_Toc5403)

[10 Resultaat 19](#_Toc21897)

[11 Versie beheer 19](#_Toc226)

# Inleiding

Dit document is voor de ALA opdracht van de 4e periode. In dit document staan de volgende punten omschreven:

Beschrijf beknopt en puntgewijs de inhoud van dit rapport als het gaat om de uitwerking van de hierna genoemde punten:

* Opdrachtomschrijving
* Probleemstelling
* Oplossing
* Planning en begroting
* Functioneel ontwerp
* Technisch ontwerp
* Bouw ontwerp
* Testen
* Presenteren

# Opdrachtomschrijving

### Opdrachtomschrijving

De casus -> een casus komt over algemeen tot stand na interview met de opdrachtgever en op basis hiervan wordt een opdrachtomschrijving opgesteld.

### Probleemstelling

Wat zijn de problemen die zich voordoen met betrekking tot de casus

### Oplossing

Bestudeer probleemstelling en werk op basis hiervan oplossingsvoorstellen uit

Te gebruiken technieken en methoden

***Denk aan UML, OOP, ERD, Agile/Scrumm etc***

## Afstemmen met betrokkenen

Pitchen/presenteren

## Maak een overzicht User Stories en een product Backlog

***User story***

User stories passen goed in een Agile/Scrum frame-work.

Een user story is geen functionele beschrijving, maar het maakt duidelijk wat een (eind-) gebruiker wil en waarom.

Bij het opstellen van een user-story moet rekening gehouden worden met de hierna volgende punten:

* Het is een functionele wens van de gebruiker
* Het is in gebruikerstaal geschreven
* Het heeft een toegevoegde waarde voor de gebruiker
* Het is een discussie en geen contract

Een userstory wordt als volgt opgesteld:

Als {wie} wil ik {wat} zodat ik {waarom}

Hieronder wat voorbeelden:

*“Als een manager wil ik een dashboard hebben waarop ik de prestaties van mijn team kan zien, zodat ik de voortgang van projecten kan monitoren en bijsturen indien nodig.*

*“Als een student wil ik feedback ontvangen op mijn huiswerkopdrachten, zodat ik mijn prestaties kan verbeteren en mijn cijfers kan verhogen.”*

*“Als een softwareontwikkelaar wil ik toegang hebben tot de documentatie van de API, zodat ik mijn code kan schrijven op basis van de juiste specificaties.”*

*“Als een marketingmedewerker wil ik rapportages kunnen genereren van de websitebezoekers, zodat ik de effectiviteit van marketingcampagnes kan meten en verbeteren.”*

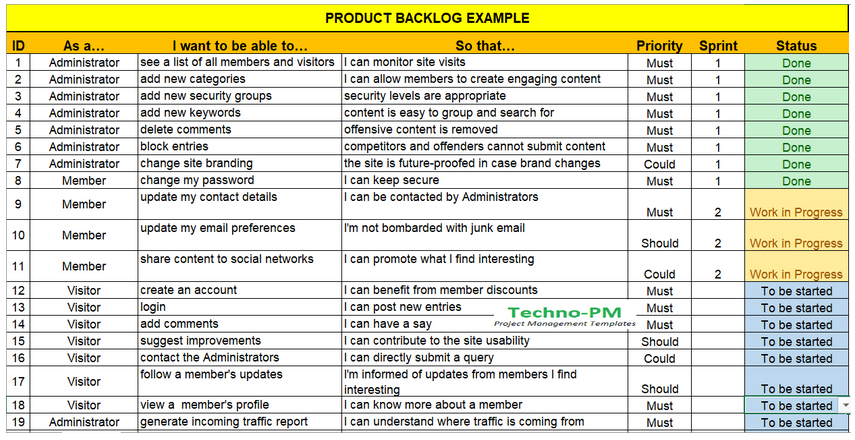
*“Als een forens wil ik real-time updates ontvangen over het openbaar vervoer, zodat ik op tijd op mijn bestemming aankom en mijn reistijd kan optimaliseren.”*

*“Als een online shopper wil ik de mogelijkheid hebben om producten te vergelijken, zodat ik de beste keuze kan maken en geld kan besparen.”*

***Product Backlog:***

Het ordenen van user-stories gebeurt in een zogenaamde “product backlog” en deze voorzien van prioriteiten conform, bijvoorbeeld MoSCoW.

Hieronder een voorbeeld:



# Planning en begroting

Planning en begroting betreffen de hieronder genoemde activiteiten:

**Projectfases:**

Initiatief Analyse (opdrachtsomschrijving, probleemstelling en oplossing

Ontwerp Functioneel- en technisch ontwerp

Realisatie Bouwen en testen

Implementatie implementatie plan (in productie nemen) en acceptatie-testen

Evaluatie Opleveren en resultaten evalueren

“Planning en Begroting” kan gewoon op 1 excel-lijst worden gezet.

**Begroting**

Een begroting is een inschatting. Alle activitenten projectfases op een lijst zetten met inschatting uren en tarieven. Hieronder een voorbeeld:

Afbeelding met tafel

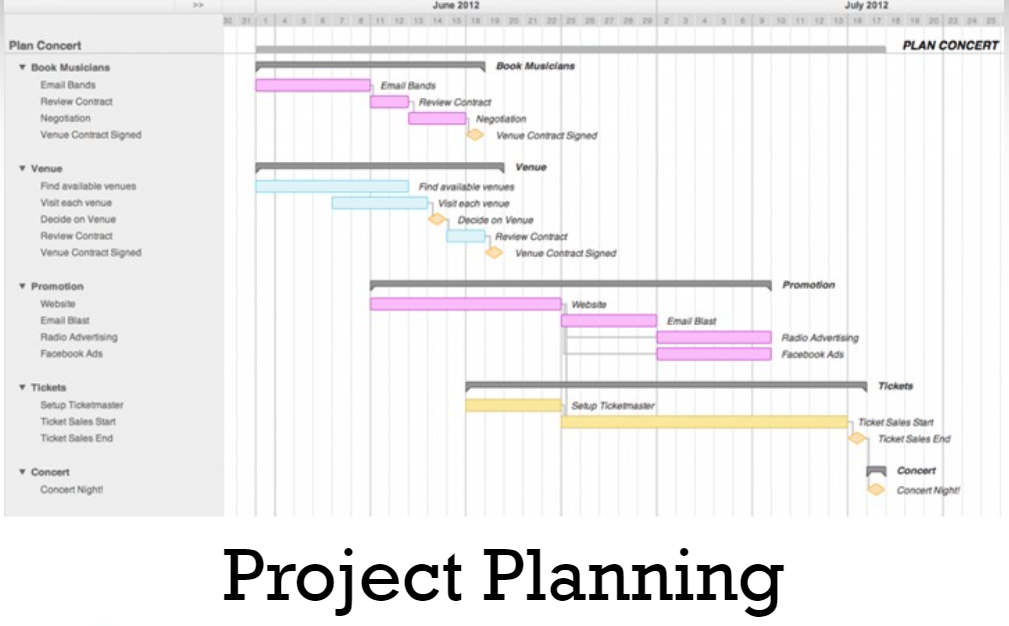
Automatisch gegenereerde beschrijving

**Planning:**

Een planning is een inschatting als het gaat om wanneer te beginnen aan welke taak, en wanneer is het af.

* Neem als basis de overzicht van de activiteiten (zie begroting)
* Bepaal prioriteiten (welke volgorde zal een activiteit worden uitgevoerd)
* Wie doet wat
* Een tijdslijn opstellen wanneer te beginnen en wanneer te eindigen

Hieronder een voorbeeld:



# Functioneel ontwerp

In dit hoofdstuk worden een x-aantal uit te voeren stappen/activiteiten uitgewerkt.

## Uitvoeren iteraties

Plan en plaats de user stories. Schat onder andere de tijd en de prioriteit/volgorde in (bijvoorbeeld met de MoSCoW-methode). Plaats de user stories op de iteratie/sprint backlog.

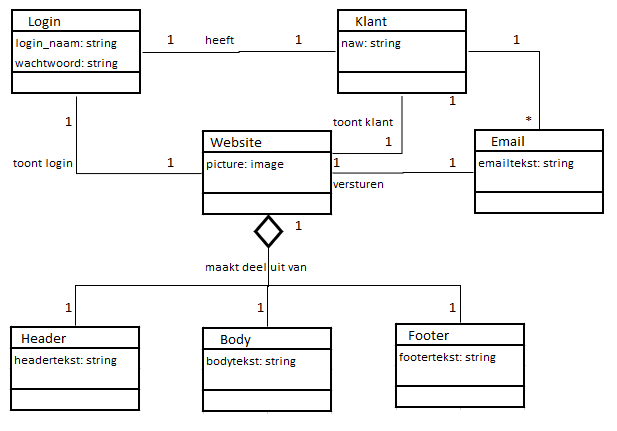
Verdeel binnen het team de eerste op te pakken user stories. Zorg voor een evenwichtige verdeling.

Werk de eigen user stories uit in taken, criteria en een ontwerp. Stel hierbij, passend bij de opdracht, de volgende documenten op of voer de volgende activiteiten uit, zoals:

* Klassendiagram (context)
* Usecase diagram
* ERD (datamodel)
* Normaliseren
* Wireframes
* Mock-ups
* Activiteitendiagram
* Keuze framework

## Context Klassen diagram (domein analyse)

Een Domein analyse is een contextueel klasse diagram als uitgangspunt voor de verder ontwerpen.



Legenda domein diagram:

|  |  |  |
| --- | --- | --- |
| ***Nr*** | ***Klasse*** | ***Omschrijving*** |
| 1 | Login |  |
| 2 | Klant |  |
| 3 | Website |  |
| 4 | Header |  |
| 5 | Body |  |
| 6 | Footer |  |
| 7 | email |  |

[

* Hoe te beginnen:  
  maak een overzicht van alle objecten (vanuit opdrachtsomschrijving, probleemstelling en oplossingen)
* Bepaal relevante en niet relevante
* Vanuit overzicht relevante objecten een context-diagram opstellen

Bepaal de relaties:

* Generalisatie
* Associaties
* Aggregatie
* Compositie

]

## Use case diagram met beschrijving (domein analyse)

Een Use Case diagram is een diagram met als doel een overzicht te geven van het te ontwikkelen systeem en hoe deze zal/kan worden gebruikt.

Dit is de basis voor het beschrijven van de functionaliteiten van het systeem in relatie tot het te ontwerpen systeem.

[

De Use Case is een ovaal met daarin een naam die een bepaalde functionaliteit omschrijft

Use Case wordt met een rechthoek omkaderd (afbakening van het systeem)

De actor is een uitbeelding van een rol of een groep die iets moet kunnen doen op het systeem

De actoren staan altijd buiten het systeem

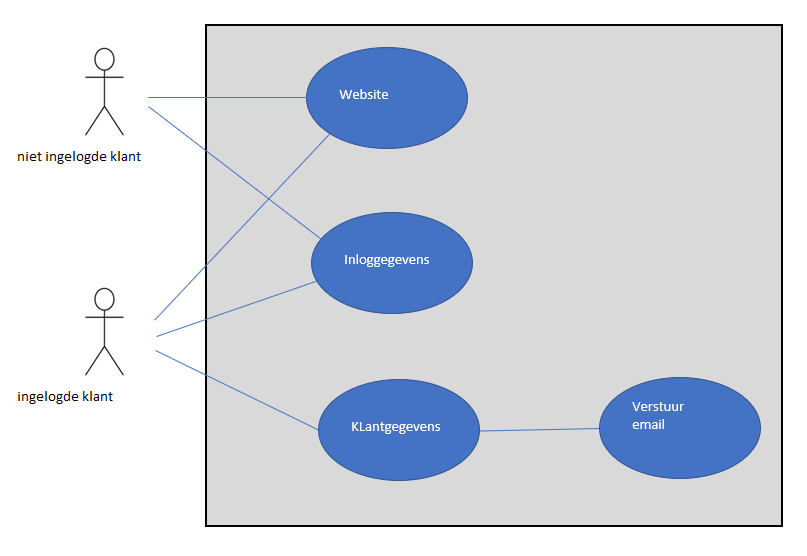
De actoren en use cases worden verbonden met pijlen

Hoe te beginnen:

* Analyseer en maak een overzicht van functionaliteiten (op basis van opdrachts-omschrijving)
* Bepaal relevante en niet relevante functionaliteiten
* Van relevante functionaliteiten een usecase-diagram maken
* Van elke usecase een data dictionary maken

]

***Use Case Diagram***



Data Dictionary Use Case:

|  |  |
| --- | --- |
| UseCase Naam | Website |
| Samenvatting | Boek website {ontwikkel\_systeem} |
| Actoren | Niet ingelogde klanten |
| Aannamen | Met behulp van ip\_adres/url het bezoeken van een website |
| Beschrijving | Klant bezoekt website:  (1)website wordt getoond  (2)met tab inloggen |
| Uitzonderingen | Bij inloggen, indien niet aanwezig melding geven |
| Resultaat | Website wordt getoond met tab login |

|  |  |
| --- | --- |
| UseCase Naam | Inloggegevens |
| Samenvatting |  |
| Actoren |  |
| Aannamen |  |
| Beschrijving |  |
| Uitzonderingen |  |
| Resultaat |  |

|  |  |
| --- | --- |
| UseCase Naam | Klantgegevens |
| Samenvatting |  |
| Actoren |  |
| Aannamen |  |
| Beschrijving |  |
| Uitzonderingen |  |
| Resultaat |  |

|  |  |
| --- | --- |
| UseCase Naam | Email |
| Samenvatting |  |
| Actoren |  |
| Aannamen |  |
| Beschrijving |  |
| Uitzonderingen |  |
| Resultaat |  |

## Database Ontwerp (ERD)

Data modelling is een techniek om te komen tot een goed ontworpen database. Voor het ontwerpen

van een database (het informatie-systeem) zullen de hierna genoemde modellen worden uitgewerkt:

* Een conceptueel model (ERD)
* Normaliseren
* Een implementatie model

### Conceptueel model (ERD)

Een ERD toont het conceptuele ontwerp van een database. Zij laten de verschillende objecten in het informatiesysteem zien, en de bestaande relaties en voorwaarden ertussen.

[

Let wel: er bestaan standaard notatie voor het weergeven van ER-Diagrammen.

Hoe te beginnen:

Bestudeer het klasse-diagram

Maak een kopie van het klasse-diagram en noem het ERD-diagram

Bepaal relevante en niet relevante klassen voor een database (object wel/geen bestand)

De relevante objecten in ERD-diagram behouden en niet relevante objecten verwijderen

Voeg per klasse de nodige attributen, kenmerken en operaties toe

Leg de onderlinge relaties van de objecten (bestanden in wording) vast

Onderlinge relaties kunnen zijn:

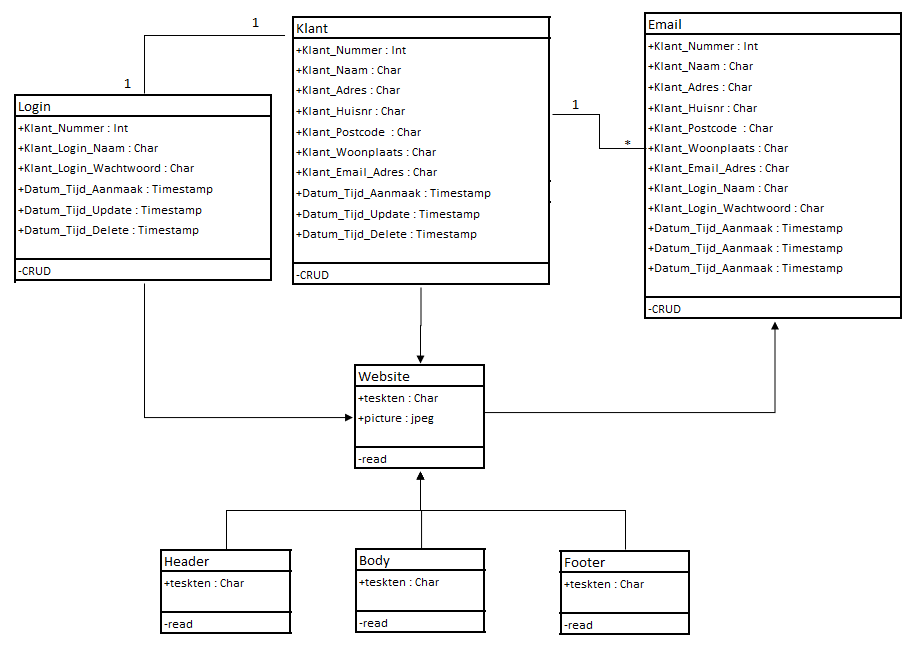
1 op 1 relatie

1 op n relatie

N op n relatie

]

*Uitwerking Entity Relationship Diagram (ERD)*



***Legenda:***

***“+” = public***

***“-“= private***

### Normaliseren

**Het Normalisatie proces:**

Normalisatie Bij het bepalen van de juiste entiteiten en attributen kan normaliseren een belangrijk hulpmiddel zijn. Hiervoor zijn door de Amerikaan Ted Codd regels opgesteld waarvan hier de drie belangrijkste worden toegelicht. In de praktijk worden deze regels niet strikt toegepast omdat ze tot performance verlies kunnen 18 leiden. Stapsgewijs helpen ze echter wel om belangrijke fouten in het ontwerp te ontdekken.

**In totaal zijn er 6 normaalvormen:**

* Nulde normaalvorm (0NV)
* Eerste normaalvorm (1NV)
* Tweede normaalvorm (2NV)
* Derde normaalvorm (3NV)
* Vierde normaalvorm (4NV)
* Vijfde normaalvorm (5NV)

Voor het ontwerpen van een gestructureerde database is het uitwerken van 0 tm 3 voldoende.

(zie ook boek “Backend Programmeren blz 131 tm 143)

**Hoe met behulp van het normalisatie-proces te komen tot een gestructureerde database:**

Stappen

Uitgangspunt van het normaliseren is steeds de informatiebehoefte van de toekomstige

gebruiker van de database. De indeling en inhoud van de tabellen wordt bepaald door de

informatie die de gebruiker wenst te zien.

0NV: Breng alle gegevens in één entiteit

1NV: Laat de velden die kunnen berekend worden uit andere velden weg

Bepaal de primaire sleutel

Verwijder repeterende groepen en steek deze in een nieuwe entiteit. Bij het afsplitsen van de repeterende groep wordt de sleutel van de oorspronkelijke groep meegenomen in de attributenlijst. Duid in deze nieuwe entiteit de primaire sleutel aan

2NV: Zoek naar ieder attribuut dat geen deel uitmaakt van de sleutel en die afhankelijk is van slechts een deel van de sleutel. Maak nieuwe entiteiten met gevonden attributen samen met dat deel van de sleutel waarvan ze afhankelijk zijn. Definieer primaire sleutel in de nieuwe entiteiten.

3NV: Zoek naar ieder niet-sleutelattribuut dat functioneel afhankelijk is van een ander niet-sleutelattribuut. Maak nieuwe entiteiten met gevonden attributen samen met het niet-

sleutel attribuut waar ze functioneel afhankelijk van zijn. Definieer primaire sleutels in deze nieuwe entiteiten.

**Eerste stap: Nulde normaalvorm (0NV)**

* Inventarisatie (van wat?)
* Alle tabellen zijn benoemd (vanuit ERD)
* Alle attributen zijn benoemd
* Attributen komen maar 1 keer voor
* Bepaal/identificeer primaire sleutels

**Tweede stap: Eerste normaalvorm (1NV)**

* Controleer/elimineer herhalende groepen in afzonderlijke tabellen
* Maak aparte tabel voor elke set gerelateerde gegevens
* Identificeer voor elke set gerelateerde gegevens een primaire sleutel
* Identificeer voor elke set gerelateerde gegevens een samengestelde sleutel

**Derde stap: Tweede normaalvorm (2NV)**

* Controleer/elimineer herhalende groepen in afzonderlijke tabellen
* Maak aparte tabel voor elke set gerelateerde gegevens
* Identificeer voor elke set gerelateerde gegevens een primaire sleutel (zie 1NV)
* Identificeer voor elke set gerelateerde gegevens een samengestelde sleutel (zie 1NV)
* Identificeer voor elke set gerelateerde gegevens een vreemde sleutel (foreign key)

**Vierde stap: Derde normaalvorm (3NV)**

* Controleer op herhalende en dubbele attributen (velden)
* Maak aparte tabel voor elke set gerelateerde gegevens
* Controleer op gegevens die regelmatig veranderen
* In voorkomende gevallen maak een nieuwe tabel
* Benoem sleutelvelden (foreign key)

### Implementatie model

Het doel van deze activiteit is om na het normaliseren te komen tot een fysiek ontwerp van de database (opslag van

***Datadictionary ERD***

|  |  |  |
| --- | --- | --- |
| ***Nr*** | ***KLasse*** | ***Omschrijving*** |
| 1 | LOGIN | File Name: Login\_Data |
| 2 | KLANT | File Name: Klant\_Data |
| 3 | EMAIL | File Name: Email\_Data |
| 4 | Website | File Name: teksten voor opbouw website (landingpage) |
| 5 | Header | File Name: Header\_Info voor opbouw website (landingpage) |
| 6 | Body | File Name: Body\_Info voor opbouw website (landingpage) |
| 7 | Footer | File Name: Footer\_Info voor opbouw website (landingpage) |

[

Benoem de bestanden zoals die zullen worden aangemaakt op een computer-systeem

]

### Relationeel Representatie model

De namen van de velden ten behoeve van de op te bouwen bestanden zullen in groepen worden benoemd. Zo nodig zullen ook de primary- en secundary –keys worden beschreven.

***Opslagstructuur database namen velden LOGIN\_DATA***

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nbr | Field name | From | To | Length | Type | Description |
| 1 | Klant\_Nummer |  |  | 6 | Num | Primary Key (Unique) |
| 2 | Klant\_Login\_Naam |  |  | 50 | Char |  |
| 3 | Klant\_Login\_Wachtwoord |  |  | 20 | Char |  |
| 4 | DatumTijd\_Aanmaak |  |  |  |  | Time\_Stamp |
| 5 | DatumTijd\_Update |  |  |  |  | Time\_Stamp |
| 6 | DatumTijd\_Delete |  |  |  |  | Time\_Stamp |

***Opslagstructuur database namen velden KLANT\_DATA***

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nbr | Field name | From | To | Length | Type | Description |
| 1 | Klant\_Nummer |  |  | 6 | Char | Primary Key (Unique) |
| 2 | Klant\_Naam |  |  | 50 | Char |  |
| 3 | Klant\_Adres |  |  | 50 | Char |  |
| 4 | Klant\_Huisnr |  |  | 10 | Char |  |
| 5 | Klant\_Postcode |  |  | 6 | Char |  |
| 6 | Klant\_Woonplaats |  |  | 50 | Char |  |
| 7 | Klant\_Email\_Adres |  |  | 50 | Char |  |
| 8 | DatumTijd\_Aanmaak |  |  |  |  | Time\_Stamp |
| 9 | DatumTijd\_Update |  |  |  |  | Time\_Stamp |
| 10 | DatumTijd\_Delete |  |  |  |  | Time\_Stamp |

***Opslagstructuur database namen velden EMAIL\_DATA***

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nbr | Field name | From | To | Length | Type | Description |
| 1 | Klant\_Nummer |  |  | 6 | Char | Primary Key |
| 2 | Klant\_Naam |  |  | 50 | Char |  |
| 3 | Klant\_Adres |  |  | 50 | Char |  |
| 4 | Klant\_Huisnr |  |  | 10 | Char |  |
| 5 | Klant\_Postcode |  |  | 6 | Char |  |
| 6 | Klant\_Woonplaats |  |  | 50 | Char |  |
| 7 | Klant\_Login\_Naam |  |  |  |  |  |
| 8 | Klant\_Login\_Wachtwoord |  |  |  |  |  |
| 9 | Klant\_Email\_Adres |  |  | 50 | Char |  |
| 10 | Email\_Tekst |  |  | 300 | Char | {bevat voorgaande als gewijzigde klantgegevens} |
| 11 | DatumTijd\_Aanmaak |  |  |  |  | Time\_Stamp |

***Opslagstructuur database namen velden Website\_Info***

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nbr | Field name | From | To | Lenght | Type | Description |
| 1 | Tekst |  |  | 300 |  | Website opbouw tekst |

***Opslagstructuur database namen velden Header\_Info***

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nbr | Field name | From | To | Lenght | Type | Description |
| 1 | Tekst |  |  | 300 |  | Website opbouw tekst |

***Opslagstructuur database namen velden Body\_Info***

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nbr | Field name | From | To | Lenght | Type | Description |
| 1 | Tekst |  |  | 300 |  | Website opbouw tekst |

***Opslagstructuur database namen velden Footer\_Info***

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nbr | Field name | From | To | Lenght | Type | Description |
| 1 | Tekst |  |  | 300 |  | Website opbouw tekst |

# Technisch ontwerp

## Inleiding

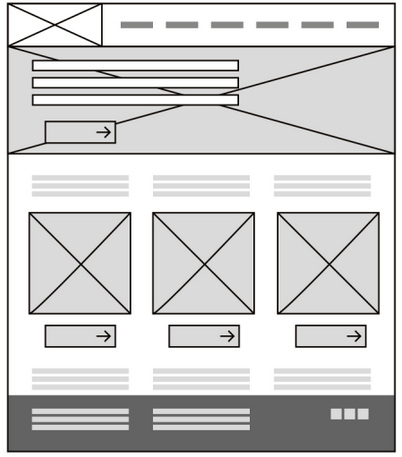
Dit hoofdstuk zal een gedetailleerde beschrijving bevatten van de technische specificaties van de te ontwikkelen applicaties inzake het *{Ontwikkel\_Systeem}*. Het is tevens een handvat voor de te bouwen applicaties ten behoeve van dit systeem met de hierna volgende inhoud:

* Wireframes
* Mock-ups
* Keuze Framework

## Wireframes

Wireframes is een bouwtekening van een website (schermprogramma) waarin een overzicht wordt gegeven van de verschillende onderdelen die op een website (schermprogramma) aanwezig zullen zijn.

Hieronder een voorbeeld:



Refereer puntgewijs de functionaliteiten vanuit de product-backlog (alle functionaliteiten uit het functioneel ontwerp)

## Mock-Ups

Een mock-up id een op schaal gemaakte model van een ontwerp of product. Voor een nog te bouwen website/schermprogramma maak je een “dummy website” zodat een goed beeld kan worden verkegen hoe de website (schermprogramma) in het echt eruit zal zien.

## Keuze framework

Een framework is een soort sjabloon (een voorbeeld/macro) voor het bouwen van een website (schermprogramma).

***De voordelen hiervan:***

* De basis bestaat al, je hoeft niet vanaf 0 te beginnen
* Het scheelt tijd
* Het zou een standaard kunnen zijn
* Het is minder foutgevoelig

# Bouw

In dit hoofdstuk worden een x-aantal uit te voeren stappen/activiteiten uitgewerkt.

## Architectuur Client/Server

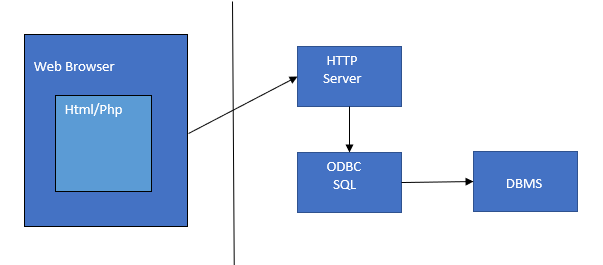
Middels een situatie-schets zal de client/server architectuur nader worden toegelicht. Een html-pagina presenteert de user-interface aan de gebruiker met daarin de functionaliteiten:

Aanroepen webpagina

Login

Vastleggen-, opvragen en muteren van gegeens

Web-Architectuur:



**Legenda:**

|  |  |  |
| --- | --- | --- |
| **Nr** | **Naam** | **Omschrijving** |
|  | Web Browser |  |
|  | HTTP |  |
|  | ODBC |  |
|  | DBMS |  |
|  |  |  |
|  |  |  |

## Bouw ontwerp modules

**Module- modellen:**

Het modulemodel komt tot uiting tijdens de bouw van de programmatuur en zal worden opgedeeld in een logische- en een fysieke module.

[

Hoe te beginnen:

* Ontwerp logische module
* logische modules -> Software Program Flow {program structure design}
* fysieke modules -> implementatie van de logische module {welke software taal}

]

## Logische module beschrijving

[

Maak een kopie van het concept klassendiagram, werk entiteiten/attributen bij zoals bestanden ontworpen volgens het ERD model.

Maak klassenmodel compleet voor het OOP model (voeg bij kenmerken de nodige methoden(functies) toe. (zie ook les documentatie ontwerpen en bouwen volgens OOP)

Beschrijvingen middels Program Flow diagrammen

]

## Fysieke module beschrijving

In dit hoofdstuk zullen de aan te passen cq nieuw te bouwen programmatuur worden beschreven.

**De Programmas**

|  |  |  |
| --- | --- | --- |
| Nr | Programmanaam | Omschrijving |
| 1 | Pgm\_naam\_html001 |  |
| 2 | Pgm\_naam\_html002 |  |

***Controles Pgm\_naam\_htlm001:***

De hierna volgende controles moeten worden uitgevoerd:

[

Speficiceer de controles voor invoer

]

# Versie beheer

Het op na houden van versie beheer. Dit betreft alle documenten, rapporten en programmatuur

# Testen

Maak van alle functionaliteiten (zie ook usecase, product backlog) een testplan en de per functie een test-scenario. Stel de test-scripten op.

# Opleveren (implementeren)

Lever het product op aan de klant (review).

* Houd bijvoorbeeld een demo of een presentatie. Ieder lid van het projectteam presenteert hierbij een of meer user stories.
* Stel naar aanleiding van de oplevering verbetervoorstellen voor het product op indien nodig.
* Bundel de verbetervoorstellen van de integratietest en review van jouw user stories in een document.

# Resultaat

Als resultaat van deze opdracht lever je de volgende producten en/of diensten op.

* Document uitgangspunten, eisen en wensen
* Notulen reflectiemeeting
* Applicatie/user stories
* Document met verbetervoorstellen

# Versie beheer

Revisie en opmerkingen

|  |  |  |  |
| --- | --- | --- | --- |
| Nr | Datum | Auteur | Omschrijving |
| 1 | 17-04-2023 | An | Initiële set-up / concept ontwerp |
| 2 |  |  |  |