**Hardstyle Shop**

A store with many clothes on swingers

Description automatically generated

MBO Rijnland Gouda

Versie : 0.1

Datum : 12 januari 2024

Auteur : Diana van Kampen

**Inhoud**

[1 Inleiding 3](#_Toc9770)

[2 Opstarten project 3](#_Toc2810)

[2.1 Ontvang opdracht van de opdrachtgever 4](#_Toc14561)

[2.1.1 Opdrachtomschrijving 4](#_Toc30611)

[2.1.2 Probleemstelling 4](#_Toc2554)

[2.1.3 Oplossing 4](#_Toc12349)

[2.2 Aanvullende informatie 4](#_Toc2623)

[2.3 Formuleer uitgangspunten, eisen en wensen 4](#_Toc9805)

[2.4 Afstemmen met betrokkenen 4](#_Toc25237)

[2.5 Maak een overzicht User Stories en een product Backlog 4](#_Toc4176)

[3 Planning en begroting 6](#_Toc25493)

[3.1 Budget 8](#_Toc2004)

[3.2 Begroting 8](#_Toc26271)

[3.3 Uitgaven 9](#_Toc13978)

[3.4 Begroting informatie en archivering 9](#_Toc21702)

[3.5 Verschil budget en uitgaven 9](#_Toc5719)

[4 Functioneel ontwerp 10](#_Toc27160)

[4.1 Uitvoeren iteraties 10](#_Toc8038)

[4.2 Context Klassen diagram (domein analyse) 10](#_Toc16384)

[4.3 Use case diagram met beschrijving (domein analyse) 11](#_Toc26134)

[4.3.1 Object Diagram {Ontwikkel\_Systeem} 13](#_Toc16341)

[4.3.2 Sequence-Diagram {Ontwikkel\_Systeem} 13](#_Toc25477)

[4.3.3 Toestands-Diagram {Ontwikkel\_Systeem} 15](#_Toc26329)

[4.4 Database Ontwerp (ERD) 17](#_Toc20001)

[4.4.1 Conceptueel model (ERD) 17](#_Toc20273)

[4.4.2 Normaliseren 18](#_Toc18235)

[4.4.3 Implementatie model 20](#_Toc19870)

[4.4.4 Relationeel Representatie model 20](#_Toc3682)

[5 Technisch ontwerp 22](#_Toc12800)

[5.1 Inleiding 22](#_Toc12377)

[5.2 Wireframes 22](#_Toc26855)

[5.3 Mock-Ups 23](#_Toc5853)

[5.4 Activiteiten diagram (UML) 23](#_Toc28356)

[Use case lijst 23](#_Toc18342)

[Use case: Project toevoegen 24](#_Toc18799)

[5.5 Keuze framework 26](#_Toc21559)

[6 Bouw 26](#_Toc8996)

[6.1 Architectuur Client/Server 26](#_Toc18819)

[6.2 Bouw ontwerp modules 27](#_Toc14883)

[6.3 De gebruikte Standaarden 27](#_Toc9589)

[6.4 Logische module beschrijving 27](#_Toc12953)

[6.5 Fysieke module beschrijving 27](#_Toc18895)

[7 Versie beheer 28](#_Toc1145)

[8 Testen 28](#_Toc24327)

[9 Opleveren (implementeren) 28](#_Toc19819)

[10 Reflecteren en Evalueren 28](#_Toc28425)

[11 Eind gesprek 29](#_Toc20227)

[12 Resultaat 29](#_Toc32517)

[13 Versie beheer 29](#_Toc25564)

# Inleiding

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

* Opstarten project
* Opdracht omschrijving
* Probleemstelling
* Oplossing
* Planning en begroting
* Functioneel ontwerp
* Technisch ontwerp
* Bouw ontwerp
* Testen
* Presenteren
* Implementeren (opleveren)
* Versie beheer
* Reflecteren
* Eind gesprek
* Resultaat

# Opstarten project

1. Ontvang de opdracht van de opdrachtgever.
2. Verzamel eventuele aanvullende informatie. Bijvoorbeeld door internet of de klant te raadplegen.
3. Formuleer de uitgangspunten, eisen en wensen voor het project in een document (bijvoorbeeld Definition of done). Doe dit volgens de geldende bedrijfsprocedures. Dit omvat bijvoorbeeld het volgende.
   1. Projectdoel: Wat wil de klant bereiken met het product?
   2. Doel van het team en eigen doelen (bijvoorbeeld security-/performance-eisen die niet binnen een user story passen).
   3. Te gebruiken technieken en codeconventies
4. Stem af met de betrokkenen over het document met uitgangspunten, eisen en wensen.
5. Stel op basis van het document de user stories op. Zet deze user stories op de Product backlog.

## Ontvang opdracht van de opdrachtgever

### Opdrachtomschrijving

Mevrouw Stephanie Magic wil haar Hardstyle shop uitbreiden naar een online shop. Ze wil de mensen die haar producten kopen een herinnering is van het moment dat ze naar hardstyle luisteren. Omdat er veel vraag is naar haar producten wilt ze het graag uitbreiden naar meerder shops en ook naar een website.

### Probleemstelling

Voor eventuele problemen:

Mensen kunnen ziek worden waardoor de project achter schema gaat lopen.

Dat sommige functies die niet na toe behoren gaan werken.

### Oplossing

Een oplossing voor planning is dat het bij sommige ruim gepland wordt en niet strak.

Ook voor functies kan er een alternatief gevonden worden zodat de website prima werkt.

## Aanvullende informatie

De klant geeft aan dat we haar logo gebruiken en dat er een aantal bedrijven zoals Defqon 1, Ql

Op de achtergrond de muziek Never Be Forgotten, Wild Ones en dat de video gebruikt wordt.

[Code Black feat. Matthew Steeper - Never Be Forgotten (Official Video) (youtube.com)](https://www.youtube.com/watch?v=RNdz9AinbUg&ab_channel=CodeBlack)

[Code Black feat. Elle Vee - Wild Ones (Official Video) (youtube.com)](https://www.youtube.com/watch?v=WQoN6fUcmb8&ab_channel=CodeBlack)

## Formuleer uitgangspunten, eisen en wensen

Uitgangspunten kunnen zijn:

Op welk systeem zal de programmatuur worden ontwikkeld

Zijn rapporten/documentaties aanwezig die hiervoor de nodige informatie kunnen verschaffen

Projectdoel

De opdrachtgever wil online producten verkopen om meer producten te laten zien dat zij maakt.

Doel van het ontwerp team:

Denk aan security, performance etc

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***

Een userstory wordt als volgt opgesteld:

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

Hieronder wat voorbeelden:

*“Als eigenaar wil ik een dashboard hebben waarop ik de producten kan zien, zodat ik weet hoeveel producten er zijn en welke ik bij moet bestellen.*

*“Als klant wil ik feedback/commentaar over de product kunnen geven, zodat ik aan kan geven of het product goed of slecht is.”*

*“Als eigenaar wil ik toegang hebben voor het inloggen, zodat ik producten kan toevoegen, bewerken of verwijderen.”*

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

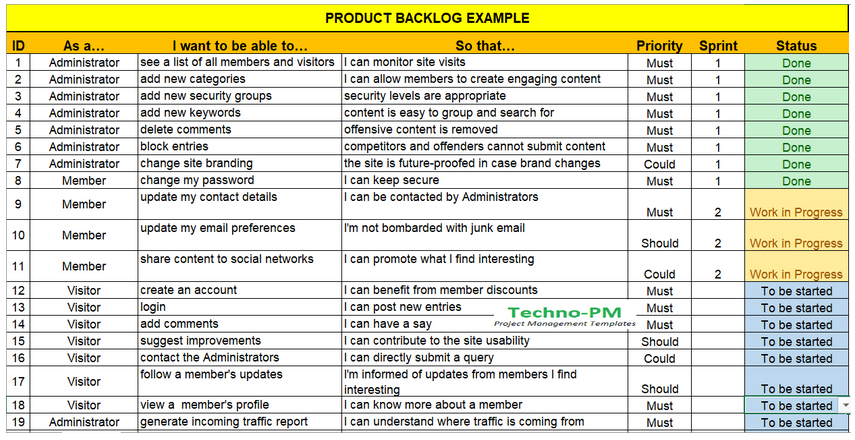
*“Als een forens wil ik real-time updates ontvangen over het openbaar vervoer, zodat ik op tijd op mijn bestemming aankom en mijn reistijd kan optimaliseren.”*

*“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

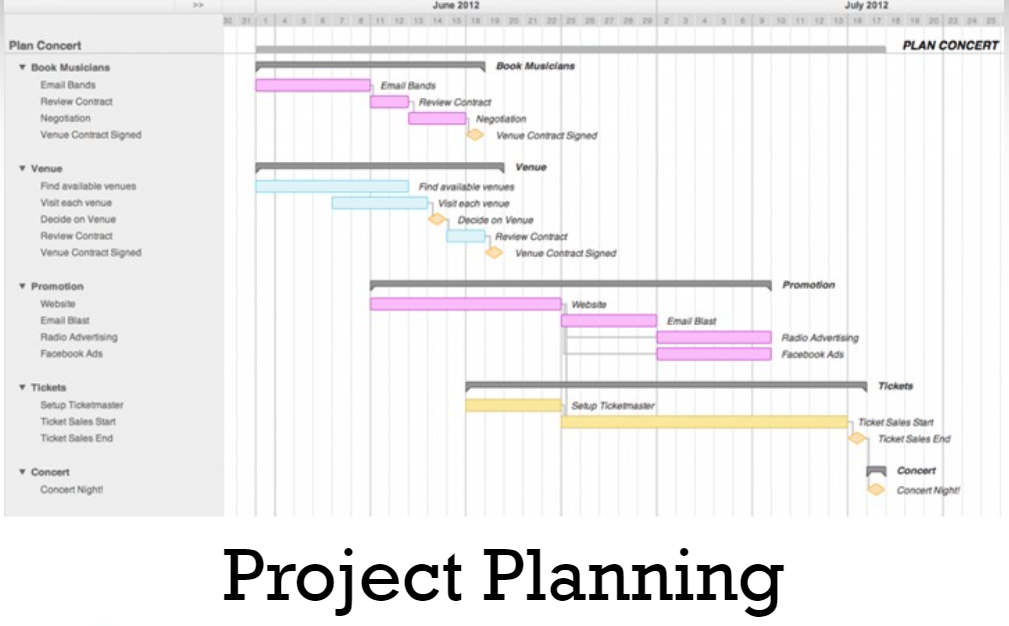
Automatisch gegenereerde beschrijving

**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:



**Een uitgebreide stappenplan om te komen tot een begroting**

Bron: https://werken-aan-projecten.nl/projectbegroting/begroting-voorbeeld-en-format/

## Budget

Als projectleider krijg je altijd een budget mee. Dit is de basis voor jouw begroting. Het is het bedrag dat je beschikbaar krijgt om het resultaat te maken. Soms krijg je een bedrag voor het hele project. In andere situaties krijg je een bedrag voor bepaalde fases en moet je daarna geld vragen voor de rest van het project.

## Begroting

1. **activiteiten**: dit zijn alle activiteiten die in je project worden uitgevoerd. Per activiteit bepaal je wat de kosten zijn. Zo kan je direct het effect op je hele begroting zien wanneer je activiteiten schrapt, toevoegt of hier extra;s uitvoert
2. **wie**: hier vul je in wie deze activiteit uitvoert. Je weet dan gelijk of voor deze uren geld in rekening moet worden gebracht of niet. Wanneer je wil weten hoeveel geld je aan een bepaalde externe partij gaat besteden kan je selecteren op de naam van deze partij
3. **bedrag**: hier vul je het bedrag in dat materiaal, een bepaald product etc. kost. Vraag offertes op of onderzoek wat de kosten waren bij vergelijkbare projecten. Bekijk wel of de omstandigheden in dat project hetzelfde waren als bij jouw project. Want dat kan invloed hebben op het bedrag dat je moet betalen
4. **uurtarief**: Bij veel projecten zijn de ingehuurde uren een grote kostenpost. Wanneer je mensen inhuurt neem je hier het bedrag in dat je per uur moet betalen
5. **aantal uren**: hier vul je in hoeveel uren je de externe persoon denkt nodig te hebben. Je kan dit aan de deskundige zelf vragen. Vraag altijd om meerdere offertes en vertel de ander dat je dat gaat doen. Je kan ook nagaan hoeveel uren in vergelijkbare projecten nodig waren
6. **btw**: een bedrijf moet altijd de btw betalen, maar kan deze later ook weer aftrekken bij de belastingdienst of doorberekenen aan de klant. Overheden hoeven in bepaalde gevallen geen btw te betalen. De afdeling financiën weet meestal wel wanneer dit het geval is
7. **totaal**: hier vindt je de totale kosten per activiteit

## Uitgaven

1. **betaald**: hou vul je per activiteit op je begroting in welke rekeningen je voor jouw project al hebt betaald
2. **verplichtingen**: hier vul je de kosten in waarvoor je wel al akkoord hebt gegeven, maar waarvoor je de rekening nog niet hebt betaald
3. **nog te verwachten**: hier vul je de kosten in die je nog niet hebt betaald en waarvoor je je nog niet hebt vastgelegd, maar waarvan je zeker weet dat ze er nog aankomen
4. **totale** **uitgaven**: hier zie je het bedrag dat je voor deze activiteit in totaal denkt te gaan uitgeven
5. **vrije ruimte**: hier vul je het verschil in tussen de totale begrote kosten en de totale verwachte uitgaven  
   hier zie je hoeveel geld je voor deze activiteit overhoudt of juist tekort komt. Als je geld overhoudt kan je dat gebruiken voor één van de andere activiteiten die mogelijk een tekort heeft. Anders kan het budget vrijgegeven worden voor de opdrachtgever, die het dan elders kan gebruiken.

## Begroting informatie en archivering

1. **laatst aangepast**: vul bovenaan het overzicht in op welke datum je dit voor het laatst hebt aangepast. Dat helpt je wanneer je niet zeker weet welke betalingen je nog wel en niet hebt doorgevoerd.
2. geef de begroting een titel: waar gaat deze begroting over?
3. Neem een paginanummer op
4. Geef versienummers aan.

## Verschil budget en uitgaven

Tenslotte zorg je dat altijd duidelijk is wat het verschil tussen de begrote en de werkelijke uitgaven is. Zo kan je snel zien of je nog op het goede spoor zit of dat je het project moet bijsturen.

# 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.

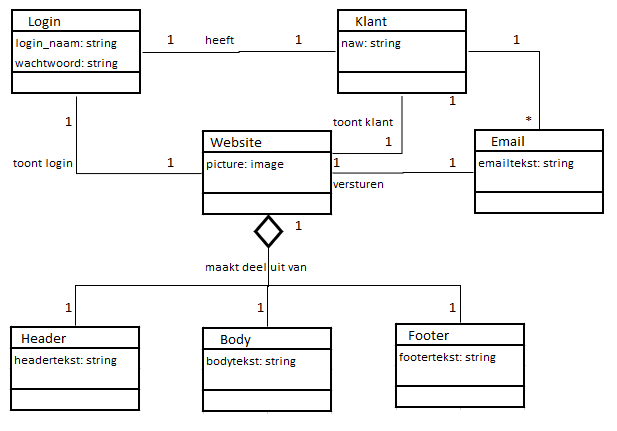
Verdeel binnen het team de eerste op te pakken user stories. Zorg voor een evenwichtige verdeling.

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

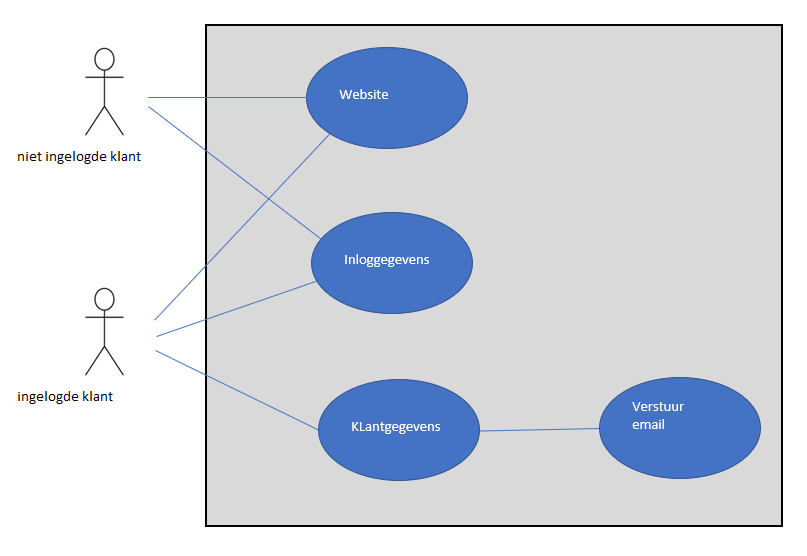
De actoren en use cases worden verbonden met pijlen

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 |  |

### Object Diagram {Ontwikkel\_Systeem}

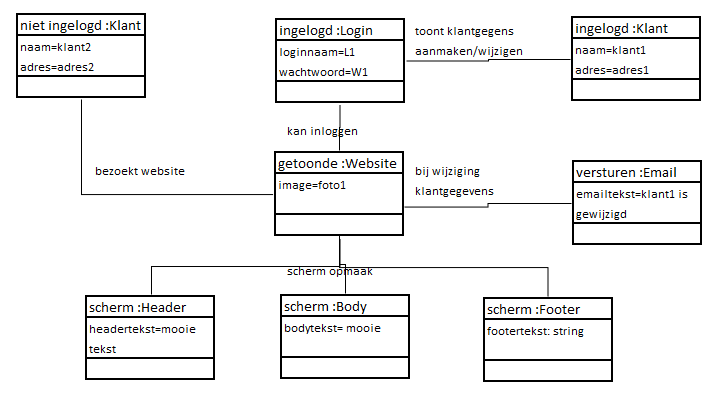
[

Een object diagram is niet verplicht, wel handig voor het uitleggen of als voorbeeld te gebruiken bij een iet-wat minder makkelijk te begrijpen (domein) klasse-diagram. Een object diagram is een snapshot/screenshot (moment-opname) van een entiteit (object in enkelvoud/of een record=object van een bestand=klassen/entiteit)

Hoe te beginnen:

* Maak een kopie van het klasse-diagram (context-diagram)
* Maak object inzichtelijk door het invullen van de attributen per object
* Een object-diagram stelt 1 record (info per object) voor

]



### Sequence-Diagram {Ontwikkel\_Systeem}

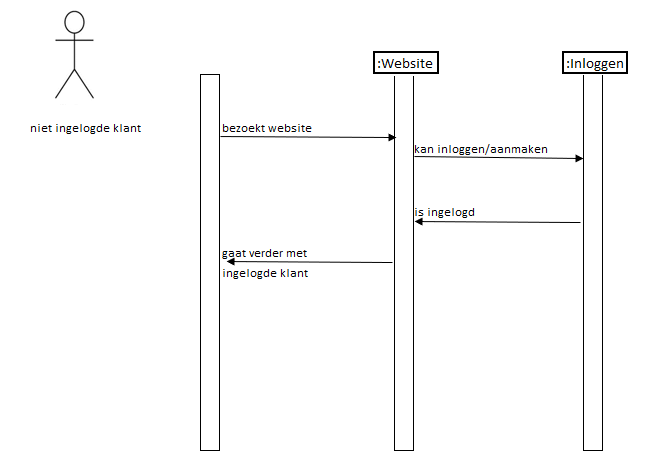
[

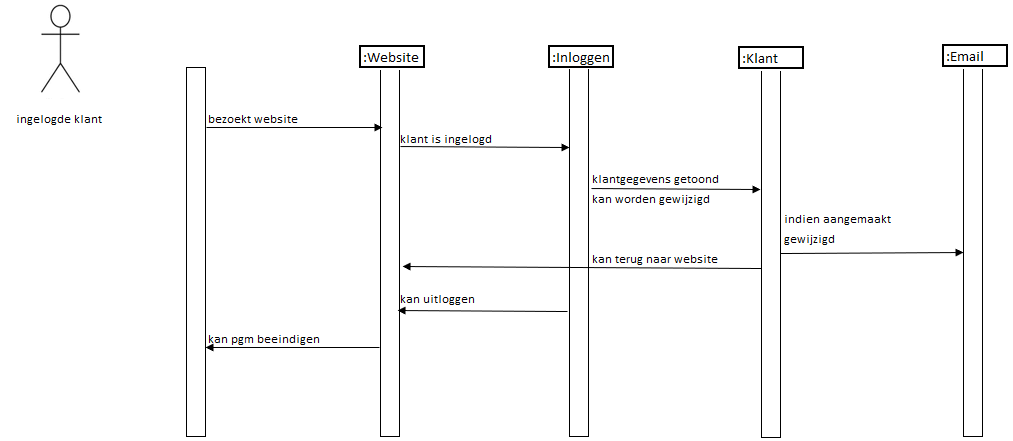
Uitgangspunt van een sequence diagram is het use-case diagram en toont de volgorde in de tijd, die in het systeem worden verstuurd of ontvangen. Een sequence diagram toont een interactie tussen de objecten. Een sequence diagram is een diagram op object (enkelvoud) niveau.

Hoe te beginnen:

* Bestudeer usecase diagram en het object diagram
* Per use-case 1 sequence diagram uitwerken
* Bepaal de volgorde van de onderlinge aktie van de objecten

]





### Toestands-Diagram {Ontwikkel\_Systeem}

Toestandsdiagrammen zijn activiteits diagrammen waarin alle toestanden zogenaamde activiteitstoestanden zijn. In elke activiteitstoestand voert het object een activiteit uit waarna het in een volgende toestand terecht komt. In feite is een toestand een stabiele staat waarin een object zich gedurende enige tijd bevindt.

[

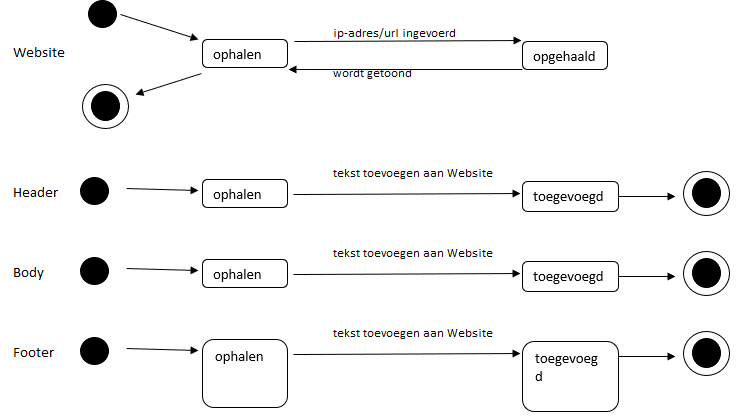
Hoe te beginnen:

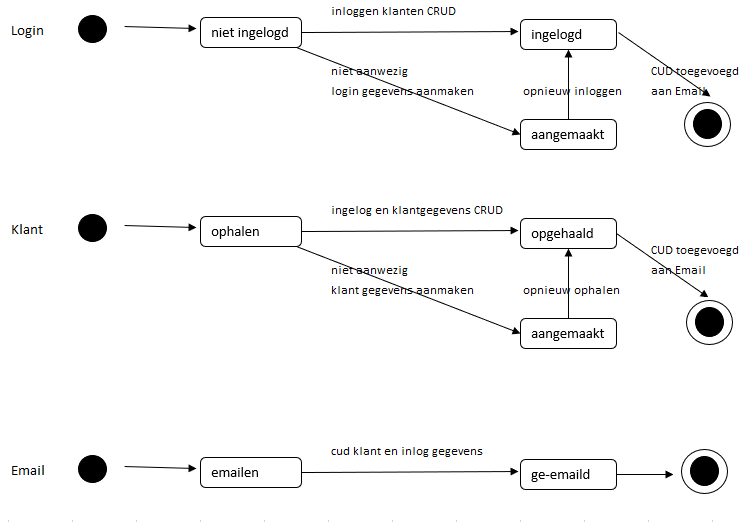
Bestudeer het klasse-diagram

Per klasse de toestand bepalen waarin zich een object zou kunnen verkeren

]

Met behulp van toestandsdiagrammen zullen wij de processen nader uitwerken met de bijbehorende pre- en post condities:





|  |  |
| --- | --- |
| ***Proces*** | ***Omschrijving*** |
| Website Ophalen  Precondities  Postcondities | Website opmaken  Pre-> Website is aanwezig en te benaderen  Post-> Website is opgemaakt en beschikbaar gesteld |
| Inloggen  Precondities  Postcondities | Inloggegevens aanmaken  Pre-> Inloginfo komt niet voor in bestand (klasse LOGIN)  Post-> Login\_info is aangemaakt in bestand (klasse LOGIN) |
| Header/Body/  Footer | Pre-> tekst gegevens zijn aanwezig in een tekst-bestand (klasse Header, Body en Footer  Post-> tekst gegevens zijn toegevoegd aan schermopmaak website |
| Klant Aanmaken  Precondities  Postcondities | Klantgegevens aanmaken  Pre-> Klantgegevens komt niet voor in bestand (klasse KLANT)  Post-> Klantgegevens aangemaakt in bestand (Klasse KLANT)  Post-> Klantgegevens aangemaakt in bestand (Klasse EMAIL) |
| Login Valideren  Precondities  Postcondities | Logingegevens controleren/valideren  Pre-> Logingegevens komen voor in bestand (klase LOGIN)  Post-> Logingegevens zijn gevalideerd |
| Klant Tonen  Precondities  Postcondities | Klantgegevens tonen nadat is ingelogd  Pre-> Klantgegevens komen voor in bestand (Klasse KLANT)  Post-> Klantgegevens worden getoond op het scherm |
| Klant wijzigen/  verwijderen  Precondities  Postcondities | Wijzigen Klantgegevens  Pre-> De te wijzigen/verwijderen klantgegevens komen voor in  Post-> Klantgegevens zijn gewijzigd in/verwijderd uit (KLANT)  Post-> Klantgegevens aangemaakt in bestand EMAIL\_DATA |
| Emailen  Precondities  Postcondities | Gewijzigde klantgegevens worden ge-emaild  Pre-> Klantgegevens komen voor in bestand (klasse EMAIL)  Post-> Klantgegevens zijn ge-emaild |

## 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:

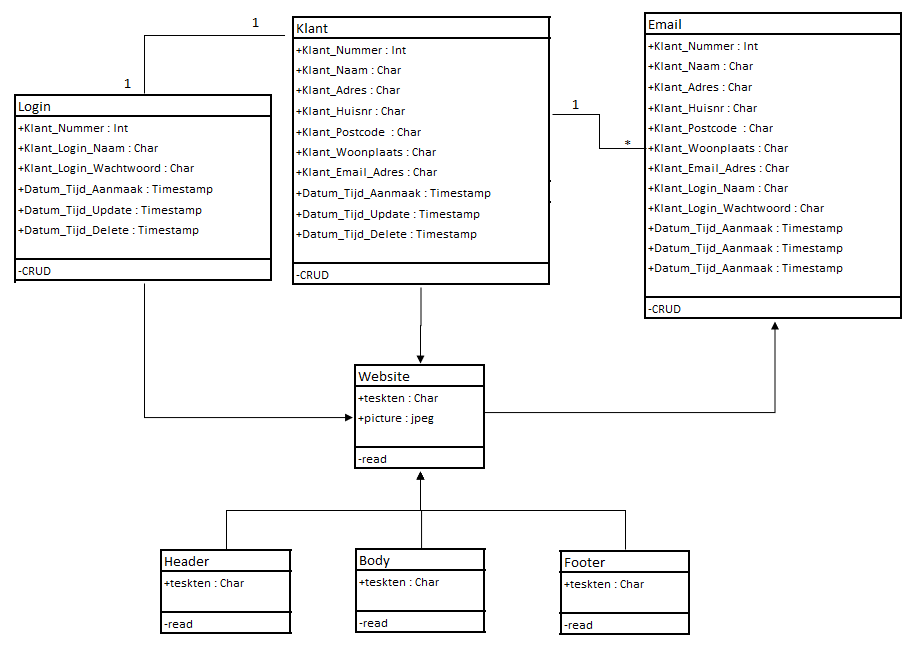
1 op 1 relatie

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
* Activiteiten diagram
* Keuze framework

## Wireframes

Hieronder zijn de wireframe voor de website:

A screenshot of a web page

Description automatically generated

Home pagina

De home pagina is waar het begint en laat 2 video’s zien, stukje tekst over de video’s, navigatie naar andere pagina’s, inlog button

A screenshot of a computer

Description automatically generated

Product pagina

Hier krijg je te zien de producten dat Stephanie Magic wilt verkopen aan de klant.

A screenshot of a computer

Description automatically generated

Contact pagina

Hier kan de klant contact gegevens vinden en zien waar de winkels zijn.

A screenshot of a computer

Description automatically generated

Checklist pagina

Hier zie je de overzicht van de klant die hij of zij wilt bestellen voor dat hij of zij gaat bestellen.

A screenshot of a computer

Description automatically generated

Thank you pagina

Bij de thank you pagina krijgt de klant te zien dat hij of zij succesvol betaalt heeft.

A screenshot of a web page

Description automatically generated

Inlog pagina

De klant, medewerker(s) en eigenaar kunnen hier inloggen om bepaalde handeling te kunnen doen zoals voor klant producten bestellen of als eigenaar/medewerker producten toevoegen, bijwerken of verwijderen.

A screenshot of a computer

Description automatically generated

Registratie pagina

Hier kan de klant aanmelden voor een account als hij of zij die nog niet heeft.

A screenshot of a computer

Description automatically generated

Dashboard

Hier kan de eigenaar en/of medewerker producten toevoegen, bijwerken of verwijderen. De eigenaar kan de gegevens van de medewerker toevoegen, bijwerken of deactiveren. Ook is er een grafiek te zien over de omzet.

Bron van de wireframes:

* <https://wireframe.cc/Hoj8mC> : home pagina
* <https://wireframe.cc/pfbJwM> : product pagina
* <https://wireframe.cc/KGc5Sk> : contact pagina
* <https://wireframe.cc/27Wd0u> : checklist pagina
* <https://wireframe.cc/PItcfr> : thankyou pagina
* <https://wireframe.cc/XUR35d> : inlog pagina
* <https://wireframe.cc/c6nJbk> : registratie pagina
* <https://wireframe.cc/pOyp4e> : dashboard pagina

## Mock-Ups

Kleuren die ze wilt zijn rood, zwart en wit. De kleuren van haar logo.

Er is nog een kleur dat ze in gedacht heeft en dat is blauw.

Background: #353434

Second background: #979595

A screenshot of a web page

Description automatically generated

Home pagina

A screenshot of a computer

Description automatically generated

Product pagina

A screenshot of a web shop

Description automatically generated

Checklist pagina

A screenshot of a website

Description automatically generatedContact paginaA screenshot of a computer

Description automatically generated

Dashboard pagina

## Activiteiten diagram (UML)

Op basis van een “Use Case” met beschrijving maak je een “activiteiten diagram”. Het is een soort stroomdiagram waarmee de activiteiten worden gemodelleerd. Een activiteiten diagram heeft altijd een begin-toestand en een eind-toestand.

Hieronder een voorbeeld hoe vanuit een usecase een actitiveiten diagram op te stellen:

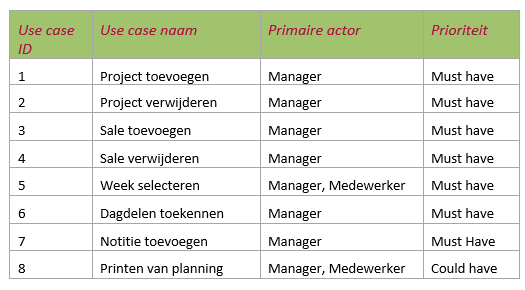


## Use case lijst

Onderstaande tabel toont de use cases die zijn beschreven in de zogenaamde hoofdfuncties. Hierbij wordt aangegeven welke actor gebruik maakt van deze functies en welke prioriteit deze functie krijgt binnen het project.

De functies worden vervolgens uitgewerkt in use cases. Bij elke use case wordt een activiteiten diagram getoond ter verduidelijking van de functie.

Alleen de functies met de prioriteit ‘Must have’ zijn uitgewerkt tot een usecase en activiteiten diagram.



### Use case: Project toevoegen

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case ID:** | 1 | | |
| **Use Case Naam:** | Project toevoegen | | |
| **Opgesteld door:** | Danny Vosman | **Laatst bijgewerkt door:** | Danny Vosman |
| **Datum gemaakt:** | 1 oktober | **Datum bijgewerkt:** | 9 december |
| **Actoren:** | Manager | | |
| **Omschrijving:** | Het toevoegen van een project aan de projectplanning. De actor selecteert een project vanuit de beschikbare projecten. | | |
| **Precondities:** | - | | |
| **Hoofdscenario**: | 1. Actor geeft zoek waarden voor project. 2. Systeem weergeeft zoekresultaten 3. Actor selecteerd project 4. Systeem sluit popup en voegt project toe aan planning | | |
| Postcondities: | Project toegevoegd aan planning | | |
| **Alternatief scenario:** | 1.5 Systeem sluit popup | | |
| Precondities: | Actor sluit popup zonder selectie | | |
| Postcondities: | Geen | | |
| **MoSCoW prioriteit:** | Must have | | |
| **Opmerkingen en issues:** |  | | |

#### Ativiteiten diagram: Project toevoegen



## 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

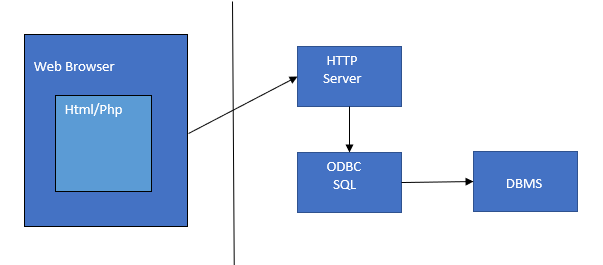
Middels een situatie-schets zal de client/server architectuur nader worden toegelicht. Een html-pagina presenteert de user-interface aan de gebruiker met daarin de functionaliteiten:

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:

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

]

## De gebruikte Standaarden

***IT Standaard en kenmerken ontwikkel omgeving:***

De bouw van het Transport *{Ontwikkel\_systeem}* zal plaatsvinden op een {specificeer hard-/software waarop programmatuur zal worden geprogrammeerd}

[

Specificeer hard- en software waarop zal worden geprogrammeerd

Specificeer kenmerken hard- en software

Specificeer te gebruiken tools

Specificeer te gebruiken database

]

***De benaming***

De benaming volgens standaarden binnen softwareontwikkeling impliceert het volgende:

De aan te maken databases en programmatuur zullen beginnen met….

***Beschrijving Menustructuur:***

Benoem de menu opties/tabs etc voor zover van toepassing

## 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.

# Reflecteren en Evalueren

**Reflecteren**

Houd een reflectiemeeting (retrospective).

* Stel naar aanleiding van de reflectiemeeting verbetervoorstellen op voor het proces in de volgende iteratie/sprint indien nodig.
* Verwerk de verbetervoorstellen in nieuwe epics/user stories en zet deze op de Sprint backlog.

**Evalueren**

Bij projectevaluatie kijk je terug naar het resultaat van het project en de manier waarop je dat hebt bereikt. Zo krijg je inzichten die je helpen om het volgend project nog beter uit te voeren

# Eind gesprek

Individueel technisch inhouden gesprek over je opgeleverde werk (max.10 minuten)

# 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 | 16-04-2023 | An | Initiële set-up / concept ontwerp |
| 2 |  |  |  |