Michelle Broens

S33 | GSO

KillerApp –

SAD

# Document historie

|  |  |  |  |
| --- | --- | --- | --- |
| **Versie** | **Wijzigingen** | **Auteur** | **Datum** |
| **1.0** | Document opgezet d.m.v. template in grote lijnen in te vullen | Michelle Broens | 15/11/2017 |

Inhoudsopgave

[Document historie 1](#_Toc498527099)

[Inleiding 3](#_Toc498527100)

[Domeinmodel 4](#_Toc498527101)

[Diagram 4](#_Toc498527102)

[Bijlage 4](#_Toc498527103)

[Afbakening 4](#_Toc498527104)

[Opdeling in componenten 5](#_Toc498527105)

[Componenten diagram 6](#_Toc498527106)

[bijlage 6](#_Toc498527107)

[Koppeling tussen componenten 6](#_Toc498527108)

[Synchronisatie tussen componenten 6](#_Toc498527109)

[Services per component 6](#_Toc498527110)

[Allocatie van objecten 6](#_Toc498527111)

[Remote objecten 6](#_Toc498527112)

[Packagestructuur 6](#_Toc498527113)

[Communicatie tussen componenten 7](#_Toc498527114)

[Klassendiagram 7](#_Toc498527115)

[Bijlage 7](#_Toc498527116)

[Persistentie per component 8](#_Toc498527117)

[Detailed design per component 9](#_Toc498527118)

[Realisatie niet-functionele eisen 10](#_Toc498527119)

[Deployment 11](#_Toc498527120)

[deployment diagram 11](#_Toc498527121)

[Bijlage 11](#_Toc498527122)

[Specificatie van interfaces 12](#_Toc498527123)

# Inleiding

*In dit architectuur document worden aanwijzingen gegeven in italic (schuinschrift). Een voorbeelduitwerking van het architectuur document is gegeven in* roman (rechte letters)*.*

*Beschrijf in de inleiding:*

* *De context van het project*
* *De applicatie*
* *De niet-functionele eisen met betrekking tot betrouwbaarheid, performance, beveiliging en schaalbaarheid*
* *Het doel van dit document*

**Context**

**Applicatie**

**Niet-functionele eisen**

**Doel van dit document**

# Domeinmodel

*In dit hoofdstuk worden een of meer klassendiagrammen van het domein getoond. Daarnaast wordt een toelichting gegeven en worden gemaakte keuzes verantwoord. Tot slot wordt de afbakening van het project besproken.*

## Diagram

## Bijlage

## Afbakening

# Opdeling in componenten

*In dit hoofdstuk wordt de opdeling in softwarecomponenten besproken aan de hand van een Componentendiagram met toelichting. Daarnaast wordt koppeling en synchronisatie tussen de componenten besproken en de opdeling in software packages.*

*Een component is een modulair deel van het systeem dat beschreven wordt in termen van aangeboden en gevraagde interfaces. In principe kan een component vervangen worden door een andere component, mits die andere component aan dezelfde interfaces voldoet. Er wordt onderscheid gemaakt tussen componenten en subsystemen: een component is altijd een executeerbare eenheid, maar een subsysteem is geen executeerbare eenheid. Tip: kies voor iedere component een naam die duidelijk verschilt van de namen van de klassen die in die component zitten.*

*De volgende onderwerpen worden in dit hoofdstuk besproken:*

* *Componentendiagram (opdeling componenten)*
* *Soort koppeling (RMI, HTTP, SQL, …)*
* *Synchronisatie (zie ook persistentie/communicatie)*
* *Service(s) per component (voor welke component interessant; aanduiding van behoeften (naam methode) per interface)*
* *Allocatie objecten (binnen welke component)*
* *Remote objecten (welke objecten worden op afstand aangesproken)*
* *Packagestructuur*

*Er zijn verschillende manieren om componentdiagrammen te tekenen, zelfs binnen UML. Hier is gekozen voor componenten opgebouwd uit klassen en interfaces. Klassen kunnen interfaces realiseren of hebben behoefte aan interfaces. Deze interfaces kunnen binnen een component met elkaar verbonden worden en/of via een zogenaamde poort beschikbaar worden gesteld aan andere componenten. Een poort (engels: port) wordt weergegeven door middel van een vierkantje op de rand van de component. Een poort kan een interface aanbieden (aangeboden poort) of behoefte hebben aan een interface (benodigde poort). Een poort die zowel een interface aanbiedt als een interface nodig heeft noemen we een complexe poort. Meestal wordt een poort gerealiseerd door middel van een object dat de aanvraag van services doorgeeft naar een ander object binnen de component. Tijdens uitvoering van het programma zal een benodigde poort een verbinding leggen met een aangeboden poort van een andere component. Voor meer informatie over componentendiagrammen, zie Hoofdstuk 14 van Praktisch UML van Jos Warmer en Anneke Kleppe (vijfde editie, uitgever Pearson).*

## Componenten diagram

## bijlage

## Koppeling tussen componenten

## Synchronisatie tussen componenten

## Services per component

## Allocatie van objecten

## Remote objecten

## Packagestructuur

# Communicatie tussen componenten

*In dit hoofdstuk wordt de communicatie van en tussen objecten van verschillende componenten beschreven. Beschrijf welke eigenschappen van welke objecten moeten worden gecommuniceerd. Beschrijf hoe communicatie wordt gerealiseerd (bijvoorbeeld m.b.v Remote Method Invocation, RMI). Dit kan verschillen per eigenschap. Beschrijf tot slot wanneer communicatie plaatsvindt (push-pull, welk object is leidend? etc.) Ook dit kan verschillen per eigenschap.*

*In het geval RMI wordt gebruikt voor communicatie tussen componenten: stel een klassendiagram op en geef per klasse/interface aan of deze UnicastRemoteObject, Remote of Serializable is. In het geval gebruik gemaakt wordt van (SOAP) webservices: stel een klassendiagram op en geef aan welke services door welke klasse worden geleverd.*

## Klassendiagram

## Bijlage

# Persistentie per component

*In dit hoofdstuk wordt de persistentie van de objecten beschreven voor iedere component. Beschrijf welke eigenschappen van welke objecten moeten worden bewaard. Beschrijf hoe opslag wordt gerealiseerd (database, serialiseren, XML, etc.). Dit kan verschillen per eigenschap. Voeg het ERD met beschrijving toe indien gegevens worden opgeslagen in een database. Beschrijf tot slot wanneer opslag plaatsvindt. Ook dit kan verschillen per eigenschap.*

# Detailed design per component

*In dit hoofdstuk wordt het detailed design (implementatiemodel) per component beschreven. Maak hiervoor gebruik van klassendiagrammen en sequence diagrammen. Voor specifieke doeleinden kun je gebruik maken van objectdiagrammen, statemachinediagrammen, activiteitendiagrammen, etc. Voor meer informatie over UML-diagrammen, zie Praktisch UML van Jos Warmer en Anneke Kleppe (vijfde editie, uitgever Pearson).*

# Realisatie niet-functionele eisen

*In dit hoofdstuk wordt de realisatie van (overige) niet-functionele eisen besproken. De volgende onderwerpen dienen tenminste te worden besproken: betrouwbaarheid, performance, beveiliging, schaalbaarheid. Eventueel kunnen nog extra niet-functionele eisen worden besproken.*

**Betrouwbaarheid**

**Performance**

**Beveiliging**

**Schaalbaarheid**

# Deployment

*In dit hoofdstuk wordt de toekenning van softwarecomponenten aan hardware besproken aan de hand van een Deploymentdiagram met toelichting. Daarnaast wordt de koppeling (lokaal netwerk, internet, etc.) tussen de nodes besproken.*

## deployment diagram

## Bijlage

# Specificatie van interfaces

*In dit hoofdstuk wordt de specificatie van interfaces besproken. Voor ieder interface wordt per methode gedefinieerd:*

* Naam methode
* Naam en type argumenten
* Precondities
* Type returnwaarde
* Beschrijving
* Aanleiding voor excepties