**Software Architectuur Document (SAD)**

Project: MyKoiPond

Projectteam: Rik

Versie: 0.2

Versiedatum: 26 November 2019

Status: Concept

# Documenthistorie

|  |  |  |  |
| --- | --- | --- | --- |
| Versie | Wijzigingen | Auteur | Datum |
| 0.1 | Opzet document | Nico Kuijpers | 14 dec 2017 |
| 0.2 | Beschrijving specificatie van interfaces aangepast. | Rik Peeters | 11 Feb 2020 |
| 0.3 | Beschrijving C2, C3, C4 en specificatie van interfaces aangepast. | Rik Peeters | 11 Feb 2020 |

Inhoud

Documenthistorie 2

H1 Inleiding 4

H2 Systeem Context (C1) 5

H3 Containers en technologiekeuzes (C2) 6

H4 Componenten (C3) 7

H5 Klassendiagrammen en sequence diagrammen (C4) 8

H6 Persistentie per component 9

H7 Specificatie van interfaces 10

# H1 Inleiding

In dit document wordt de globale technische implementatie en architectuur van de applicatie uiteengezet. Voor het domeinmodel en functionele requirements verwijzen we u naar het analysedoucment waarin deze worden beschreven.

De applicatie gaat dienen als een app waarmee je gegevens van een koi vijver kunnen worden opgeslagen, hierbij kun je denken aan ph waarden maar ook ziektes van een vis en over welke parasieten hij geschikt. Het moet dus mogelijk worden externe apparaten te koppelen dit gaat de applicatie mogelijk maken door gebruik te maken van een Rest-API. Hiermee kan de gebruiker een inzicht krijgen hoe het met zijn koi vijver gaat. Dit moet de omgang en het behouden van gerelativeerde gegevens vermakelijke waardoor de eindgebruiker nog meer van zijn hobby kan genieten.

De frequente communicatie tussen het backend en front-end (Vue) verloopt via web een spring boot REST API.

# H2 Systeem Context (C1)

In dit hoofdstuk wordt de context van de applicatie beschreven aan de hand van een context diagram.

*Afbeelding met schermafbeelding

Automatisch gegenereerde beschrijving*

# H3 Containers en technologiekeuzes (C2)

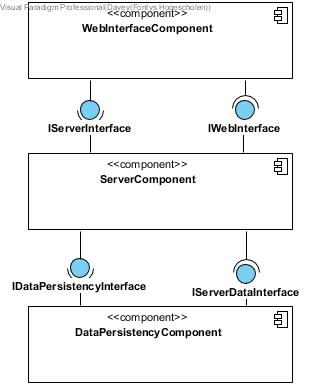
De MyKoiPond applicatie maakt gebruik van 3 containers. Als front-end wordt Vue js als technologie gebruikt. Hier is voor gekozen omdat met dit framework makkelijk realtime web-applicaties ontwikkeld kunnen worden. Vue JS is een stand allone server die draait op node JS als deze in productie wordt gebracht draait deze op apache. Frequente communicatie gaan via REST interfaces verlopen. Het back-end communiceert vanuit de Data Access Layer met de Persistence Layer m.b.v. REST interfaces. Hier is voor gekozen om de database veilig te stellen door er een extra laag aan toe te voegen. Ook is het hierdoor makkelijker om, indien van overbelasting van het back-end, meerdere back-ends te gebruiken.

Afbeelding met schermafbeelding

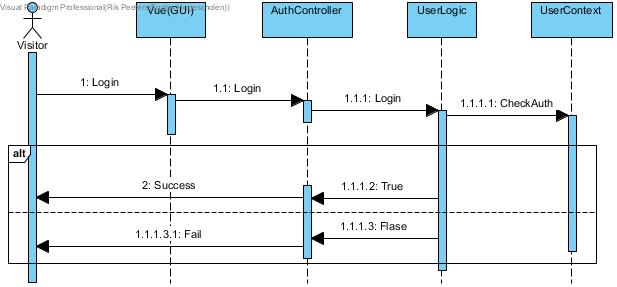
Automatisch gegenereerde beschrijving

# H4 Componenten (C3)

De MyKoiPond applicatie bestaat uit 3 componenten. De WebInterface wordt gebruikt door de gebruiker en communiceert met het server component. Het server component communiceert met het DataPersistency component. Het DataPersistency component bevat de MySQL-database. De database is dus losgekoppeld van de server.

**

# H5 Klassendiagrammen en sequence diagrammen(C4)

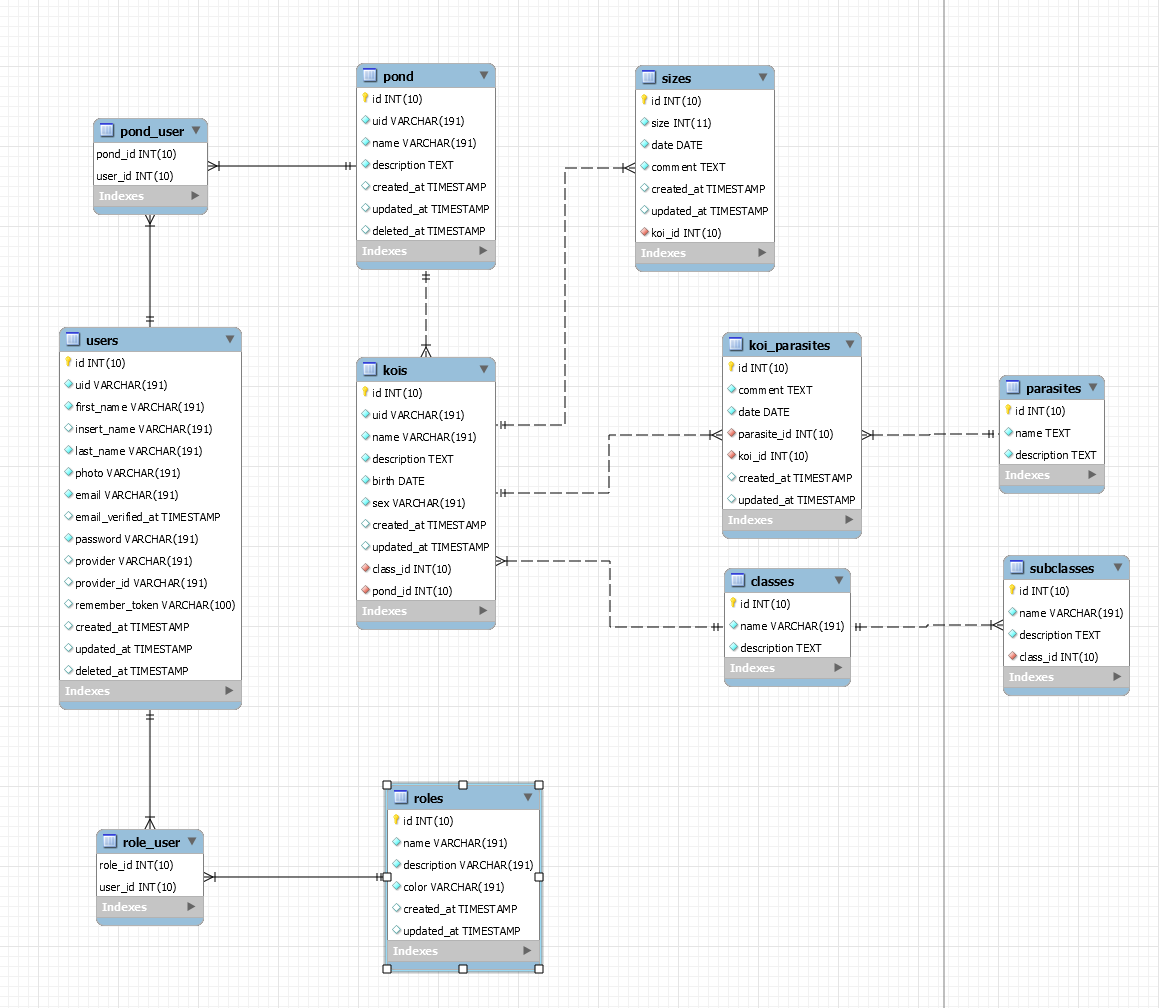
**Afbeelding met kaart

Automatisch gegenereerde beschrijving**

# H6 Persistentie per component

De applicatie zal persistentie hebben deze wordt gerealiseerd doormiddel van een mysql database. De objecten die worden opgeslagen staan in het klassendiagram. Hieronder is een ERD-diagram te vinden van de relaties tussen de verschillende tabellen.

ERD Diagram



# H7 Specificatie van interfaces

De implementatie van de koi is volgens de crud standaard alle opeenvolgende crud objecten volgen dezelfde endpoints standaard.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Protocol | URL | Returns |
| GetKoiList | Get | /koi | List of Koi |
| GetKoi | Get | /koi/{id} | Found Koi |
| CreateKoi | Post | /koi | Created Koi |
| UpdateKoi | Put | /koi/{id} | Updated Koi |
| DeleteKoi | Delete | /koi/{id} | Deleted Koi |