**Software Architectuur Document (SAD)**

Project: Pharmapartners Medicom Agenda

Project team: Big Pharma

Teamleden: Sander Berntsen (3239780)

Luuk Vermeer (3304698)

Simon Westerburger (3274209)

Jay van Helderen (2742373)

Jochem Wienk (389746)

Kelly van der Hoek (2461900)

Opdrachtgever: Pharmapartners

Versie: 0.3

Versie datum: 24-03-2021

Status: Concept

# Documenthistorie

|  |  |  |  |
| --- | --- | --- | --- |
| Versie | Wijzigingen | Auteur | Datum |
| 0.1 | Opzet document | Jay van Helderen, Sander Berntsen, Jochem Wienk | 11 feb. 2021 |
| 0.2 | C1, C2, C3 toegevoegd | Jay van Helderen, Sander Berntsen, Jochem Wienk | 11 feb. 2021 |
| 0.3 | C4 geüpdatet | Simon Westerburger | 24 mrt. 2021 |
| 0.4 |  |  |  |

Inhoud

[Documenthistorie 2](#_Toc63942947)

[Inleiding 4](#_Toc63942948)

[Systeem Context (C1) 5](#_Toc63942949)

[Containers en technologiekeuzes (C2) 6](#_Toc63942950)

[Componenten (C3) 7](#_Toc63942951)

[Klassendiagrammen en Sequence diagrammen (C4) 8](#_Toc63942952)

[Persistentie per component 9](#_Toc63942953)

[Specificatie van interfaces 10](#_Toc63942954)

# Inleiding

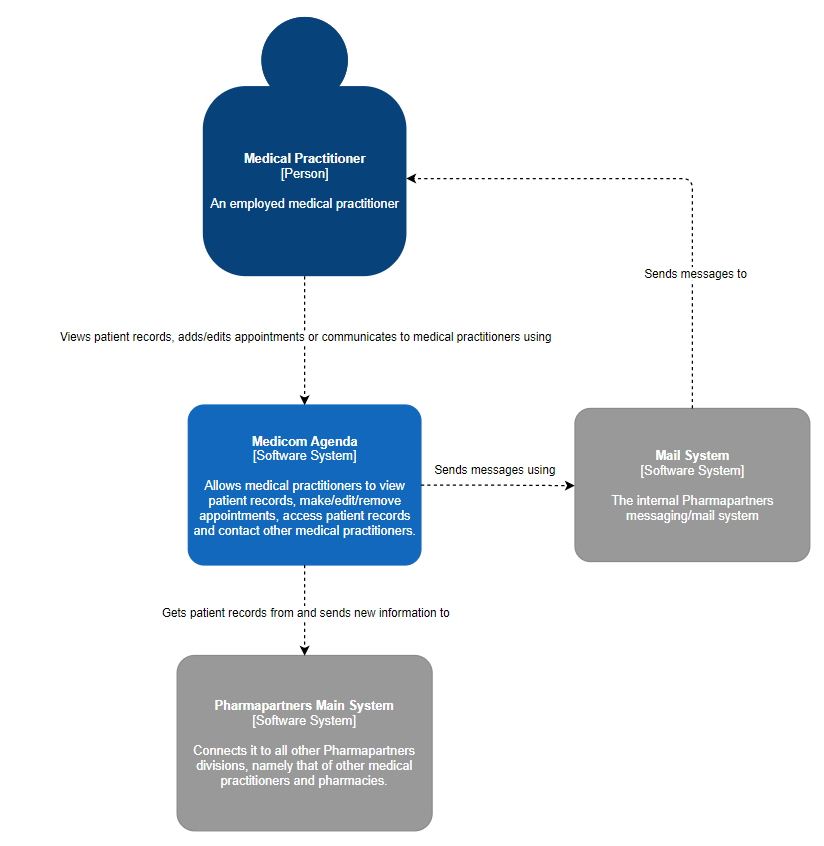
Dit document legt de software architectuur van het Medicom Agenda project uit. Het systeem wordt gebruikt door met name huisartsen om afspraken met klanten in te plannen en om zo ook informatie beschikbaar te maken voor andere doctoren en/of apotheken.

Verdere informatie over de requirements, het domeinmodel, etc. kunt u vinden in het Analysedocument.

# Systeem Context (C1)

*In dit hoofdstuk wordt de context van de applicatie beschreven aan de hand van een context diagram. Gebruik hiervoor het Data Flow Diagram. Je kunt dit diagram tekenen met Visual Paradigm (zie Business Modeling).*

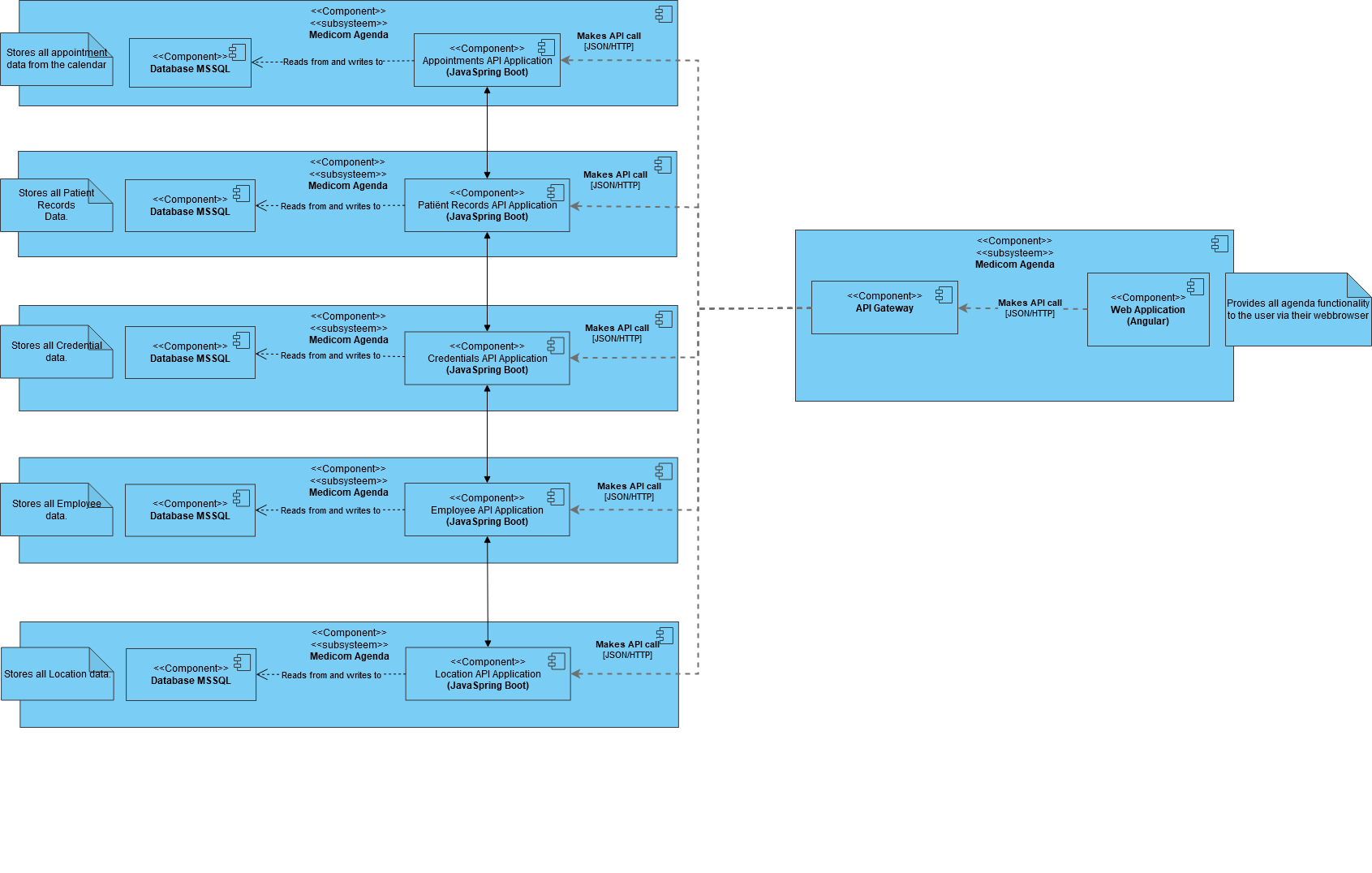
*Teken de applicatie als een Process (afgeronde rechthoek) in het midden en teken gebruikers en andere systemen als External Entity (rechthoek) eromheen. Geef een database aan door middel van een Data Store. Geef de flow van data aan door middel van pijlen (Data Flow en Bidirectional Data Flow). Plaats tekst bij de pijlen zodat de betekenis van de pijlen duidelijk wordt.*



# Containers en technologiekeuzes (C2)

*In dit hoofdstuk wordt de globale architectuur van het systeem beschreven aan de hand van containers en technologiekeuzes. Gebruik het UML Deployment Diagram om de verschillende hardwaresystemen (containers) te beschrijven met hun onderlinge communicatie. Geef op de verbindingslijnen aan welke communicatietechniek wordt gebruikt (REST, Websockets, etc.).*

*Beschrijf tevens de verantwoording van je technology keuzes (c.q. waarom heb je voor welke technologie gekozen voor weke communicatie).*

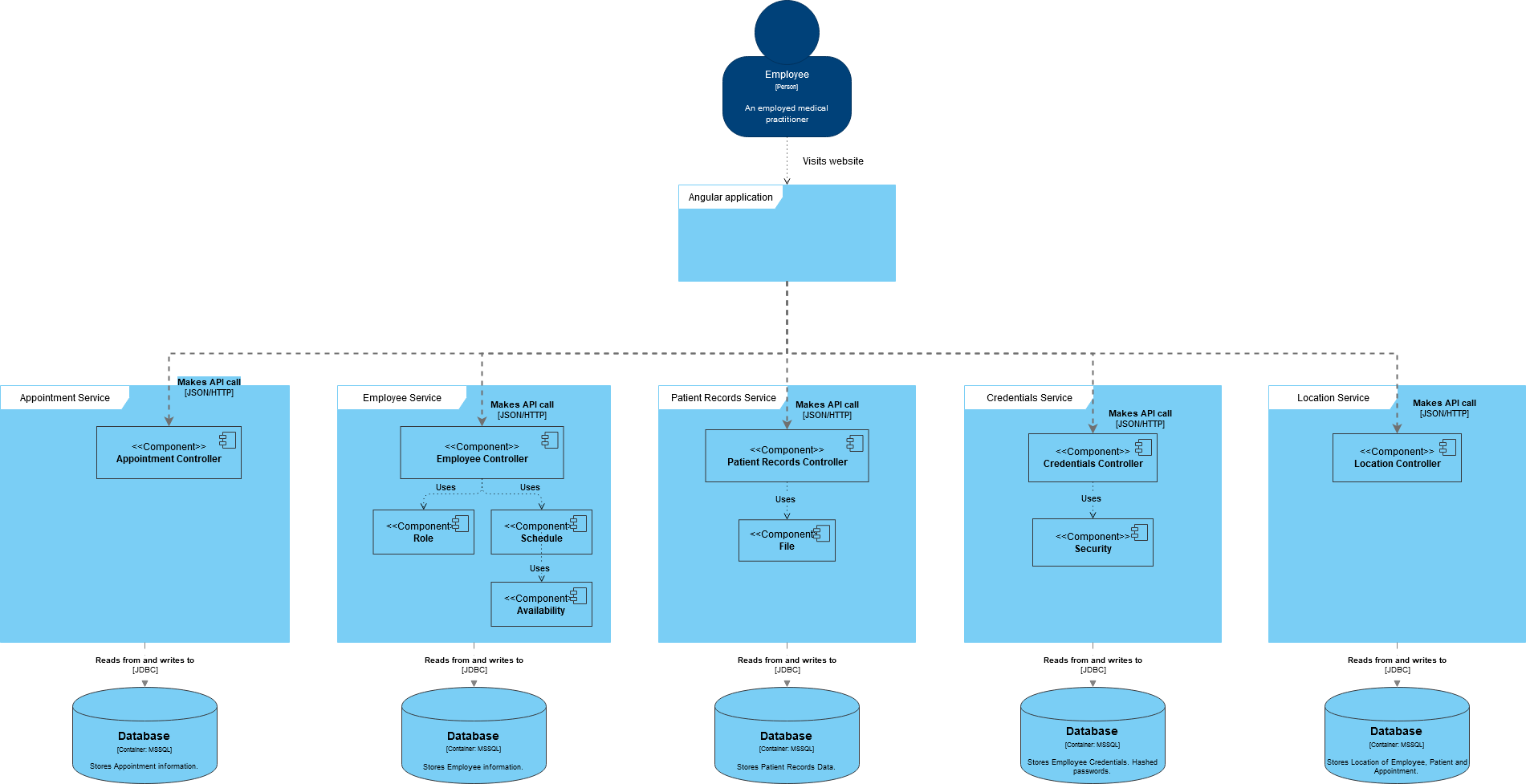


# Componenten (C3)

*In dit hoofdstuk wordt voor iedere hardware container besproken hoe de opdeling in softwarecomponenten is aan de hand van een componentendiagram met toelichting. 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. Geef een globale beschrijving voor iedere component en geef een beschrijving van de aangeboden en gevraagde interfaces. Kies voor iedere component een naam die duidelijk verschilt van de namen van de klassen die in die component zitten.*

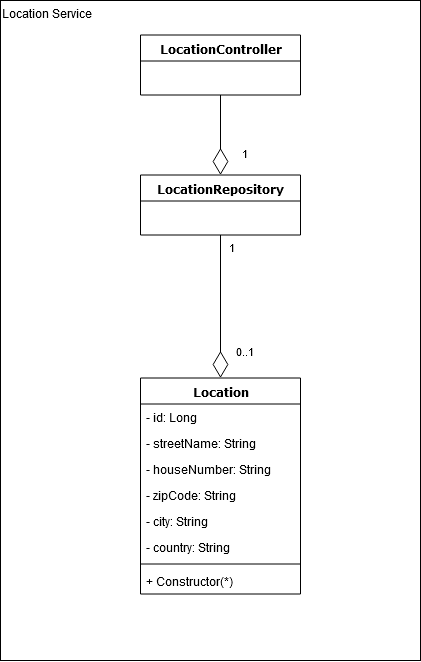
*Er zijn verschillende manieren om componentdiagrammen te tekenen, zelfs binnen UML. Interfaces kunnen binnen een component met elkaar verbonden worden en/of via een zogenaamde poort beschikbaar worden gesteld aan andere componenten. Een poort (engels: port) wordt weergegeven door middel van een vierkantje 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).*

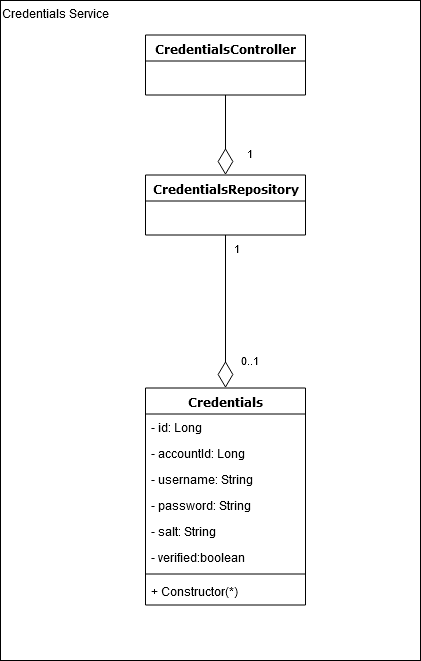
*Geef per component aan welke functionaliteit die bevat. Maak hierbij duidelijk in welke component(en) de business logica zich bevindt. Vaak heb je keuze om delen van business logica op de server of de client te beleggen. Verantwoord je keuzes (onderzoek !).*

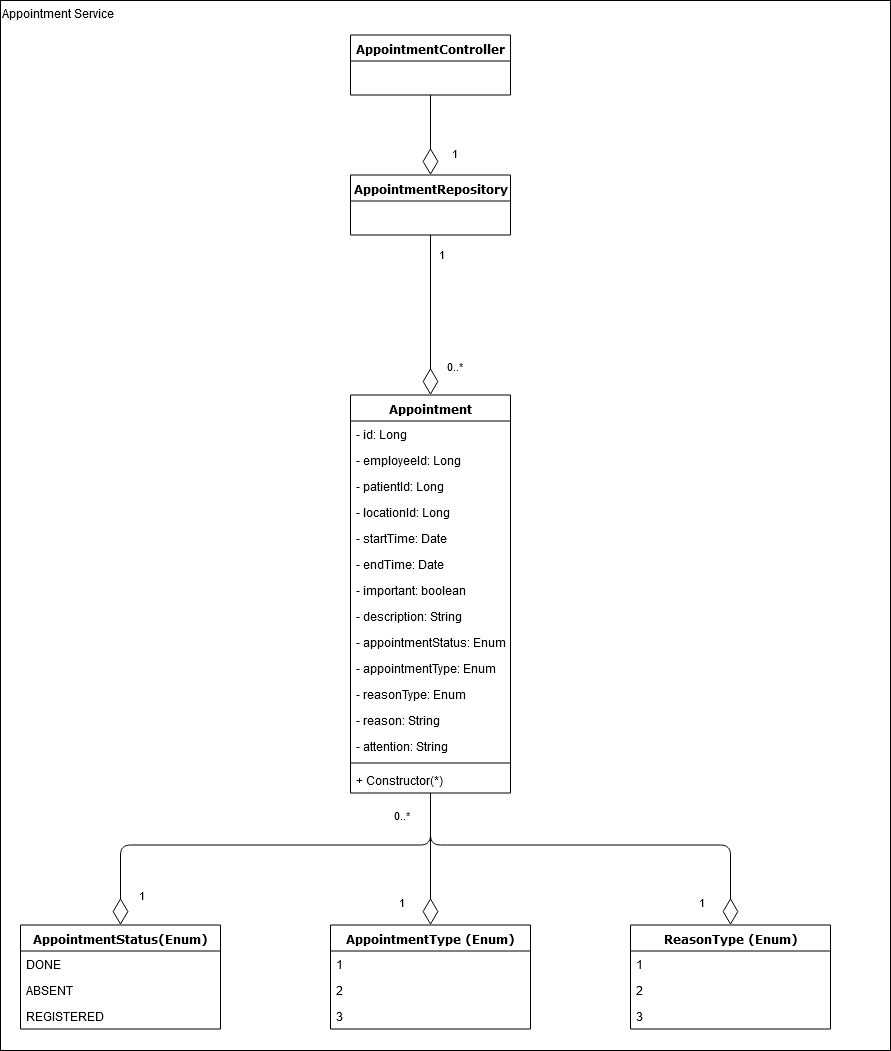
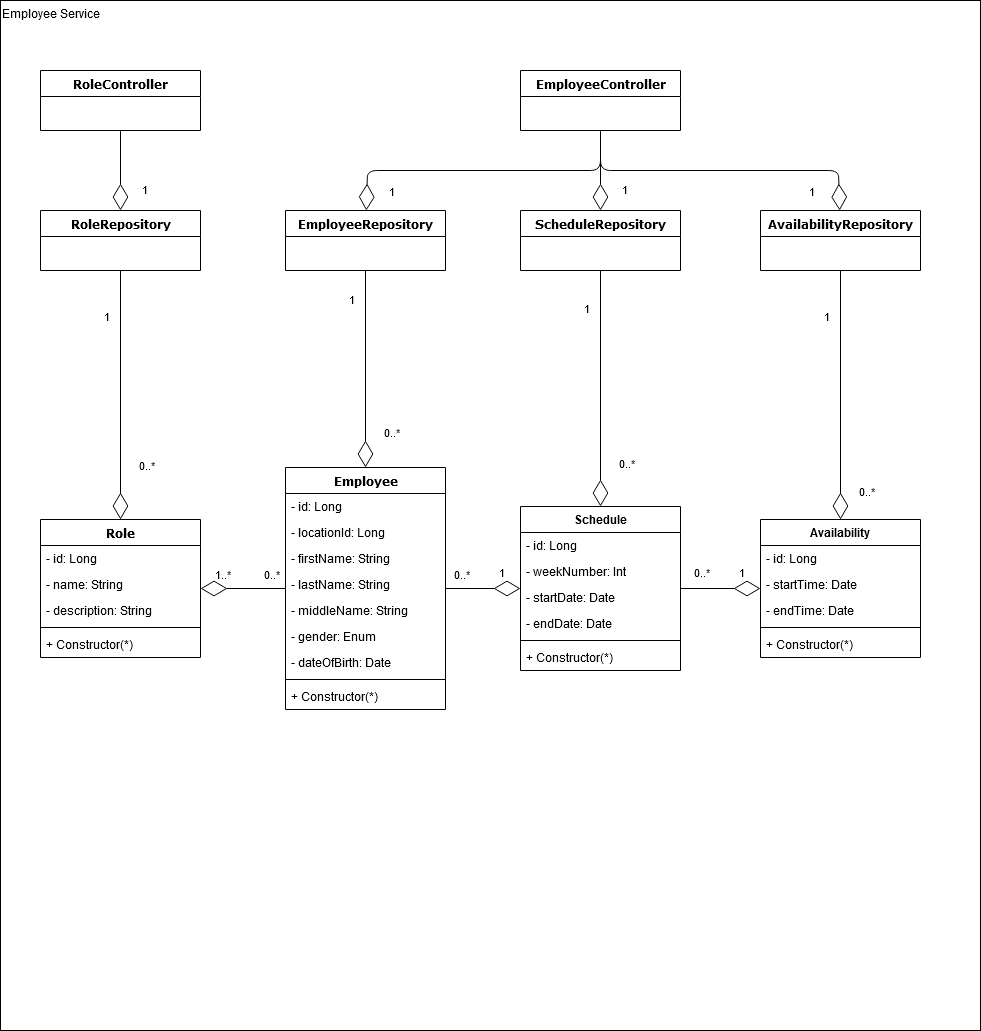
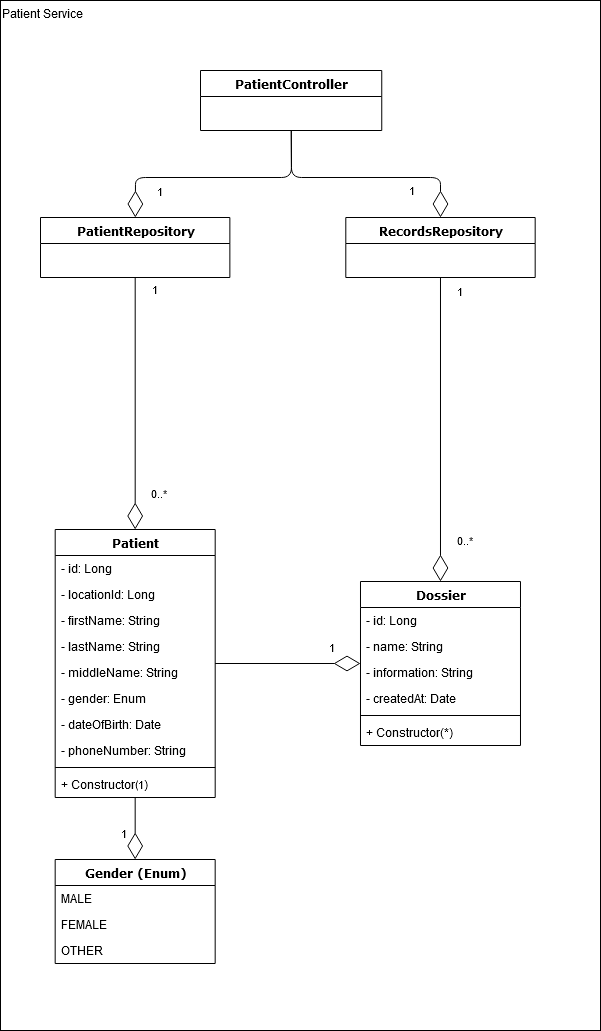


# Klassendiagrammen en Sequence diagrammen (C4)

*In dit hoofdstuk wordt het (detail)ontwerp van de componenten beschreven aan de hand van klassendiagrammen (applicatiemodel) en sequence diagrammen. Sequence diagrammen kunnen worden toegepast om het dynamisch gedrag binnen een component tussen instanties van klassen te beschrijven en ook om de communicatie tussen componenten te beschrijven. Werk minimaal 1 sequence diagram uit voor zowel communicatie tussen componenten en binnen een component.*





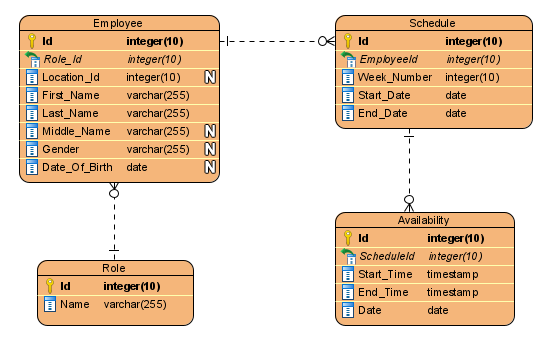
**

# Persistentie per component

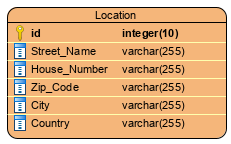
*In dit hoofdstuk wordt de persistentie van de objecten beschreven voor iedere component. Beschrijf welke eigenschappen van welke objecten moeten worden opgeslagen. Beschrijf hoe opslag wordt gerealiseerd (database, serialiseren, XML, etc.). Dit kan verschillen per eigenschap. Beschrijf ook wanneer opslag plaatsvindt. Ook dit kan verschillen per eigenschap.*

*Voeg het database model met beschrijving toe indien gegevens worden opgeslagen in een database. Hiervoor kun je een Entity Relationship Diagram (ERD) opstellen met behulp van Visual Paradigm (zie Database Modeling).*

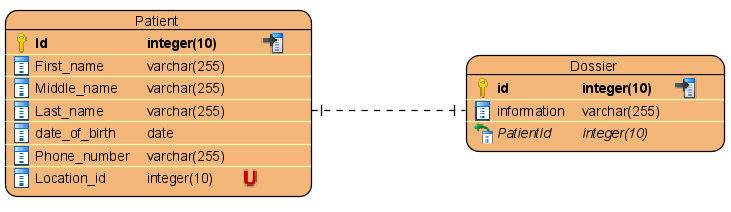
***Employee database***



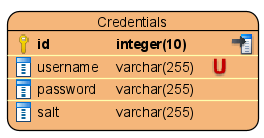
**Location Database**



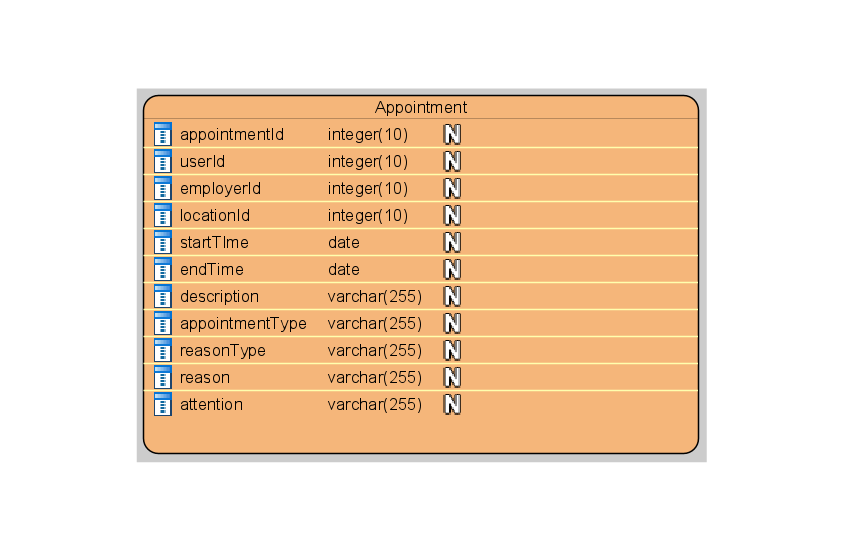
**Patient Database**



**Credentials Database**



**Appointment Database**



# Specificatie van interfaces

*In dit hoofdstuk wordt de specificatie van interfaces besproken. Voor RESTful API’s worden per URL http methods GET, PUT, POST en DELETE beschreven. Beschrijf voor iedere method het format dat wordt gebruikt om gegevens te verzenden. Denk hierbij aan parameters voor GET en DELETE requests, json voor PUT en POST requests en json en/of mogelijke excepties voor responses.*

*Voor Websockets wordt de communicatie beschreven aan de hand van een of meer sequence diagrammen. Laat zien hoe een client zich ‘aanmeldt’ bij de server en laat zien hoe ontvangen gegevens verwerkt worden en eventueel doorgestuurd naar andere clients. Geef ook voorbeelden in json om aan gegeven hoe gegevens worden gecommuniceerd.*

*Specificeer hier ook per websocket-bericht de structuur, bijvoorbeeld in JSON notatie of uit welke klassen elk bericht is opgebouwd.*

*Tip: gebruik tooling zoals Swagger (swagger.io) of RAML (raml.org) voor de beschrijving van RESTful APIs. Indien gewenst kun je met een URL verwijzen naar de beschrijving van de RESTful API.*

