Architectuur Document

Spots

Door: Paul Evers (3496783)

## Versies

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Versie** | **Datum** | **Auteurs** | **Wijzigingen** | **Versie** |
| 0.1 | 7-3-2021 | Paul Evers | Eerste opzet | Concept |
| 0.2 | 28-3-2021 | Paul Evers | C2 update, user load schatting begin | Concept |
| 0.3 | 8-4-2021 | Paul Evers | Load schatting uitgebreid | Concept |
| 0.4 | 18-4-2021 | Paul Evers | Pieken en per seconde meegeteld bij load. | Concept |

## 

## Introductie

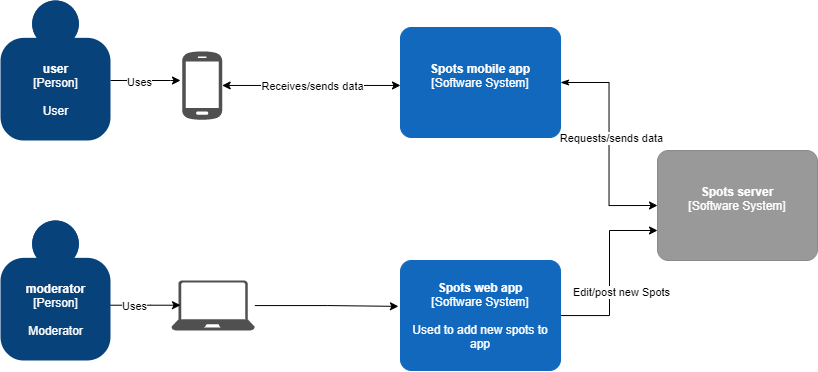
Het doel van dit document is om schematisch de software architectuur van de *Spots* applicatie weer te geven.

## C4 Model

Het C4 model geeft op verschillende niveaus de architectuur van *Spots* weer. Waar C1 het meest ‘high-level’ is.

### C1

*Spots* kent twee primaire gebruikers: de *user* en de *moderator*. De users zijn de consumenten die de app daadwerkelijk gebruiken op hun smartphone. De moderators zijn werknemers/vrijwilligers die via een webapp de verschillende *Spots* in de applicatie toevoegen, verwijderen, aanpassen, etc. Beiden hebben interactie met dezelfde back-end.



### 

### C2

*Spots* maakt gebruik van een microservice architectuur. Dit houdt in dat er verschillende losse containerized applicaties draaien die de backend van *Spots* vormen. Het voordeel hiervan is dat dit erg schaalbaar, flexibel, maintainable, en minder foutgevoelig is.

De clients gebruiken een API gateway om interactie met deze microservices te krijgen. De API gateway is een abstractielaag en fungeert als een single-point of entry voor de clients.

Achter iedere microservice zit ook een database waarin relevante data voor die service wordt opgeslagen.

De Eureka Discovery Server zorgt dat de API gateway makkelijk met alle services kan praten en fungeert daarnaast ook als een load balancer. De message queue wordt gebruikt om asynchrone taken uit te voeren. Hier kan de API gateway of andere services events op plaatsen. De andere services kunnen hier dan naar luisteren en iets mee doen indien nodig.

### 

## Geschatte load

Het is belangrijk om inzicht te hebben in hoeveel gebruikers je applicatie gaan gebruiken. Daarbij is het belangrijk om te weten hoeveel interactie ze met deze applicatie. Hiermee kan geschat worden hoeveel requests naar de server gemiddeld door de gebruikers gemaakt wordt. Deze kennis helpt bij het beslissen welke technieken het meest geschikt zijn voor de backend.

### Gemiddelde gebruiker

In de tabel beneden kun je een schatting zien van hoeveel calls een gebruiker per dag naar de server maakt, en naar welke services specifiek.

|  |  |  |  |
| --- | --- | --- | --- |
| Stap | Actie | Calls | Service |
| 1 | Inloggen | 1 | Firebase Auth service |
| 2 | Ophalen gebruikersgegevens | 1 | User Service |
| 3 | Ophalen Spots | 1 | Spot Service |
| 4 | Toevoegen aan favorieten | 5 | User Service |
| 5 | Audio streamen | 10 | Stream Service |
| 6 | Route berekenen | 3 | Route Service |
| 7 | Google Maps ophalen | 1 | Maps Service |

We kunnen de gegevens van hierboven gebruiken om te berekenen hoeveel calls er zijn als er meer users zijn. Ook wordt snel duidelijk welke services de meeste load zullen krijgen:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Aantal gebruikers: | Auth | User | Spot | Stream | Route | Maps | Gateway (Totaal) |
| 1 | 1 | 6 | 1 | 10 | 3 | 1 | 22 |
| 1000 | 1000 | 6000 | 1000 | 10,000 | 3000 | 1000 | 22,000 |
| 1,000,000 | 1,000,000 | 6,000,000 | 1,000,000 | 10,000,000 | 3,000,000 | 1,000,000 | 22,000,000 |

Het valt op dat vooral de *User* en *Stream* service de meeste calls zullen moeten behandelen. Het is dus belangrijk dat bij de deployment hier rekening mee wordt gehouden en meer resources aan wordt gegeven.

**Opsplitsing van totale load bij 100 miljoen gebruikers per dag inclusief pieken:**

|  |  |
| --- | --- |
| **Actie:** | **Totaal** |
| Aantal Gebruikers (Miljoen) | 100 |
| Aantal requests per sessie (gemiddeld) | 22 |
| Totaal requests per dag (Miljoen) | 2200 |
| Gemiddelde requests per uur (Miljoen) | 91.7 |
| Gemiddelde requests per minuut | 1.53 |
| Gemiddelde requests per seconde | 0.0255 |
| Piek requests per uur (Miljoen) X 3,14 | 287.9 |
| Piek requests per minuut  X 3,14 | 4.8 |
| Piek requests per seconde X 3,14 | 0.0801 |

**Verdeling de load aantal requests per service per seconde**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Gateway** | **Auth** | **User** | **Spot** | **Stream** | **Route** | **Maps** |
| **Totaal** | 25500 | 1159 | 6955 | 1159 | 11591 | 3477 | 1159 |
| **Totaal piek (X3,14)** | 80070 | 3639 | 21837 | 3639 | 36395 | 10919 | 3639 |

**Maximale load technieken:**

*Firebase auth:* Gratis, unlimited aantal gebruikers, 500 requests per seconde

*MongoDB:* Oneindig, schaalt met beschikbaar geheugen

*Spring boot:* Oneindig, schaalt met beschikbaar geheugen