Datagedreven webapplicatie getest voor high performance en security door Kay Verhaegh, 16-12-2019, Course NotS-WAPP

Kay Verhaegh

Ontwerpdocument Inside AirBnB Amsterdam

2019

Inhoudsopgave

[1. Inleiding 2](#_Toc10460856)

[2. Functioneel ontwerp 2](#_Toc10460857)

[3. Technisch ontwerp 2](#_Toc10460858)

[4. Performance 2](#_Toc10460859)

[4.1. AsNoTracking 3](#_Toc10460860)

[4.2. Async 5](#_Toc10460861)

[4.3. Caching 6](#_Toc10460862)

[4.4. Conclusie 7](#_Toc10460863)

[5. Security 7](#_Toc10460864)

[5.1. OWASP ZAP 7](#_Toc10460865)

[X-Frame-Options Header Not Set 7](#_Toc10460866)

[Cookie No HttpOnly Flag 7](#_Toc10460867)

[Cookie Without secure flag 8](#_Toc10460868)

[Incomplete or No Cache-control and Pragma http Header Set 8](#_Toc10460869)

[Web Browser XSS Protection Not Enabled 9](#_Toc10460870)

[X-Content-Type-Options Header Missing 9](#_Toc10460871)

[5.2. Conclusie 9](#_Toc10460872)

[Bijlages 10](#_Toc10460873)

[Bijlage “Caching load test” 10](#_Toc10460874)

[Bijlage: Literatuurlijst 11](#_Toc10460875)

# Inleiding

Het managementteam van InsideAirbnb hebben op het moment weinig tot geen inzicht in het gebruik van Airbnb locaties binnen Amsterdam. Om meer inzicht te krijgen willen zij een applicatie waarin zij kunnen bekijken wat het gemiddelde aantal overnachtingen is in een maand, wat de opbrengsten zijn per buurt, en wat de gemiddelde reviews zijn in een buurt. Dit willen zij kunnen zien doormiddel van stuurinformatie op de applicatie, er moet tevens voor gezorgd worden dat de applicatie dezelfde “look-and-feel” krijgt voor medewerkers en externe gebruikers. Het front-end van de applicatie moet gebaseerd zijn op de website van insideairbnb.com.

Tijdens het ontwikkelproject moet er stuurinformatie over de applicatie bijgehouden worden om zo adviezen te kunnen tonen.

# Functioneel ontwerp

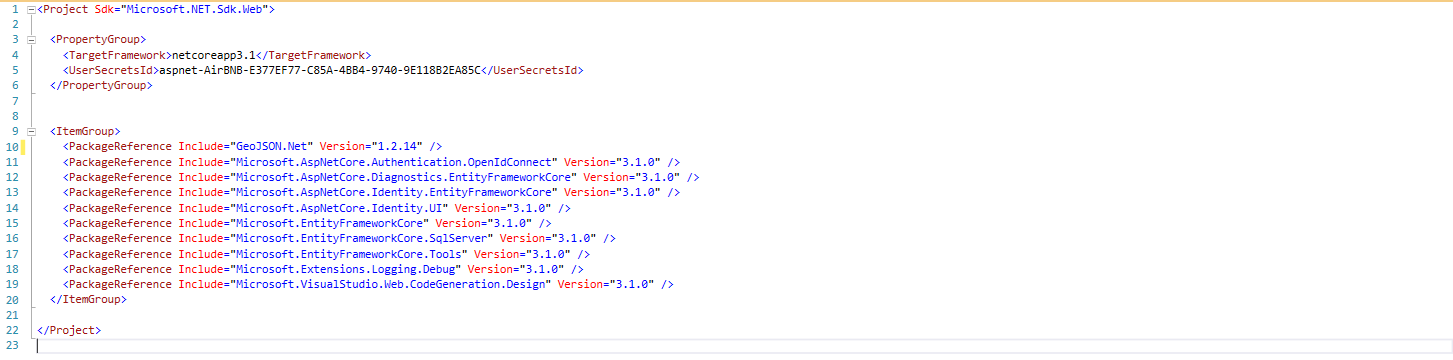
Hieronder staan de usecases van het AirBNB project beschreven met daarbij de prioriteit van deze usecases beschreven in een MOSCOW-tabel.

|  |  |
| --- | --- |
| Usecase | Prioriteit |
| Registeren en inloggen | Must |
| Filter op prijs | Must |
| Filter op buurt | Must |
| Filter op review | Must |
| Locaties van zoekresultaat zichtbaar op kaart | Could |
| Kaart is clickable, details rechts op pagina, maakt gebruik van de mapbox API | Must |
| Layout idem als insideairbnb.com | Could |
| Details per item waarop gefilterd is: #overnachtingen, #opbrengst in de maand | Must |
| Er moeten rollen toegevoegd en toegekend worden aan geregistreerde gebruikers | Must |
| Resultaten (trends, totalen, gemiddelden, etc.) worden weergegeven in charts, alleen te bekijken voor ADMINS | Must |

# Technisch ontwerp

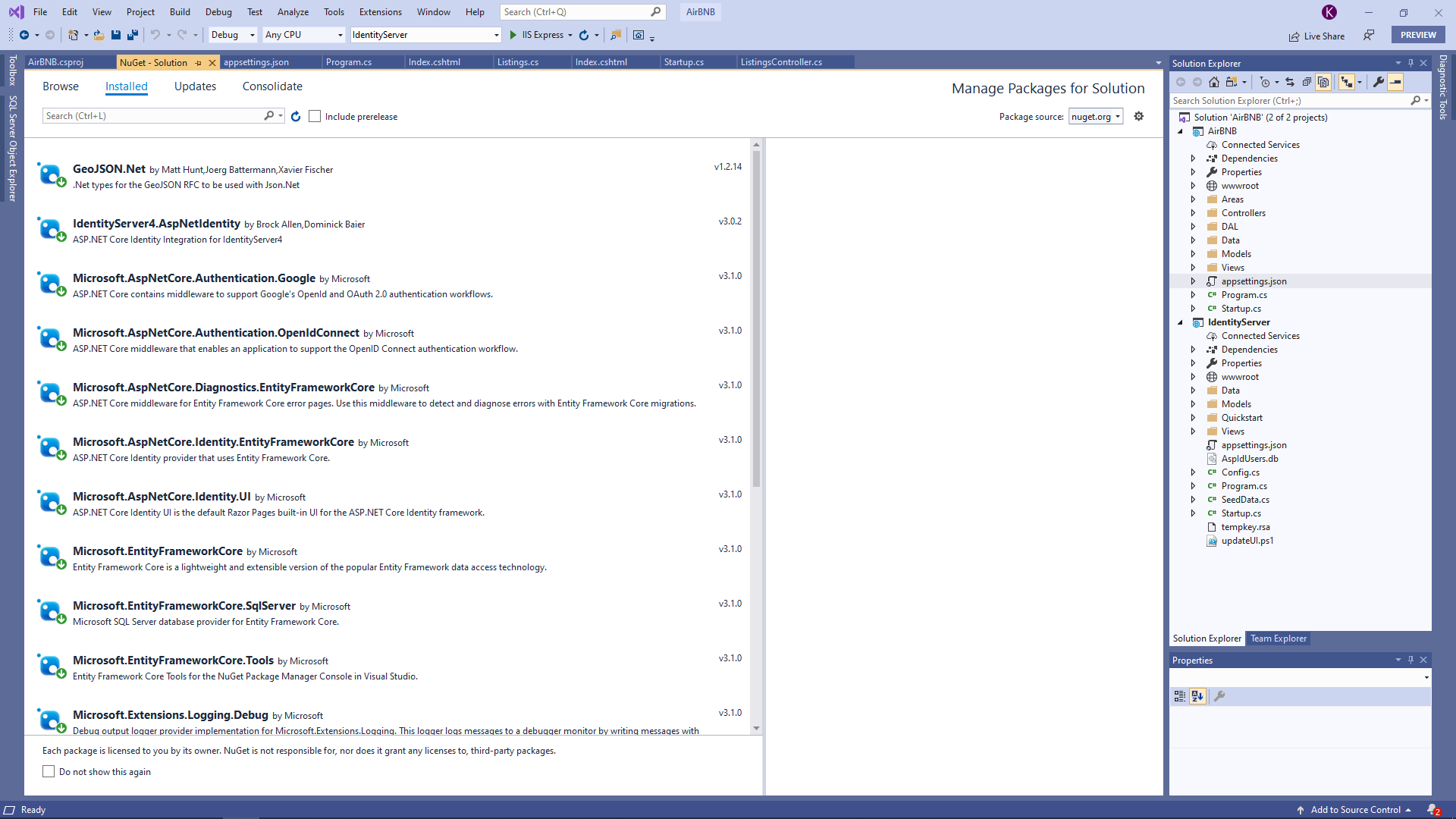
## Architectuur

## Overzicht gebruikte frameworks

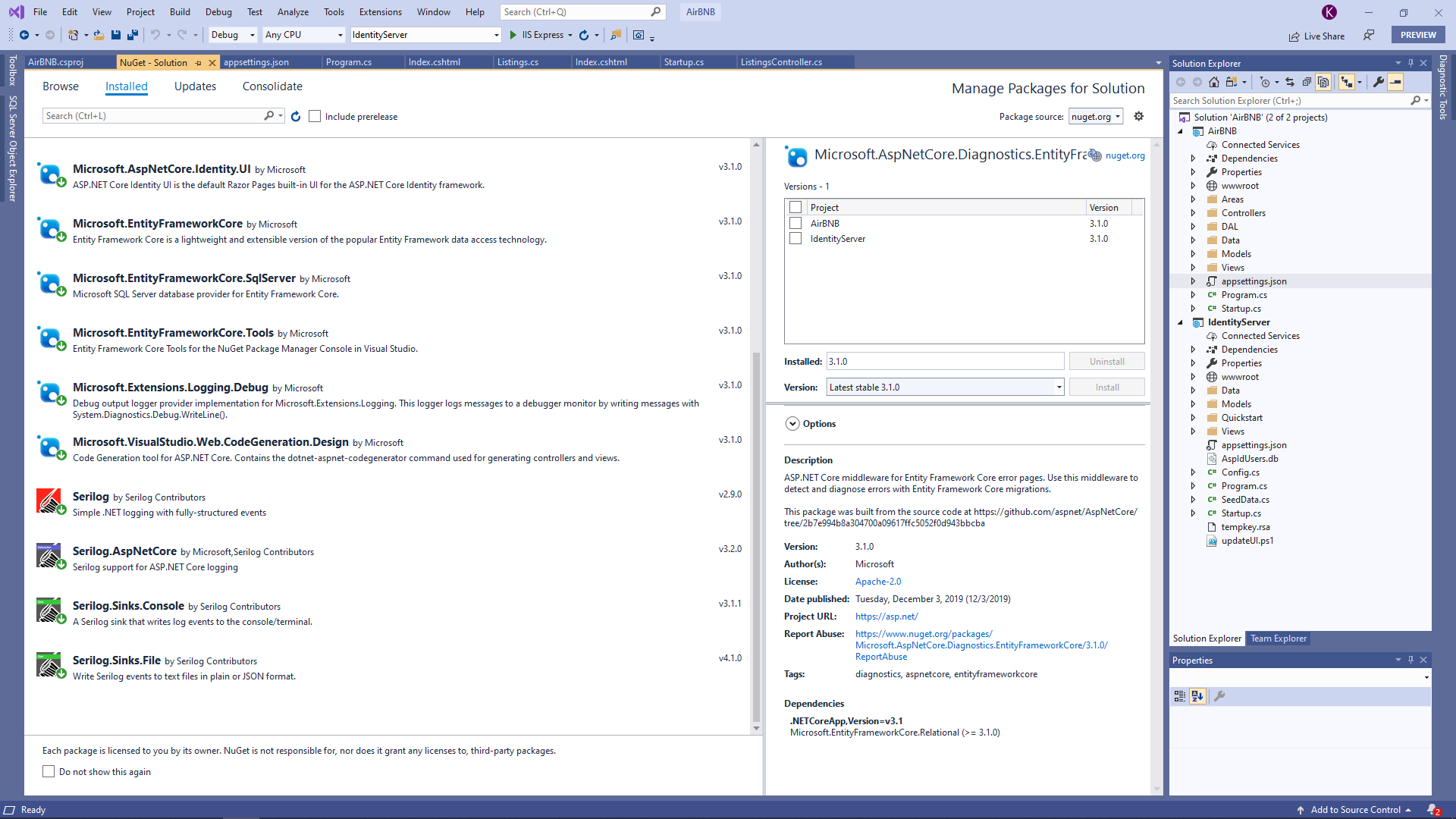


Figuur 1

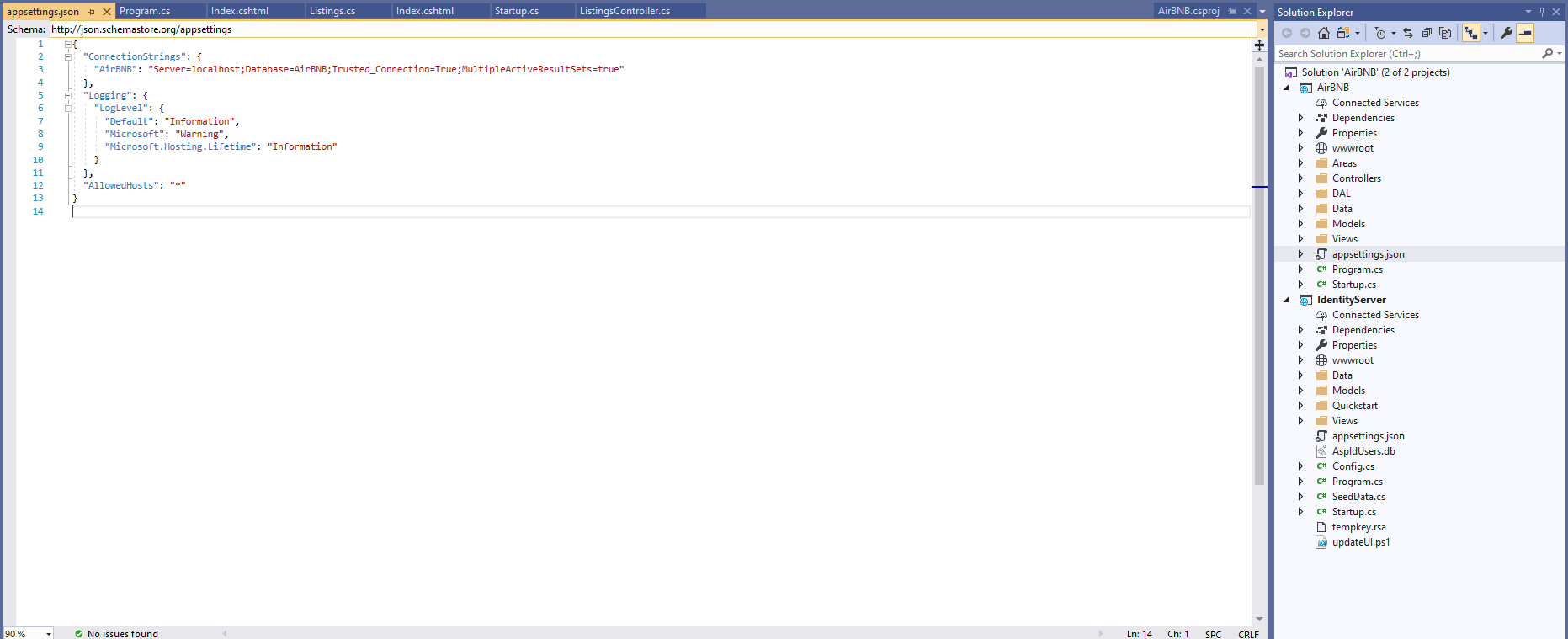
## Overzicht gebruikte packages incl. versies



Figuur 2



Figuur 3



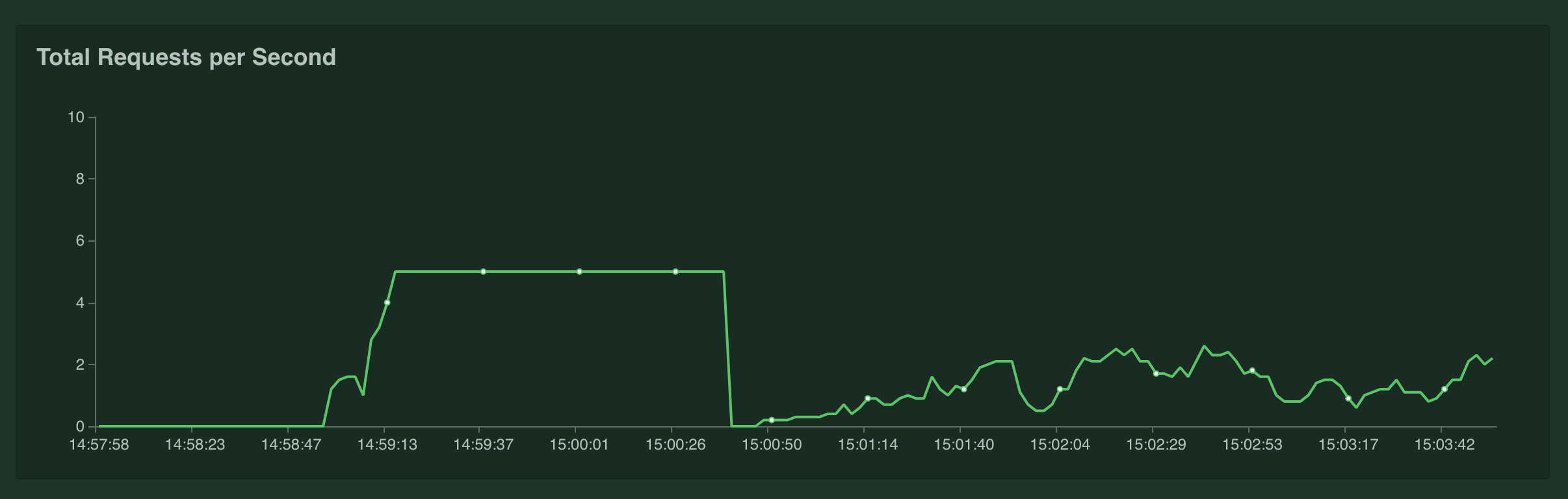
Figuur 4

# Performance

In dit hoofdstuk ga ik in op de verschillende performance optimalisaties die zijn uitgevoerd op de Inside AirbnB webapplicatie. Als eerste ga ik in op de prestatieverbeteringen die zijn toegebracht met AsNoTracking, Async (Bron) en als laatste de verbeteringen met memory caching (Bron).

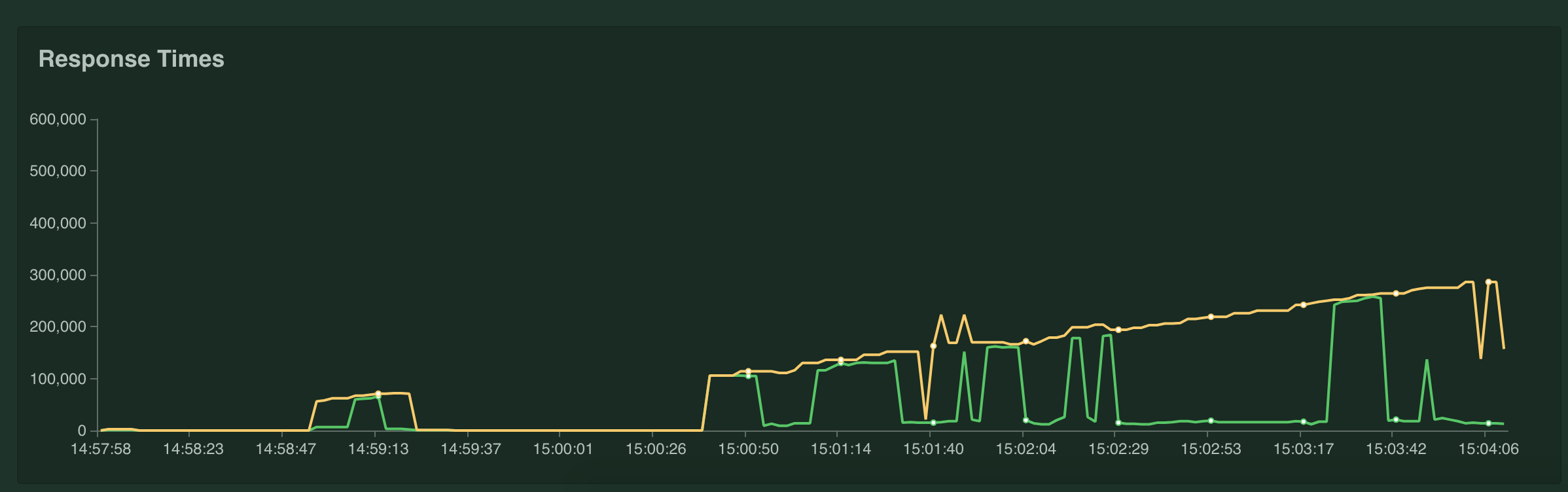
Om de verbeteringen in de prestaties te zien is het nodig om een baseline te hebben. Hiervoor heb ik *Locust (Bron)* gebruikt om honderd gebruikers te simuleren, met een spawn rate van 10 per seconde. De resultaten van Locust zijn als volgt.

Aantal requests per seconde. Zie figuur 1.



Figuur 2

Response Times:



Figuur 3

De aangesproken endpoints voor het bepalen van de prestatie verbeteringen zijn:

* GET /api/Chart/trend/neighbourhood
* GET /api/listings
* GET /api/listings/{id}

## 4.1. AsNoTracking

Entity Framework biedt een aantal opties voor het verbeteren van de prestaties. Een van deze opties is AsNoTracking(). Met deze optimalisatie kan er aangeven worden dat Entity Framework de resultaten van een query niet moet bijhouden. Dit betekent dat Entity Framework geen aanvullende verwerking of opslag uitvoert van de entiteiten die door de query worden geretourneerd. Wat ervoor zorgt dat er minimaal verbruik gemaakt wordt van memory. (Bron)

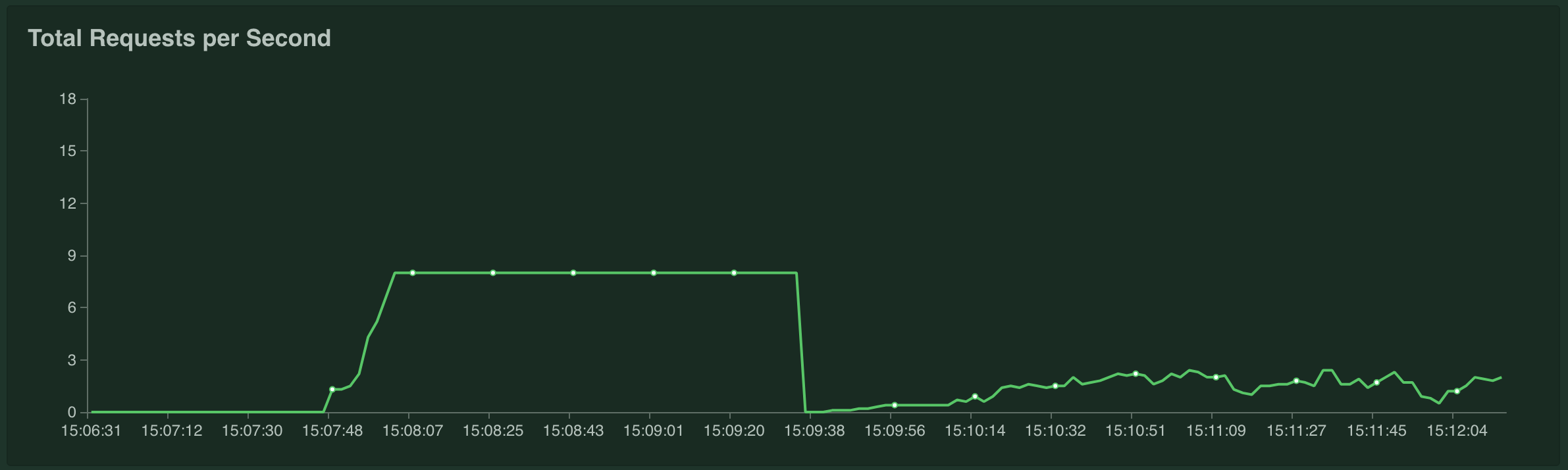
AsNoTracking () moet alleen maar gebruikt worden wanneer de intentie is om data op te halen en wijzigingen niet bij te houden.

**Code voorbeeld**

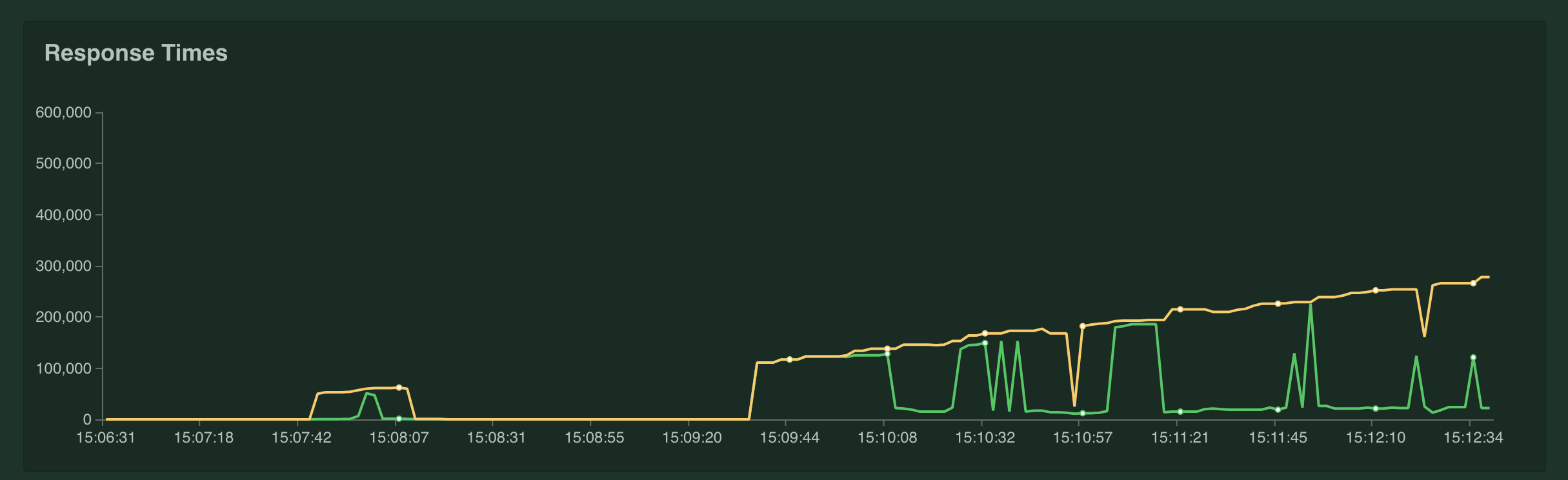


**Resultaten**

Requests per seconde



Response Times:



## 4.2. Async

Twee van de drie endpoints die aangeroepen worden zijn endpoints met een hoge response time. Aangezien nu de applicatie blijft wachten op deze synchrone endpoints heb ik ze omgezet naar asynchrone endpoints.

**Code voorbeeld**

****

**Resultaten**

Aantal Requests per seconde. Zie figuur 3.



Figuur 4

Response times. Zie figuur 4.



## 4.3. Caching

Caching in de AirBnB applicatie kan op meerdere manieren. De oplossingen die geïmplementeerd zijn, zijn Distributed, in-memory en response caching.

Van de meerdere oplossingen die er bestaan van distributed caching heb ik Redis gebruikt. Redis is een cachingtechnologie die webapplicaties snel data laat bewaren en terug oproepen uit geheugen. Redis is net als Memcached een distributed cache oplossing. Zie bijlage “Caching loadtest”

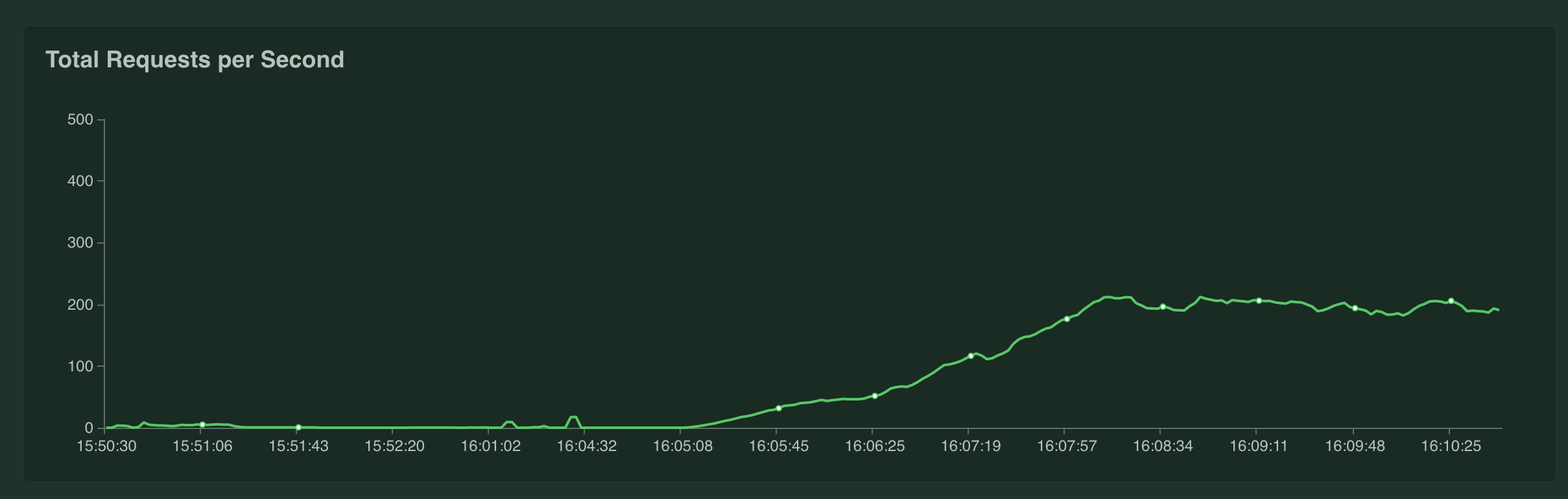
Om te testen welke ik compleet wel implementeren heb ik drie verschillende load tests gedaan. Ten eerste heb ik de database getest, daarna met in memory data en al laatste met redis.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Database | In-Memory | Redis |
| Aantal requests per seconde | 4 | 9 | 0.4 |
| Response Times (95%) | 2200 ms | 350 ms | 29000 |

Zoals in de bovenste resultaten te zien is, is het duidelijk dat In-Memory het beste presteert omdat de cache lokaal bestaat in .NET in plaats van in de cloud (redis).

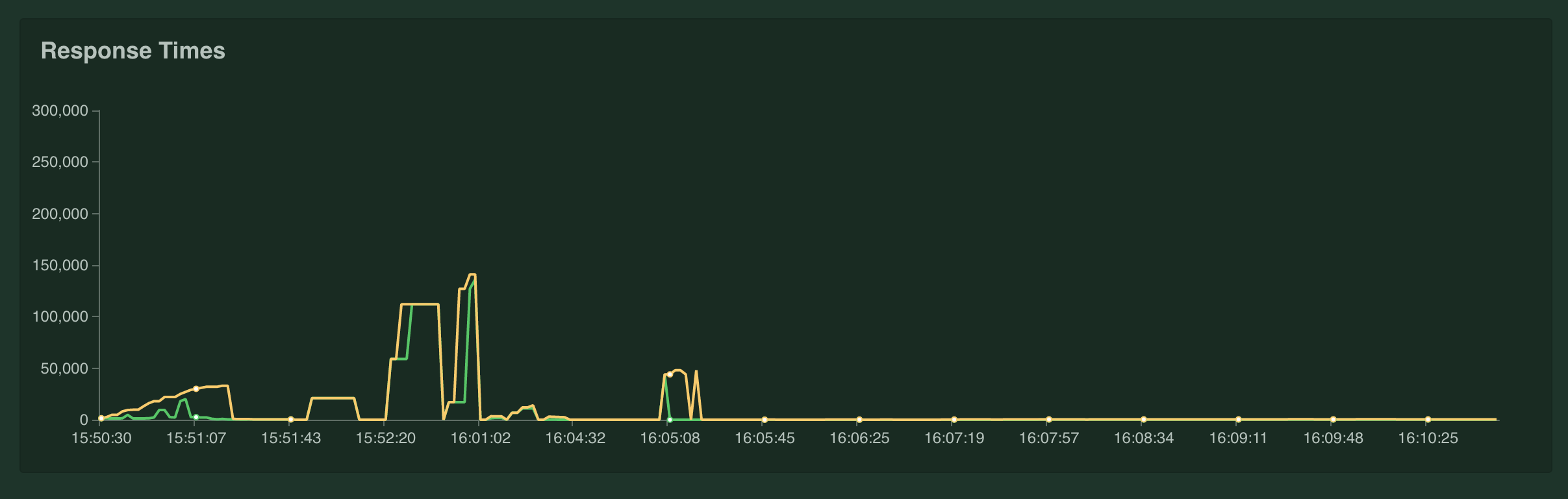
**Resultaten**

Aantal requests per seconde. Zie figuur 4

****

Figuur 5

Response time. Zie figuur 5

****

Figuur 6

## 4.4. Conclusie

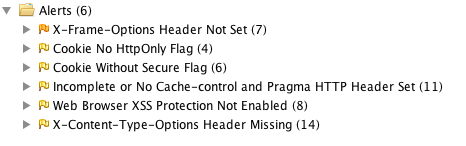
Als de hoogste aantal requests per seconde gepakt worden bestaat er een verbetering van 2122.22%. De grootste verbetering ontstaat door gebruik te maken van In-Memory caching, wat de latency omdat de applicatie niks van de database hoeft op te halen.

# 5. Security

In dit hoofdstuk wordt uitleg gegeven op de security issues die zijn gevonden met OWASP ZAP.

## 5.1. OWASP ZAP

De onderstaand Alerts zijn de issues die gevonden zijn binnen mijn webapplicatie. Deze issues worden stuk voor stuk verwerkt in dit hoofdstuk. Zie figuur 6.

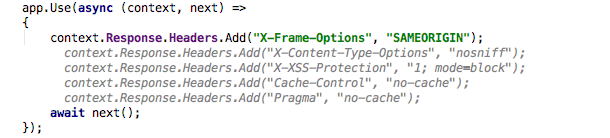


Figuur 7

### X-Frame-Options Header Not Set

X-Frame-Options header zorgt ervoor dat de browser het niet of wel toelaat om de content in een X-Frame te laten zien.

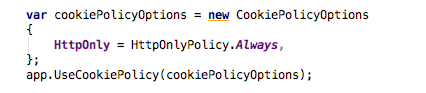
In dit geval is een X-Frame niet van toepassing en kan op de volgende manier opgelost worden. Zie figuur 7.



Figuur 8

### Cookie No HttpOnly Flag

Cookies mogen alleen maar gebruikt bij het http protocol. Dat betekent dat het niet mogelijk is om document.cookie te gebruiken vanuit een javascript bestand.

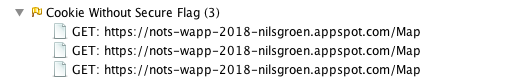


Figuur 9

### Cookie Without secure flag

Cookies met een secure kenmerk zijn een soort cookies die worden verzonden via gecodeerde http-verbindingen. Bij het instellen van de cookie geeft het secure kenmerk de browser de opdracht om de cookie alleen via geëncrypte verbindingen gebruikt moet worden.

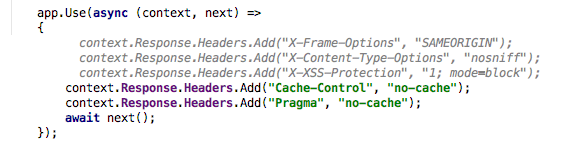
Aangezien het alleen maar te vinden is bij de Map pagina heb ik de aanname genomen dat mapbox de boosdoener is. Zie figuur 9.



Figuur 10

### Incomplete or No Cache-control and Pragma http Header Set

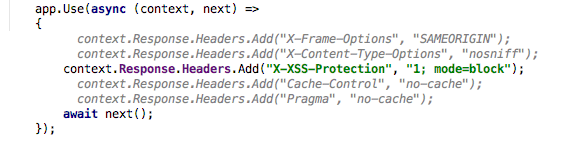
De “Incomplete or No Cache-control and Pragma” header zorgt ervoor dat de browser altijd checked bij de server of de gecachde response nog geldig is. Hierdoor is het onmogelijk dat een response te lang wordt gecached en wordt weergegeven wanneer dit niet de bedoeling is. Deze headers worden overschreven wanneer er in de code headers worden toegevoegd aan bepaalde endpoints, bijvoorbeeld wanneer ik in de code aangeef dat een response maximaal 120 seconde gecached mag worden. Zie figuur 10.



Figuur 11

### Web Browser XSS Protection Not Enabled

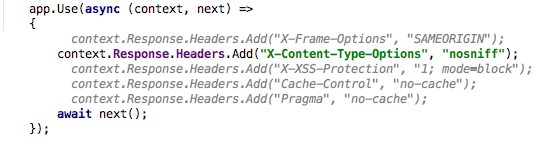
Deze header zorgt ervoor da een browser geen requests meer stuurt wanneer er XSS wordt herkend. Dit heb ik opgelost met de onderstaande code. Zie figuur 11.



Figuur 12

### X-Content-Type-Options Header Missing

Deze header zorgt ervoor dat de browser niet gaat proberen te raden wat de MIME type is van een bestand dat wordt opgehaald vanuit de server. Dit heb ik opgelost met de onderstaande code. Zie figuur 12.



Figuur 13

## 5.2. Conclusie

Zoals te zien in het bovenstaand voorbeeld zijn er door ZAP zes alerts gevonden. Hiervan heb ik vier kunnen oplossen. De rest van de alert komen de mapbox api. Daarnaast worden veel van de security issues al opgelost alleen door gebruik te maken van .Net core, identity en entity framework. Zie figuur 13 voor de overgebleven errors.

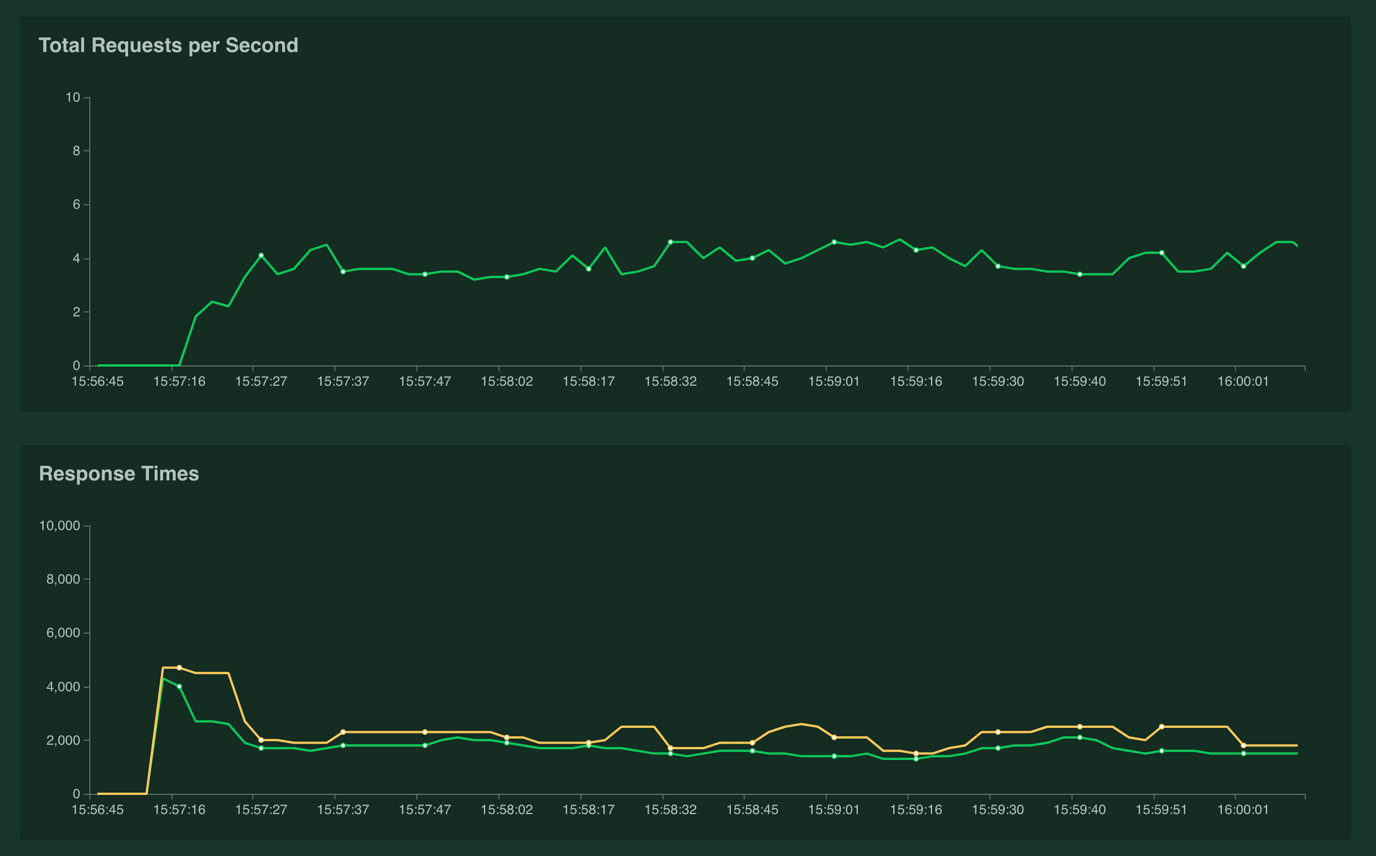


Figuur 14

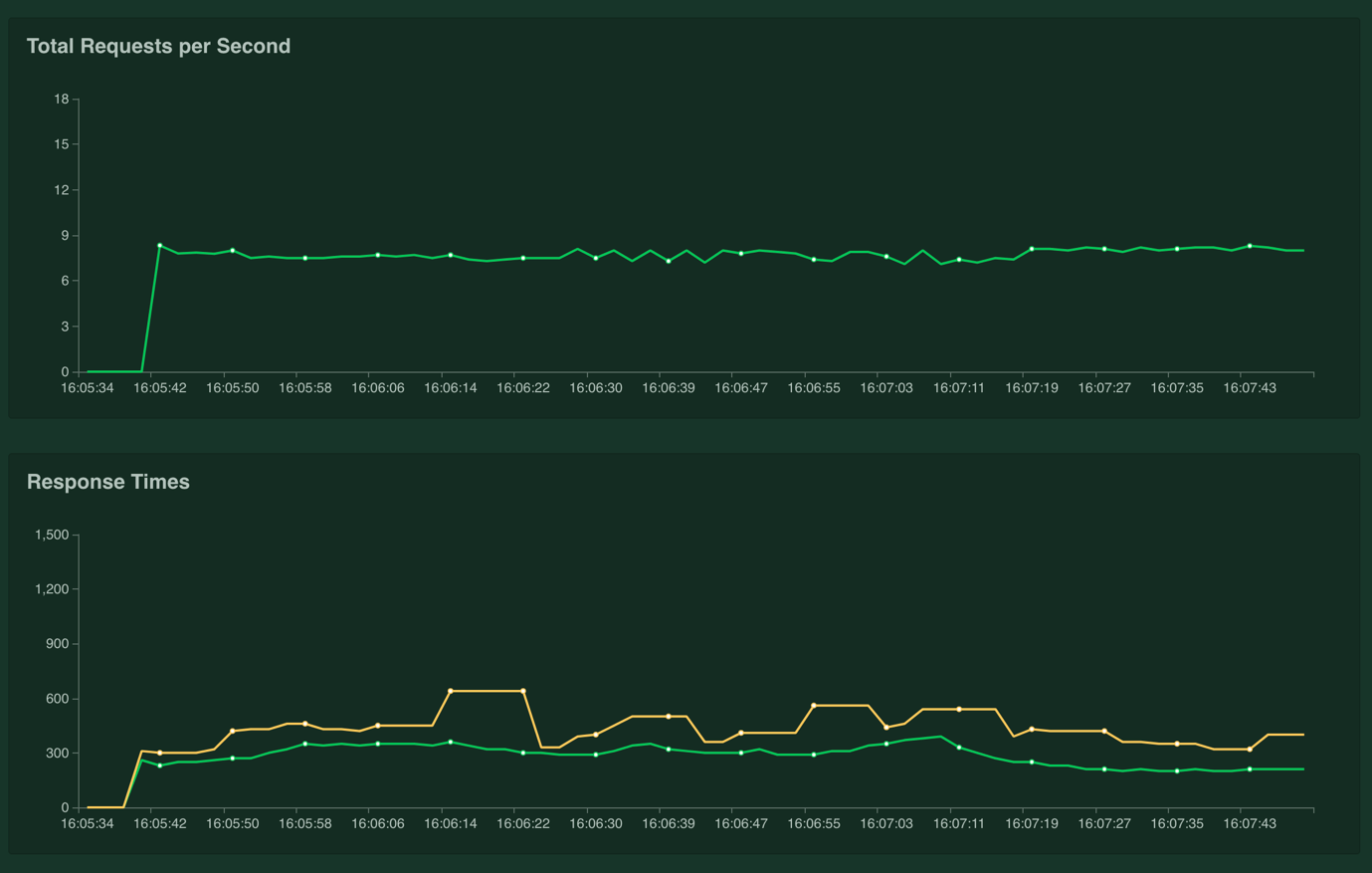
# Bijlages

## Bijlage “Caching load test”

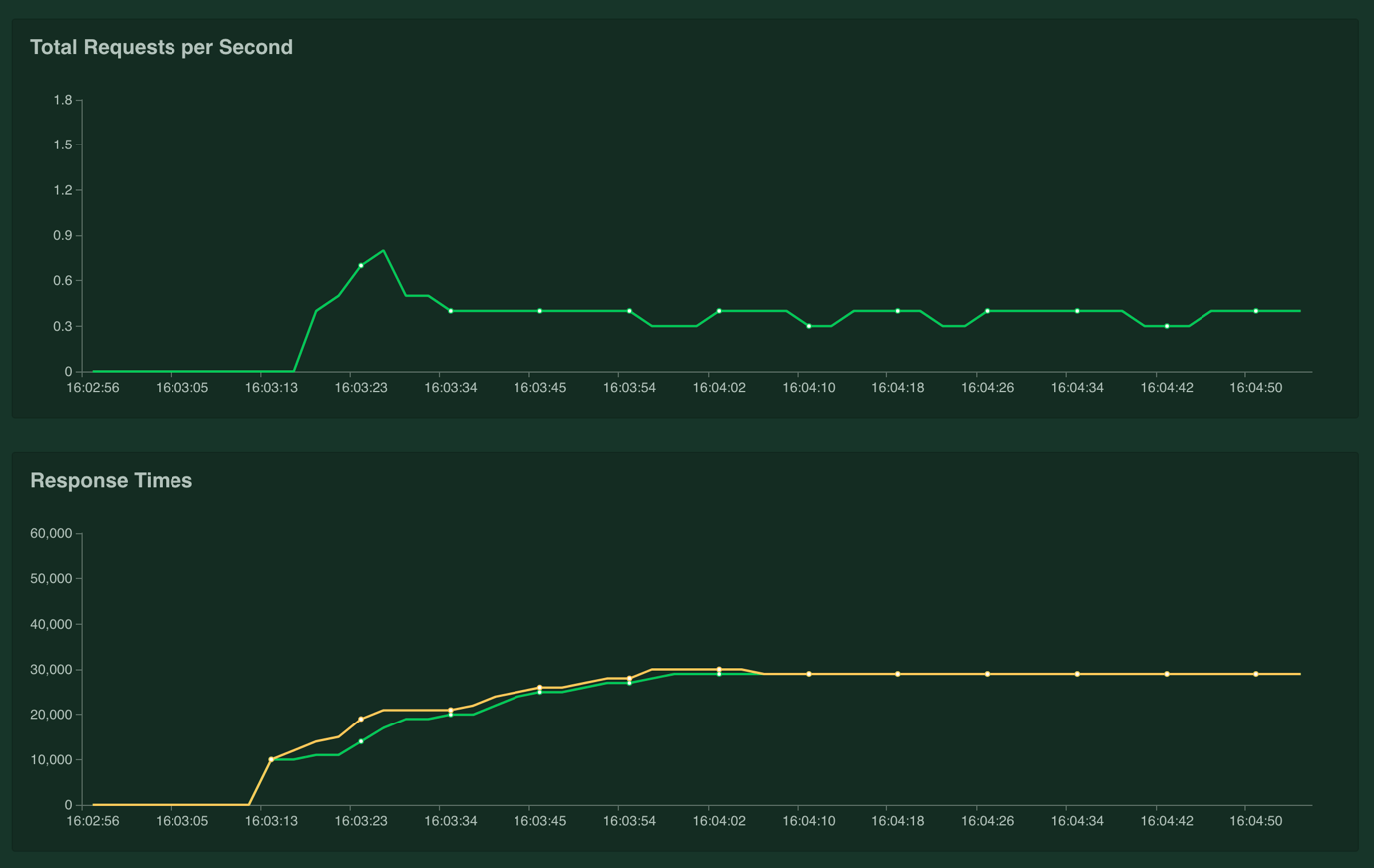
**Database**



**In-Memory**



**Redis**



## Bijlage: Literatuurlijst