

PROJECTDOCUMENTATIE INSIDE AIRBNB

Naam: Sjaak Kok

Studentnummer: 620581

Klas: ITA-NotS-A-f

Docent: Marcel Verheij

Datum: 09-06-2022

Course: WAPP

INHOUDSOPGAVE

Inleiding	2
Functioneel Ontwerp	3
Use Cases.....	3
Requirements	3
Technisch Ontwerp	4
Architectuur.....	4
Request pipeline.....	4
Frameworks & Packages	5
Performance.....	6
Nulmeting.....	7
AsNoTracking.....	9
Response Compression	11
Indexing	13
Caching	14
Security.....	17
Content Security Policy (CSP) Header Not Set	17
Missing Anti-clickjacking Header.....	18
Application Error Disclosure.....	18
Server Leaks Information	18
X-Content-Type-Options Header Missing	19
Openstaande waarschuwingen.....	19
Verwijzingen	20

INLEIDING

Dit document bevat de documentatie behorend bij mijn eigen Inside Airbnb applicatie en moet inzicht geven in de totstandkoming van de applicatie met bijbehorende keuzes. De applicatie moet inzicht geven in het gebruik van Airbnb locaties in Amsterdam. De data die voor deze applicatie is gebruikt komt van Inside Airbnb (Inside Airbnb, sd).

FUNCTIONEEL ONTWERP

USE CASES

In de tabel hieronder staat welke functionaliteiten, oftewel use cases, er in de applicatie moeten zitten.

Use case	Prioriteit
Registreren en inloggen	Must
Filteren op prijs	Must
Filteren op buurt	Must
Kaart is clickable, details rechts op pagina, maakt gebruik van de mapbox API	Must
Details per item zien waarop is gefilterd	Must
Er moeten rollen toegevoegd en toegekend worden aan geregistreerde gebruikers	Must
Resultaten zoals trends, totalen, gemiddelden, etc. worden weergegeven in charts en zijn alleen te bekijken voor admins	Must
Locaties van zoekresultaat zichtbaar op kaart	Could
Layout idem als insideairbnb.com	Could

REQUIREMENTS

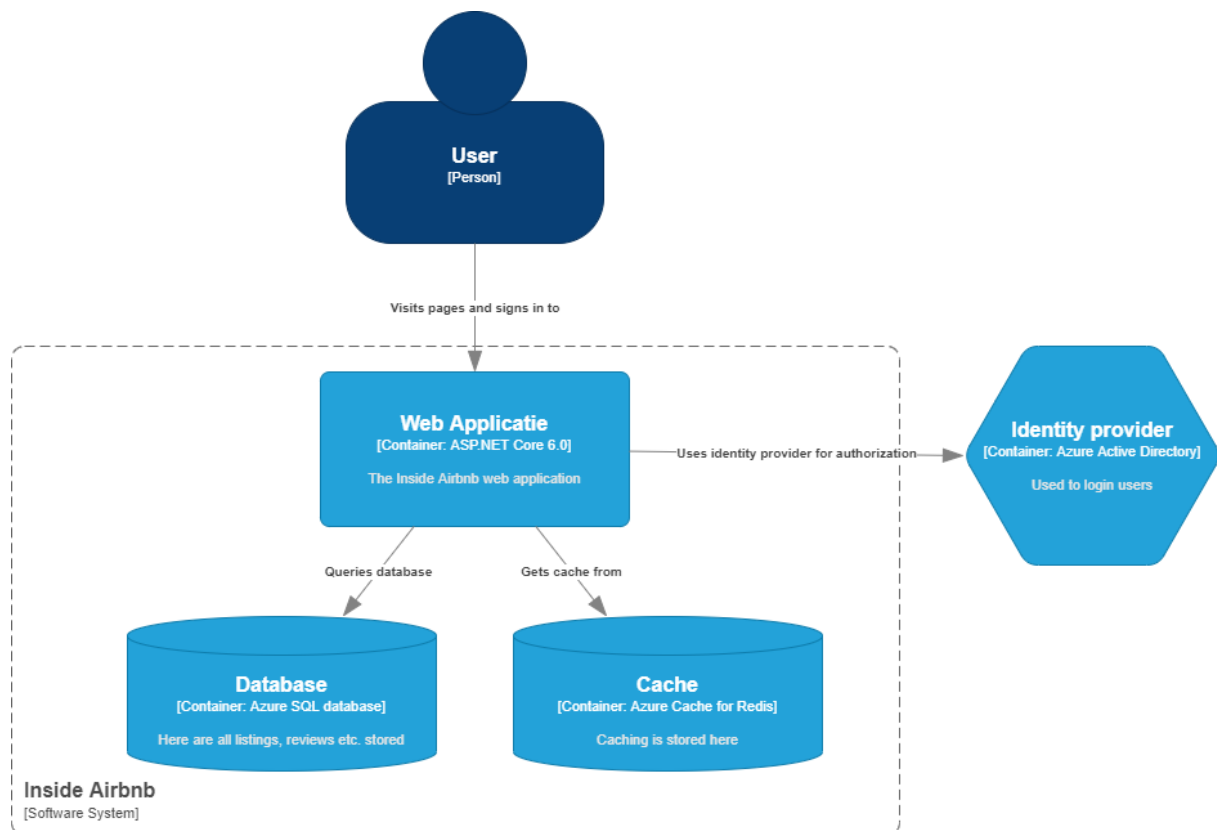
In de tabel hieronder staat aan welke requirements de applicatie moet voldoen.

Requirement	Prioriteit
Ontwikkelt met de laatste Microsoft ASP.Net Core versie	Must
Wordt gehost op het Azure Cloud Platform	Must
Maakt gebruik van ASP.Net Razor pages of MVC	Must
Maakt gebruik van MSSQL Server (versie van Azure)	Must
De applicatie moet veilig zijn	Must
De applicatie is aantoonbaar highly-scalable. Er worden daarvoor performance tests als bewijsmateriaal opgeleverd	Must
Authenticatie en autorisatie via Azure of IdentityServer4	Must
Caching service via Redis	Must
Microservice architectuur	Would
Microservice orchestration met Kubernetes	Would
Blazor applicatie met SignalR	Would

TECHNISCH ONTWERP

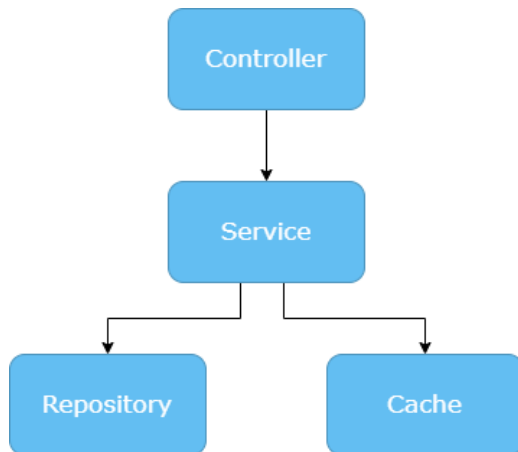
ARCHITECTUUR

Hieronder staat de architectuur van de applicatie afgebeeld. De webapplicatie is een MVC applicatie en is gebouwd met behulp van het ASP.NET Core 6.0 framework van Microsoft. Er is gebruik gemaakt van enkele Azure producten. Voor identity wordt er gebruikt gemaakt van Azure Active Directory. Daarnaast is de applicatie is verbonden met een Azure SQL database en een caching service genaamd Azure Cache for Redis.



REQUEST PIPELINE

In de afbeelding hieronder is de request pipeline weergegeven. Een request komt binnen in de controller. Om informatie op te halen wordt er vanuit de controller een call gedaan naar de betreffende service. Er zijn drie services in de applicatie om informatie op te halen; listingservice, neighbourhoodservice en reviewservice. In de service wordt eerst gekeken of de benodigde informatie al in de cache zit. Als dit het geval is wordt deze informatie teruggestuurd naar de controller. Als de benodigde informatie nog niet in de cache zit wordt er een call gedaan naar de betreffende repository om de informatie uit de database op te halen. Deze opgehaalde informatie wordt vervolgens in de cache gedaan en teruggestuurd naar de controller.



FRAMEWORKS & PACKAGES

De onderstaande afbeelding laat alle packages met versienummers zien die in de app worden gebruikt.

```
<PackageReference Include="Microsoft.AspNetCore.Authentication.JwtBearer" Version="6.0.5" NoWarn="NU1605" />
<PackageReference Include="Microsoft.AspNetCore.Authentication.OpenIdConnect" Version="6.0.5" NoWarn="NU1605" />
<PackageReference Include="Microsoft.EntityFrameworkCore.Design" Version="6.0.5">
  <PrivateAssets>all</PrivateAssets>
  <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
</PackageReference>
<PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="6.0.5" />
<PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="6.0.5">
  <PrivateAssets>all</PrivateAssets>
  <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
</PackageReference>
<PackageReference Include="Microsoft.Extensions.Caching.StackExchangeRedis" Version="6.0.5" />
<PackageReference Include="Microsoft.Identity.Web" Version="1.24.1" />
<PackageReference Include="Microsoft.Identity.Web.UI" Version="1.24.1" />
<PackageReference Include="Microsoft.VisualStudio.Azure.Containers.Tools.Targets" Version="1.15.1" />
<PackageReference Include="Microsoft.VisualStudio.Web.CodeGeneration.Design" Version="6.0.5" />
<PackageReference Include="MiniProfiler.AspNetCore.Mvc" Version="4.2.22" />
<PackageReference Include="MiniProfiler.EntityFrameworkCore" Version="4.2.22" />
<PackageReference Include="NetEscapades.AspNetCore.SecurityHeaders" Version="0.16.1" />
<PackageReference Include="NetEscapades.AspNetCore.SecurityHeaders.TagHelpers" Version="0.16.1" />
<PackageReference Include="Swashbuckle.AspNetCore" Version="6.3.1" />
```

PERFORMANCE

In dit hoofdstuk staan verschillende performance verbeteringen die ik heb gedaan voor de applicatie. Om performance verbeteringen ook daadwerkelijk in beeld te krijgen is er eerst een nulmeting gedaan met JMeter waarin de performance wordt vastgesteld zonder verbeteringen. Daarna zijn er verschillende verbeteringen geïmplementeerd.

De metingen worden gedaan op de URLs: <https://localhost:44313/Listings/> en <https://localhost:44313/Home/Statistics/>. De eerste is een pagina waarop de eerste 200 listings worden weergegeven. En de tweede is een pagina waarop de statistieken voor admins staan.

Om de metingen zo accuraat mogelijk te doen zijn tijdens elke test alleen de volgende programma's op mijn computer geopend: Visual Studio, JMeter en Google Chrome.

Daarnaast heb ik ook de response tijd bijgehouden op de homepagina. Op deze pagina staat de map met alle listings erin. Deze pagina kon ik echter niet testen met JMeter, omdat ik maar een beperkt aantal requests naar Mapbox kan doen. Echter vind ik dit wel een belangrijke pagina waarop performance goed moet zijn. Vandaar dat ik er voor gekozen heb om hier wel de response tijd te tracken met de miniprofiler.

NULMETING

HOMEPAGE

Om de response op de homepage te testen heb ik een aantal keer de pagina opnieuw geladen. De response time bleef schommelen rond de 35ms.

36.7 ms

Ook heb ik de homepage getest met de minimum aantal reviews filter en de maximum price filter als volgt aan.

Neighbourhood:	All	Minimum nr of reviews:	5	Filter
Min price:		Max price:	200	

De response tijd die daar gemiddeld uit kwam lag rond de 30ms.

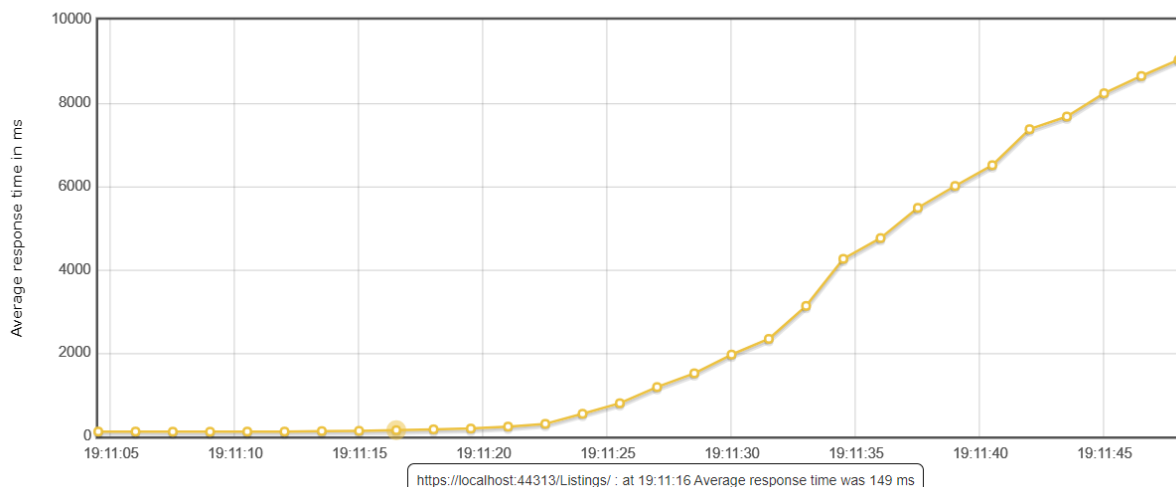
30.3 ms

LISTINGS

De nulmeting voor de listings pagina is gedaan met de instellingen in JMeter zoals in de figuur hieronder. Zoals te zien in de onderstaande afbeeldingen ligt de operational ceiling ongeveer rond de 20 seconden. Ook is de response tijd erg belangrijk. Deze ligt op dit moment rond de 150ms in de sweet spot.

Target Rate (arrivals/sec):	50
Ramp Up Time (sec):	30
Ramp-Up Steps Count:	10
Hold Target Rate Time (sec):	5

Response Times Over Time

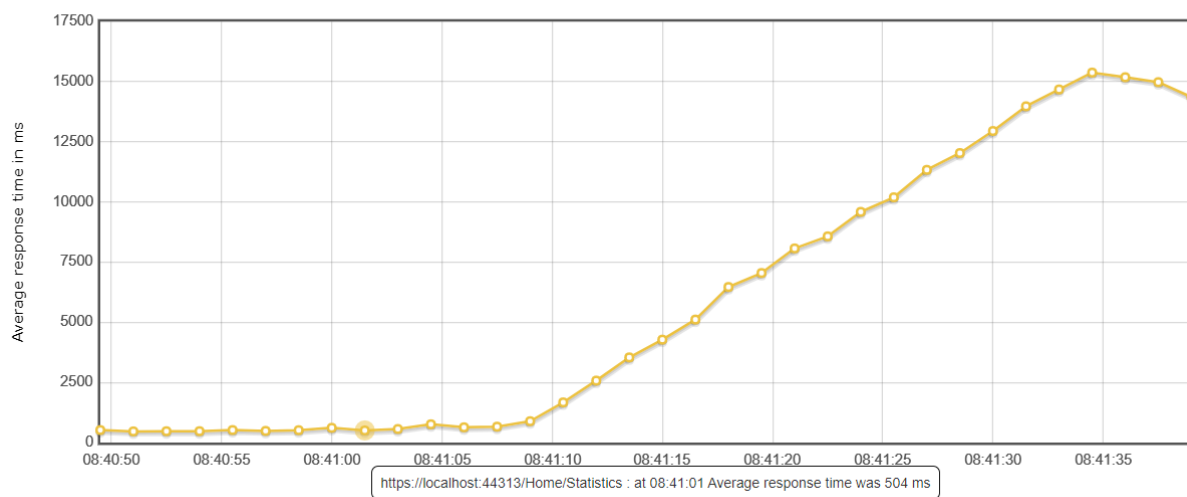


STATISTICS

De nulmeting voor de statistics pagina is gedaan met de instellingen in JMeter zoals in de figuur hieronder. Zoals te zien in de onderstaande afbeeldingen ligt de operational ceiling ongeveer rond de 20 seconden. Ook is de response tijd erg belangrijk. Deze ligt op dit moment rond de 500ms in de sweet spot.

Target Rate (arrivals/sec):	20
Ramp Up Time (sec):	30
Ramp-Up Steps Count:	10
Hold Target Rate Time (sec):	5

Response Times Over Time



ASNOTRACKING

Een optimalisatie is het gebruik van AsNoTracking in LINQ queries. Standaard zijn queries die entity types retourneren tracking. Dit houdt in dat de veranderingen aan deze entities kunnen worden bewaard door SaveChanges aan te roepen. No tracking queries zijn daarom vooral handig in read-only scenarios. Ze zijn sneller om uit te voeren, omdat er geen change tracking informatie bijgehouden hoeft te worden (Microsoft, 2022).

HOME PAGE

Ten opzichte van de vorige keer bleven de response tijden van de ongefilterde request nog steeds hangen rond de 35ms. Hier zit dus nog geen zichtbare verbetering in.

34.2 ms

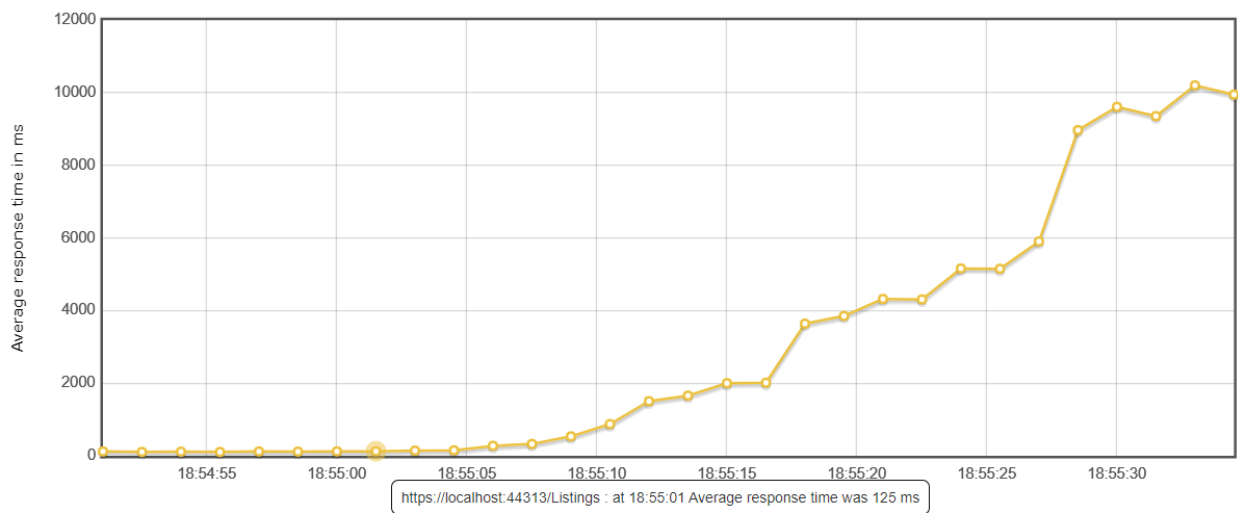
Ook de gefilterde request bleef steken rond de 30ms net als de vorige meting.

29.7 ms

LISTINGS

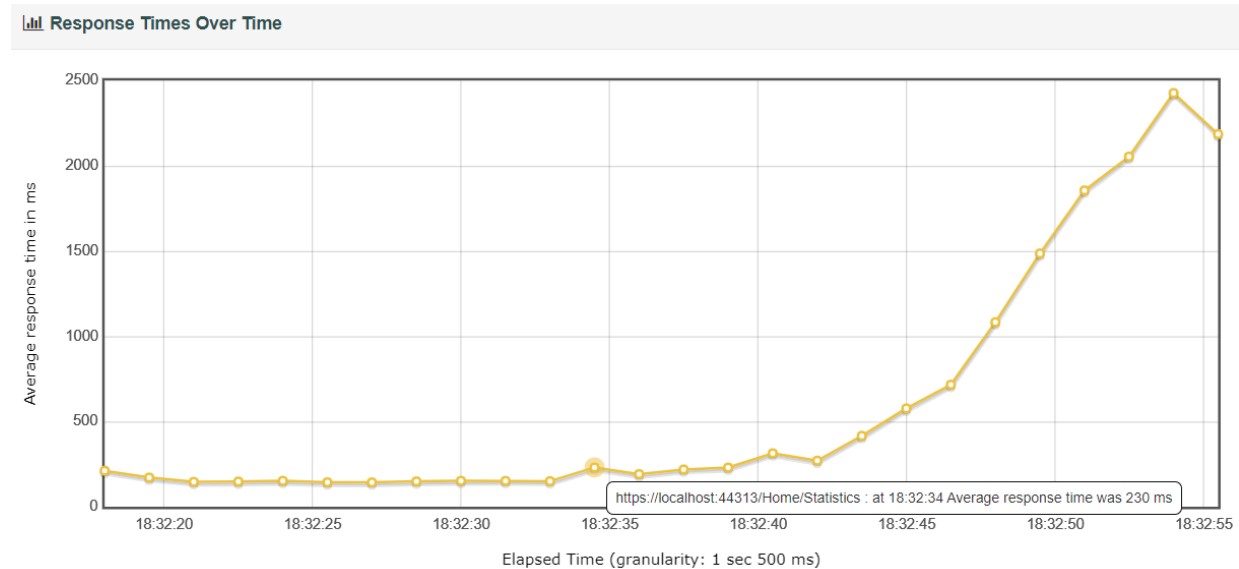
Deze meting is gedaan met dezelfde instellingen op de listings pagina als de nulmeting. De operational ceiling is op deze pagina eigenlijk hetzelfde gebleven. Hij is nu namelijk rond de 20 seconden en bij de nulmeting was dit ook het geval. Wel is er enige verbetering wat betreft de response time; bij de nulmeting lag deze rond de 150ms en nu ligt deze rond de 125ms in de sweet spot.

Response Times Over Time



STATISTICS

Deze meting is gedaan met dezelfde instellingen op de statistieken pagina als de nulmeting. De operational ceiling ligt nu iets verder, namelijk rond de 24 seconden. Deze lag bij de nulmeting nog rond de 20 seconden, dus hier zit al verbetering in. Ook wat betreft de response time zien we verbetering; bij de nulmeting lag deze rond de 500ms en nu ligt deze rond de 200ms in de sweet spot.



RESPONSE COMPRESSION

Response compression zorgt ervoor dat bestanden die verstuurd worden kleiner worden gemaakt. Hierdoor is de response die de client moet ontvangen kleiner. Dit moet ervoor zorgen dat de response sneller aankomt bij de client.

HOMEPAGE

Ook met response compression zit er weinig verbetering in de response tijden van de ongefilterde homepage. Opnieuw bleven deze namelijk schommelen rond de 35ms.

34.8 ms

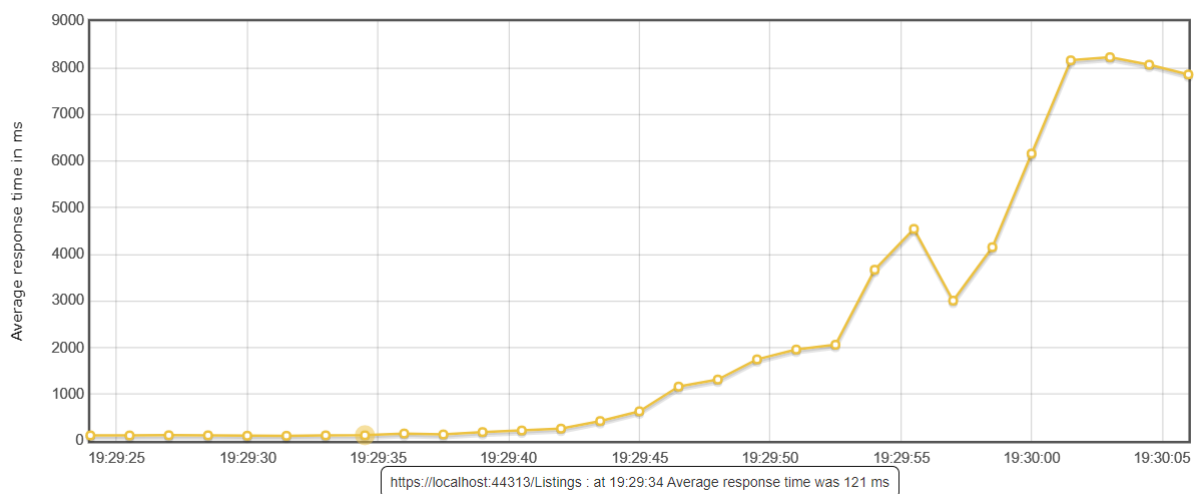
Ook bij de gefilterde requests is er opnieuw gemiddeld weinig verbetering te zien en blijven de requests schommelen rond de 30ms.

26.3 ms

LISTINGS

Deze meting is gedaan met dezelfde instellingen op de listings pagina als de vorige meting. De operational ceiling is op deze pagina eigenlijk hetzelfde gebleven. Hij is nu namelijk rond de 20 seconden en bij de vorige meeting was dit ook het geval. Ook de response time is niet echt veranderd. Deze ligt net als de vorige meting rond de 125ms

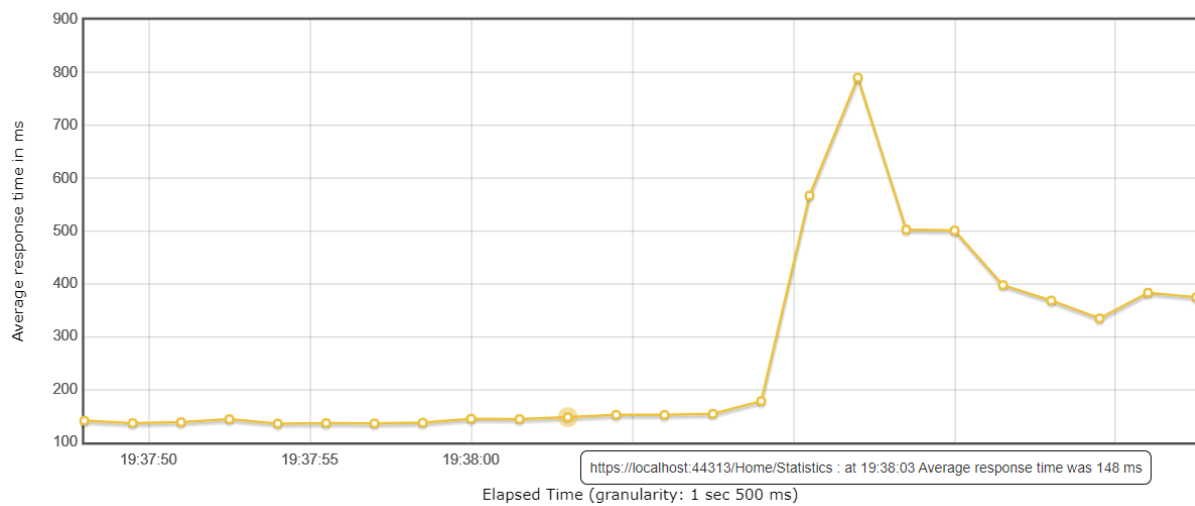
Response Times Over Time



STATISTICS

Deze meting is gedaan met dezelfde instellingen op de statistieken pagina als de vorige meting. De operational ceiling is ongeveer gelijk gebleven als de vorige meting. In deze meting ligt hij namelijk op 21 seconden. Wel zien we weer een flinke verbetering bij de response time. Deze lag bij de vorige meting rond de 200ms in de sweet spot en nu is dit rond de 150ms.

Response Times Over Time



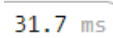
INDEXING

Indexing zorgt ervoor dat de database de benodigde data sneller kan opzoeken. Dit werkt vooral goed als je data wil filteren met WHERE. Ik heb er dus ook voor gekozen hiervoor geen meting te doen voor de listings en statistics pagina, omdat er geen joins of where's worden gebruikt in de benodigde queries voor deze pagina's.

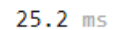
HOMEPAGE

Omdat er op de homepage gefilterd wordt op neighbourhood, price en nr of reviews heb ik een index gemaakt in de database met deze kolommen in de index key columns en de andere benodigde kolommen in de included columns.

Nu zien we wel enige verbetering qua performance op de homepagina. Bij de ongefilterde requests schommelde de response time waardes eerst rond de 35ms en nu is dit vaker rond de 30ms. Ook bij het filteren is er verbetering te zien. Eerst schommelde de response time waardes eerst rond de 30ms en nu is dit vaker richting de 25 en richting de lage 20.



31.7 ms



25.2 ms

CACHING

Caching is het in de cache opslaan van gegevens. Hierbij worden gegevens opgeslagen in een tijdelijke opslag waar de applicatie snel toegang tot heeft. Dit is vooral nuttig bij read operaties waarbij je alleen gegevens wil zien. Caching is toegepast op alle drie de pagina's.

HOMEPAGE

Deze metingen zijn gedaan op een andere dag dan de vorige metingen en ik merk dat de laptop waarop ik de metingen doe op dit moment iets trager is, terwijl dezelfde programma's open staan als bij de vorige metingen. Op deze pagina worden alle listings die opgehaald worden voor de map en ook alle neighbourhoods gecached. De resultaten van een gemiddelde meting zijn hieronder afgebeeld.

Zonder caching:

	duration (ms)	with children (ms)	from start (ms)	sql (ms)
https://localhost:44313/	0.2	58.3	+0.0	
Controller Action: inside_airbnb.Controllers.HomeContro...	46.5	46.5	+0.4	2.8 (3)
ViewResult	3.6	11.1	+47.0	
View: /Views/Home/Index.cshtml	7.2	7.2	+47.1	

Met caching:

	duration (ms)	with children (ms)	from start (ms)	sql (ms)
https://localhost:44313/	0.2	44.3	+0.0	
Controller Action: inside_airbnb.Controllers.HomeContro...	32.4	32.4	+0.7	1.5 (1)
ViewResult	3.8	11.0	+33.1	
View: /Views/Home/Index.cshtml	7.0	7.0	+33.2	

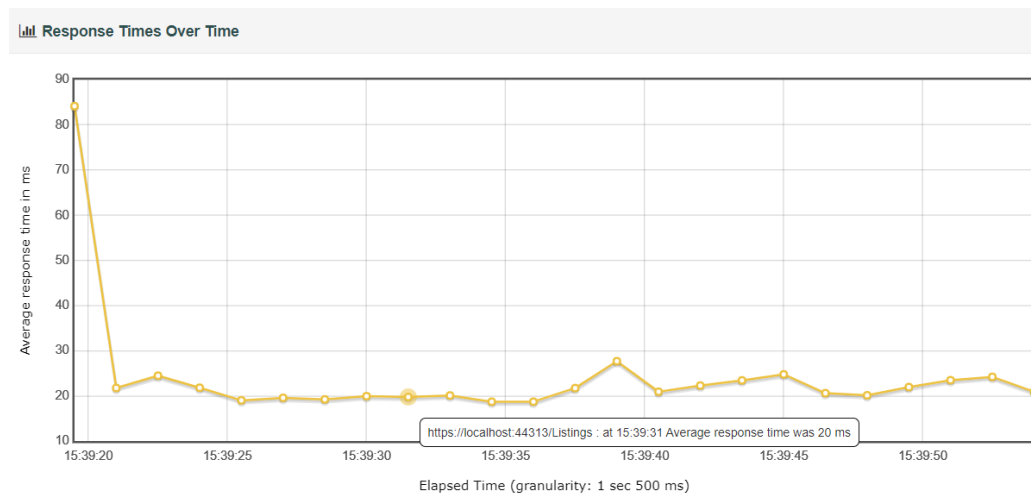
Zonder caching ligt de tijd die het uitvoeren van de queries duurt al rond de 2ms à 3ms. Dit is helemaal niet lang. Dat is wel opvallend, omdat wel alle listings worden opgehaald. Met caching geïmplementeerd is de response tijd dus een verwaarloosbaar verschil, omdat de tijd dat het uitvoeren van de queries duurt zonder caching al kort is.

LISTINGS

De eerste meting met caching op deze pagina is opnieuw gedaan met de originele settings in JMeter. Opvallend is dat de eerste request rond de 90ms ligt en de requests daarna allemaal rond de 20ms. Dit komt doordat de benodigde query voor deze pagina na de eerste load gecached is.

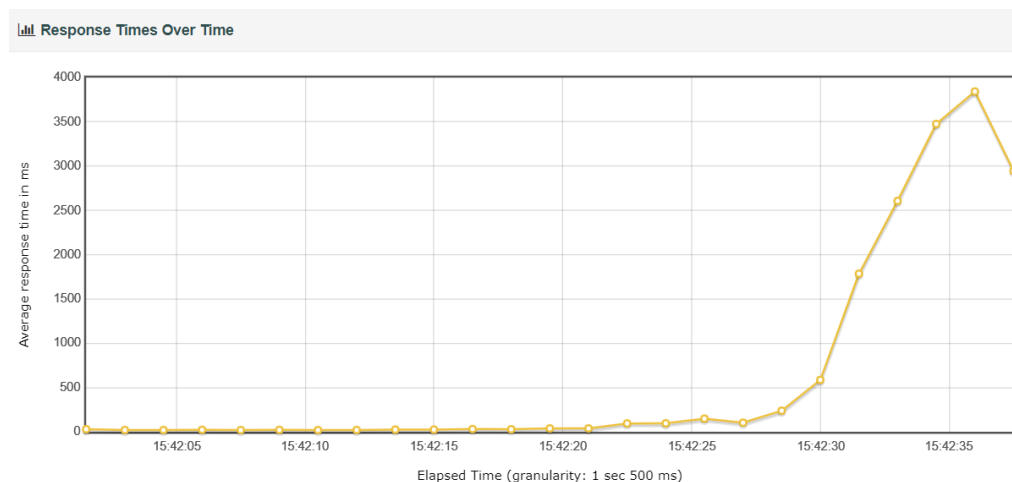
De response tijden zijn ten opzichte van de vorige meting op deze pagina dus enorm verbeterd, want eerst lag de gemiddelde response tijd rond de 125ms en nu rond de 20ms op het moment dat de response gecached is.

Ook is er geen operational ceiling meer te zien in deze grafiek. Dit laat zien dat de server nu ook veel meer requests aan kan.



Voor een nieuwe operational ceiling kon de target rate verhoogt worden van 50 arrivals/sec naar maar liefst 190 arrivals/sec!

Target Rate (arrivals/sec):	190
Ramp Up Time (sec):	30
Ramp-Up Steps Count:	10
Hold Target Rate Time (sec):	5

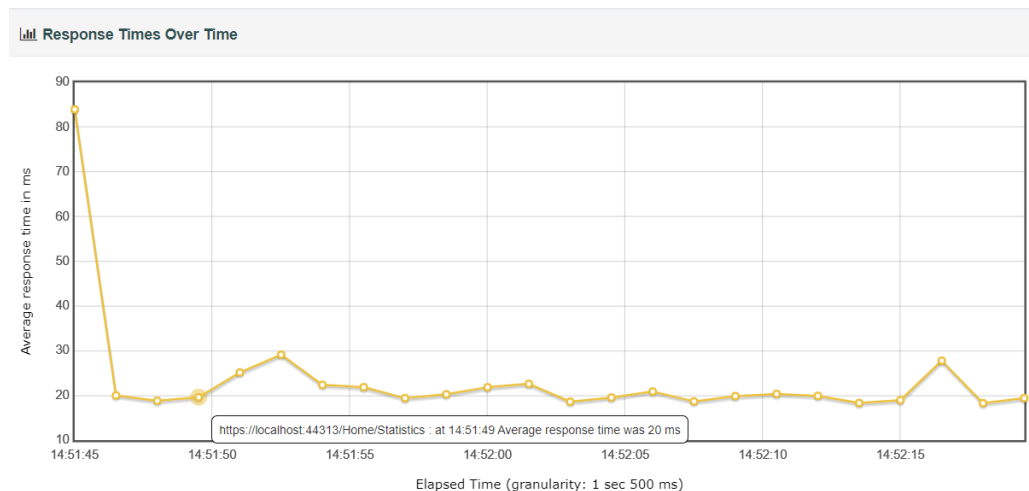


STATISTICS

De eerste meting met caching op deze pagina is opnieuw gedaan met de originele settings in JMeter. Net als bij de listings ligt de eerste request rond de 90ms en de requests daarna allemaal rond de 20ms.

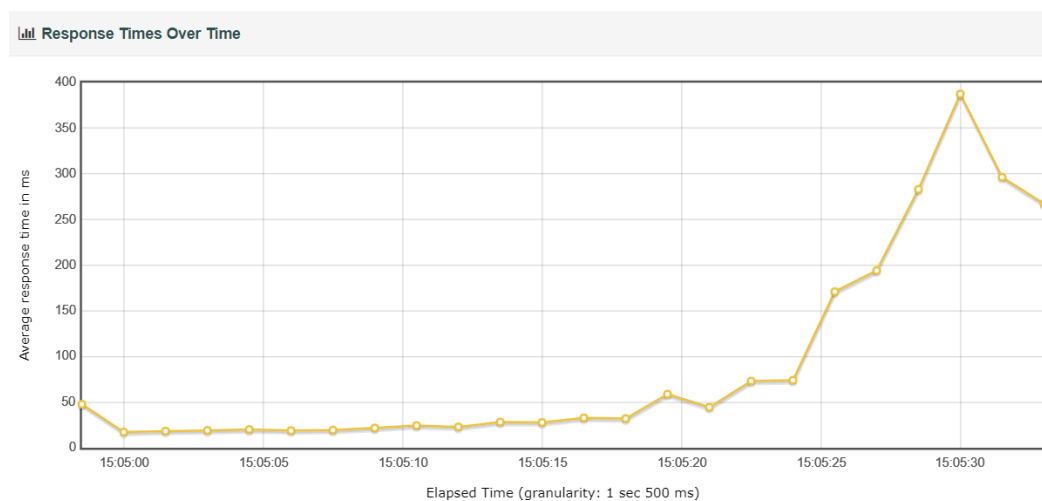
Ook bij deze pagina zijn de response tijden ten opzichte van de vorige meting dus enorm verbeterd, want eerst lag de gemiddelde response tijd rond de 150ms en nu rond de 20ms op het moment dat de response gecached is.

Ook hier is net als bij de listings geen operational ceiling meer te zien.



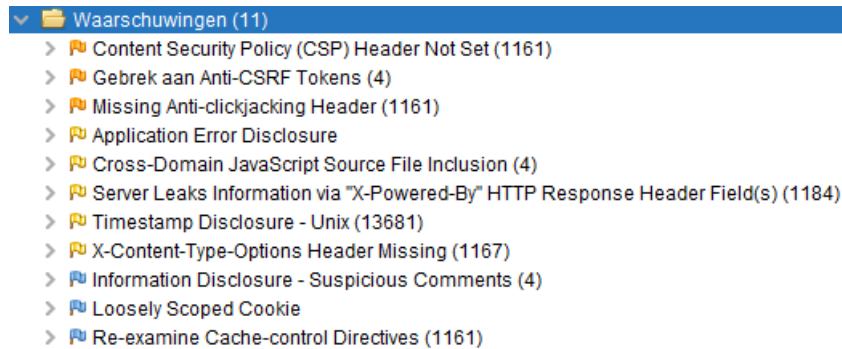
De target rate kon ook hier verhoogt worden naar 190 arrivals/sec. Op de statistics pagina is de performance winst nog groter dan bij de listings, omdat de originele target rate bij de statistics pagina lager lag.

Target Rate (arrivals/sec):	190
Ramp Up Time (sec):	30
Ramp-Up Steps Count:	10
Hold Target Rate Time (sec):	5



SECURITY

In dit hoofdstuk staan verschillende security verbeteringen die ik heb gedaan voor de applicatie. Om een overzicht te krijgen van welke potentiële security issues de applicatie is er een scan gedaan met de OWASP ZAP tool. Hieronder staat een overzicht met alle waarschuwingen na de eerste scan.



CONTENT SECURITY POLICY (CSP) HEADER NOT SET

Deze waarschuwing kan opgelost worden door het toevoegen van de Content-Security-Policy response header. Door het expliciet vermelden welke resources geladen mogen worden voorkomt deze header voornamelijk Cross Site Scripting.

De toegevoegde header paste niet volledig in de afbeelding. Als basis voor de Content-Security-Policy voor deze applicatie is de starter policy van <https://content-security-policy.com/> gebruikt. Daaraan zijn nog een aantal scripts van Mapbox en Chart.js toegevoegd die in de applicatie worden gebruikt. In de CSP is bij de CSS file van Mapbox is echter wel het keyword 'unsafe-inline' toegevoegd, omdat deze anders niet wilden werken. Het gebruik van dit keyword zou eigenlijk vermeden moeten worden, omdat dit de kans op het uitvoeren van kwaadaardige scripts vergroot.

```
app.UseSecurityHeaders(policies =>
    policies
        .AddContentSecurityPolicy(builder =>
            {
                builder.AddDefaultSrc()
                    .None();

                builder.AddScriptSrc()
                    .Self()
                    .From("https://api.mapbox.com/mapbox-gl-js/v2.8.2/mapbox-gl.js")
                    .From("https://cdnjs.cloudflare.com/ajax/libs/Chart.js/3.7.1/chart.min.js")
                    .WithNonce();

                builder.AddConnectSrc()
                    .Self()
                    .From("https://*.tiles.mapbox.com")
                    .From("https://api.mapbox.com")
            })
);
```

MISSING ANTI-CLICKJACKING HEADER

Deze waarschuwing kan opgelost worden door de X-Frame-Options header op 'deny' te zetten. Dit houdt in dat de browser de pagina niet in een <iframe> mag laden. Hiermee kan click-jacking voorkomen worden waarbij een gebruiker denkt dat hij bijvoorbeeld op een button op deze site klikt, maar in werkelijkheid wordt de site geladen in een andere site en klikt hij daar op een kwaadaardige link.

```
app.Use(async (context, next) =>
{
    context.Response.Headers.Add("X-Frame-Options", "Deny");

    await next();
});
```

APPLICATION ERROR DISCLOSURE

Application Error Disclosure houdt in dat een error of waarschuwing geeft waarin gevoelige informatie staat. Deze informatie kan een hacker bijvoorbeeld gebruiken voor een aanval op de applicatie. Mijn applicatie liet een pagina met een SQL error zien op het moment dat de query parameter 'pageNumber' op 0 werd gezet. Dit heb ik opgelost met een extra check die kijkt of de ingevoerde paginanummer groter is dan 0.

```
if (pageNumber > 0)
{
    listings = listings.Skip(((pageNumber - 1) * pageSize));
}
```

SERVER LEAKS INFORMATION

Deze waarschuwing kan opgelost worden door de X-Powered-By header niet meer mee te sturen met de response. X-Powered-By stuurt namelijk met de response mee vanaf wat voor server de response komt. Door deze header niet meer mee te sturen wordt het een eventuele hacker minder makkelijk gemaakt dit te achterhalen.

```
<httpProtocol>
  <customHeaders>
    <remove name="X-Powered-By" />
  </customHeaders>
</httpProtocol>
```

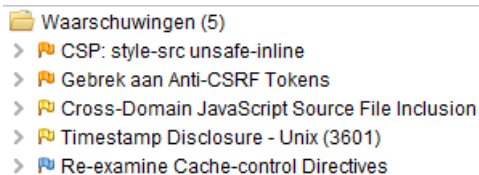
X-CONTENT-TYPE-OPTIONS HEADER MISSING

Deze waarschuwing kan opgelost worden door de X-Content-Type-Options header op 'nosniff' te zetten. Dit voorkomt dat de browser het media type (MIME type) dat de response terug stuurt gaat raden. Hierbij kijkt de browser naar de content die de response terugstuurt en aan de hand daarvan probeert het te raden van wat voor media type de content is. Dit is vooral belangrijk op websites waar gebruikers bestanden kunnen uploaden.

```
<httpProtocol>
  <customHeaders>
    <remove name="X-Powered-By" />
    <add name="X-Content-Type-Options" value="nosniff" />
  </customHeaders>
</httpProtocol>
```

OPENSTAANDE WAARSCHUWINGEN

In de afbeelding hieronder staan alle nog openstaande waarschuwingen. Onder de afbeelding staat extra toelichting hierop.



Waarschuwingen (5)

- > CSP: style-src unsafe-inline
- > Gebrek aan Anti-CSRF Tokens
- > Cross-Domain JavaScript Source File Inclusion
- > Timestamp Disclosure - Unix (3601)
- > Re-examine Cache-control Directives

CSP: style-src unsafe-inline

Hierover wordt toelichting gegeven in het gedeelte over Content Security Policy.

Gebrek aan Anti-CSRF Tokens

Cross Site Request Forgery is een aanval waarbij een ingelogde gebruiker een request doen naar de website waarop ze zijn ingelogd zonder dat ze de bedoeling hebben om die actie uit te voeren. Hierbij gaat het om state changing requests zoals het veranderen van een wachtwoord of het kopen van een product. Aangezien in deze applicatie geen POST, PUT of DELETE operaties te vinden zijn is er geen gebruik gemaakt van anti-CSRF tokens.

Cross-Domain JavaScript Source File Inclusion

Deze waarschuwing heeft betrekking op het feit dat er een JavaScript bestand wordt geladen op de website. Dit bestand is een bestand van Mapbox en is dus geen kwaadaardig bestand.

Timestamp Disclosure – Unix

De OWASP tool herkent ID's van listings als timestamp, terwijl dit geen timestamps zijn.

VERWIJZINGEN

Inside Airbnb. (sd). *Inside Airbnb: Get the Data*. Opgeroepen op Mei 23, 2022, van Inside Airbnb: <http://insideairbnb.com/get-the-data>

Microsoft. (2022, November 11). *Tracking vs. No-Tracking Queries*. Opgeroepen op Mei 23, 2022, van Microsoft Build: <https://docs.microsoft.com/en-us/ef/core/querying/tracking>