Datagedreven webapplicatie getest voor high performance en security door Michelle Jolink, 07-06-2024, Course NotS-WAPP

Ontwerpdocument Inside AirBnB Parijs

2019

Inhoudsopgave

[1. Inleiding 2](#_Toc168845863)

[2. Functioneel ontwerp 3](#_Toc168845864)

[2.1 Use Cases 3](#_Toc168845865)

[3. Technisch ontwerp 4](#_Toc168845866)

[3.1 Request pipeline 4](#_Toc168845867)

[3.2 Packages 5](#_Toc168845868)

[Backend 5](#_Toc168845869)

[Frontend 6](#_Toc168845870)

[4. Performance 7](#_Toc168845871)

[4.1 Listings 7](#_Toc168845872)

[Nulmeting listings 7](#_Toc168845873)

[AsNoTracking 8](#_Toc168845874)

[Resultaten AsNoTracking 9](#_Toc168845875)

[Redis Caching 10](#_Toc168845876)

[Indexing 11](#_Toc168845877)

[Conclusie 12](#_Toc168845878)

[4.2 Statistieken admins 12](#_Toc168845879)

[/Listings/GetAverageNightsPerMonth 12](#_Toc168845880)

[Indexing 13](#_Toc168845881)

[4.4. Conclusie 14](#_Toc168845882)

[5. Security 15](#_Toc168845883)

[5.1. OWASP ZAP 15](#_Toc168845884)

[Gevonden security issues 15](#_Toc168845885)

[X-Content-Type-Options Header Missing 15](#_Toc168845886)

[5.2 Package vulnerabilities 16](#_Toc168845887)

[5.2.1 Frontend 16](#_Toc168845888)

[5.2.2 Backend 17](#_Toc168845889)

[5.2 Backend authenticatie en autorisatie 18](#_Toc168845890)

[5.3 Conclusie 18](#_Toc168845891)

# Inleiding

Dit document beschrijft het ontwerp van de applicatie “InsideAirbnb”. Het doel van deze applicatie is het management van InsideAirbnb inzicht te geven in het gebruik van Airbnb-locaties in Parijs. Het management van InsideAirbnb wil graag inzicht krijgen in diverse statistieken, zoals het gemiddelde aantal overnachtingen per maand, de opbrengsten per buurt en de gemiddelde beoordelingen per buurt. Het ontwerp volgt de look-and-feel van de publieke site insideairbnb.com, zodat het gebruik ervan consistent aanvoelt.

In dit verslag worden de volgende onderwerpen behandeld:

**Usecases en Requirements:**

* Gedetailleerde beschrijving van de functionele en niet-functionele eisen waaraan de applicatie moet voldoen.
* Specifieke usecases die de verschillende interacties en functionaliteiten van de applicatie beschrijven.

**Architectuur:**

* Overzicht van de systeemarchitectuur, inclusief de verschillende componenten en hun interacties.
* Toelichting op de keuzes voor de architecturale opzet en hoe deze bijdraagt aan de doelstellingen van de applicatie.

**Frameworks en Packages:**

* Beschrijving van de gebruikte frameworks en softwarepakketten.
* Redenering achter de selectie van deze tools en hoe ze bijdragen aan de functionaliteit en efficiëntie van de applicatie.

**Performance:**

* Analyse van de prestaties van de applicatie.
* Inzichten verkregen uit load tests, inclusief de maatregelen die genomen zijn om de prestaties te optimaliseren.

**Security:**

* Overzicht van de beveiligingsmaatregelen die zijn geïmplementeerd om de data en de applicatie te beschermen.
* Beschrijving van de technieken en best practices die zijn gevolgd om ervoor te zorgen dat de applicatie veilig is tegen mogelijke bedreigingen.

# Functioneel ontwerp

## Use Cases

|  |  |
| --- | --- |
| Use case | Prioriteit |
| Registeren en inloggen. | Must |
| Filter op prijs. | Must |
| Filter op buurt. | Must |
| Filter op review. | Must |
| Kaart is clickable, details rechts op pagina, maakt gebruik van de mapbox API. | Must |
| Details per item waarop gefiltered is: #overnachtingen, #opbrengst in de maand. | Must |
| Er moeten rollen toegevoegd en toegekend worden aan geregistreerde gebruikers. | Must |
| Resultaten zoals trends, totalen, gemiddelden, etc. worden weergegeven in charts, alleen te bekijken voor ADMINS. Denk daarbij aan bv. Gemiddelde beschikbaarheid per maand, gemiddelde beschikbaarheid per buurt, overzicht van gemiddelde huurprijs per buurt. Andere managementoverzichten zijn ook mogelijk, ga daarvoor op zoek naar online voorbeelden | Must |
| Layout idem als insideairbnb.com. | Could |
| Locaties van zoekresultaat zichtbaar op kaart. | Could |

*Tabel 1: Use cases*

# Technisch ontwerp

In dit hoofdstuk wordt er meer verteld over de technische ontwerpkeuzes die gemaakt zijn voor de applicatie.

De gegevens voor de inside airbnb applicatie worden opgehaald uit een MSSQL-database. De data uit de database is geïmporteerd door middel van een .bacpac bestand van Inside Airbnb, en dan specifiek de data van Parijs. De data wordt opgevraagd via een ASP.NET CORE Web API die draait op .NET versie 8.0. Dit is de nieuwste stabiele versie van .NET en zo kunnen de meest recente functies binnen .NET gebruikt worden.

Om een interface aan de gebruikers te bieden, is er gekozen om het React framework te gebruiken. React is een populaire JavaScript framework die wordt gebruikt voor het bouwen van gebruikersinterfaces voor webapplicaties. Deze frontend applicatie communiceert met de API. Hierdoor zijn de gebruikers in staat om de usecases in tabel 1 uit te voeren.

Om dezelfde look en feel van insideairbnb.com te creeëren, wordt de Mapbox API gebruikt.

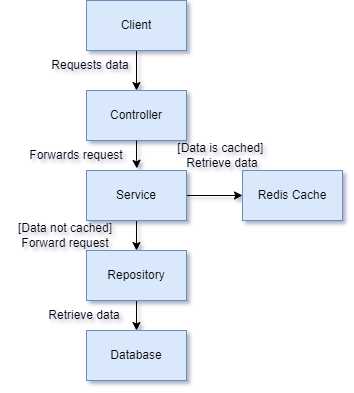
## 3.1 Request pipeline

De request pipeline wordt weergegeven in figuur 1. Wanneer de client een request doet, wordt dit afgehandeld door de API controller. Om de nodige informatie te verkrijgen, doet de controller een oproep naar de bijbehorende service. In de applicatie zijn er twee services beschikbaar: de listingservice en de neighbourhoodservice.

Binnen de service wordt eerst gecontroleerd of de benodigde informatie al in de Redis cache aanwezig is. Als dat het geval is, wordt deze informatie door de service direct teruggegeven aan de controller. Als de benodigde informatie nog niet in de cache aanwezig is, wordt er een oproep gedaan naar de bijbehorende repository om de informatie uit de database op te halen.

De opgehaalde informatie wordt vervolgens in de Redis cache opgeslagen voordat deze naar de controller wordt teruggestuurd. De controller verwerkt de informatie en stuurt deze uiteindelijk door naar de client. Deze aanpak zorgt voor efficiënte gegevensverwerking en minimaliseert de belasting van de database door gebruik te maken van caching.

Door het toepassen van het repository pattern wordt het seperation of concerns ontwerpprincipe toegepast <https://www.geeksforgeeks.org/separation-of-concerns-soc/> . Dit zorgt ervoor dat de code beter onderhoudbaar en te testen is, en er ook makkelijk verandert kan worden van database. <https://www.geeksforgeeks.org/repository-design-pattern/>



Figuur 1: Request pipeline

## 3.2 Packages

Tijdens het ontwikkelen van de inside airbnb applicatie zijn er verschillende packages gebruikt.

### Backend

In figuur 2 staan alle packages die gebruikt zijn voor het realiseren van de API.

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Figuur 2: Packages backend

### Frontend

In figuur 3 is het package.json bestand te zien waarin alle package dependencies staan die gebruikt zijn om de frontend te realiseren.

Afbeelding met tekst, schermopname

Automatisch gegenereerde beschrijving

Figuur 3: Packages frontend

# 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 en Redis caching en als laatste de verbeteringen met database indexing.

## 4.1 Listings

### Nulmeting listings

Om de verbeteringen in performance van de applicatie goed in beeld te krijgen op basis van de aanpassingen wordt er een nulmeting gedaan. Deze test wordt uitgevoerd met NBomber. Hierbij heb ik mijn laptop in een zo schoon mogelijke staat gehouden. Dit betekent dat ik geen andere programma's open heb staan nodig voor de test en mijn laptop opnieuw heb opgestart voordat de test wordt uitgevoerd. Voor NBomber worden de volgende instellingen gebruikt:

Afbeelding met tekst, Lettertype, schermopname

Automatisch gegenereerde beschrijving

Figuur 4: Instellingen NBomber

Deze instellingen bootsen een scenario na waarbij de belasting op het systeem geleidelijk wordt verhoogd. Dit wordt gedaan met een snelheid van 40 aanvragen per twee seconden gedurende een periode van 30 seconden.

De nulmeting wordt gedaan op de API route die bij het opstarten van de applicatie alle 74.000+ listings ophaalt. Deze route haalt alleen de longitude, latitude en id op.

De aangesproken endpoint voor het bepalen van de prestatie verbeteringen is:

* **GET /api/Listings/geoinfo**

Afbeelding met schermopname, Perceel, lijn, tekst

Automatisch gegenereerde beschrijving

Figuur 5: NBomber grafiek van vertragingen bij nulmeting

Figuur 5 toont de vertragingen in milliseconden over de tijd van de NBomber test. Wat opvalt is dat na ongeveer 15 seconden de vertragingen sterk op beginnen te lopen en na 20 seconden 99% van de requests stijgen naar een vertraging van meer dan 3 seconden en hierna nog sterker oploopt.

Afbeelding met tekst, schermopname, lijn, Lettertype

Automatisch gegenereerde beschrijving

Figuur 6: Nbomber globale vertraging indicators nulmeting

|  |  |  |
| --- | --- | --- |
| **Minimale responsetijd** | **Maximale responsetijd** | **Gemiddelde responsetijd** |
| 0.2129 seconden | 20.0492 seconden | 9.44228 seconden |

Tabel 2: Minimale, maximale en gemiddelde reponsetijd bij nulmeting

Zoals te zien in figuur 6 duren aardig veel requests langer dan 1.2 seconden. Dit wordt ook duidelijk in tabel 2, waar we kunnen zien dat de gemiddelde responsetijd zit op 9 seconden. Dit gaan we verbeteren.

### 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. In het geval van het ophalen van de listings is dit het geval.

**Afbeelding met tekst, schermopname, software, Lettertype

Automatisch gegenereerde beschrijving**

Figuur 7: Code na toevoegen AsNoTracking

In figuur 7 is te zien hoe simpel deze aanpak kan worden toegevoegd aan de methode. Laten we kijken wat dit voor de performance heeft gedaan.

Afbeelding met schermopname, tekst, lijn, Perceel

Automatisch gegenereerde beschrijving

Figuur 8: NBomber grafiek van vertragingen na AsNoTracking

Zoals hierboven te zien in figuur 8 is de piek ongeveer 5 seconden vertraagd, want nu loopt de piek pas op na 20 seconden. Dit is een goed teken, want dit betekend dat de API het 5 seconden langer volhoudt dan voorheen.

Afbeelding met tekst, schermopname, lijn, Lettertype

Automatisch gegenereerde beschrijving

Figuur 9: Nbomber globale vertraging indicators na AsNoTracking

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Minimale responsetijd** | **Maximale responsetijd** | **Gemiddelde responsetijd** |
| ***Nulmeting*** | 0.21290 seconden | 20.0492 seconden | 9.44228 seconden |
| ***Na asNoTracking*** | 0.20309 seconden | 16.8268 seconden | 7.34436 seconden |
| ***Verbetering*** | 0,00981 seconden | 3,2224 seconden | 2,09792 seconden |

Tabel 2: Minimale, maximale en gemiddelde reponsetijd na AsNoTracking

Ook zien we in figuur 9 hierboven dat een deel van de requests die eerst in het gebied *t > 1.2s* zaten nu in het gebied tussen *800ms en 1.2s* zitten. In tabel 2 is te zien dat de minimale responsetijd minimaal is afgenomen, maar bij de maximale en gemiddelde responsetijd is al zeker verbetering te zien. De maximale responsetijd is al meer dan 3 seconden verbeterd en de gemiddelde responsetijd meer dan 2.

### Redis Caching

Redis-caching maakt gebruik van in-memory opslag, waardoor applicaties gegevens kunnen opslaan en opvragen met een zeer lage vertraging. <https://www.geeksforgeeks.org/redis-cache/>

In dit geval wordt er gebruik gemaakt van een Redis server die runt via een Docker Container. Ik heb Redis Caching op de volgende manier in mijn backend toegevoegd:

In het bestand *Program.cs* maak ik verbinding met mijn Redis server, zie figuur 10.

Figuur 10: Voeg Redis toe aan serverconfiguratie

Hierna kan ik in mijn repository de redis cache kan gebruiken:

// TODO caching naar services plaatsen!!!!!!

Afbeelding met tekst, schermopname, software, Besturingssysteem

Automatisch gegenereerde beschrijving

Na de test te runnen in dezelfde omgeving en dezelfde instellingen met NBomber was dit het resultaat:

Afbeelding met schermopname, lijn, Perceel, tekst

Automatisch gegenereerde beschrijving

Zoals te zien is, is de piek aan het begin erg hoor, maar dat is te verwachten want dan staan de listings nog niet opgeslagen in de cache. Naar mate de tijd vordert zien we dat de lijnen ongeveer constant blijven lopen op een lage responsetijd.

Afbeelding met tekst, schermopname, lijn, Lettertype

Automatisch gegenereerde beschrijving 

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Minimale responsetijd** | **Maximale responsetijd** | **Gemiddelde responsetijd** |
| ***Nulmeting*** | 0.21290 seconden | 20.0492 seconden | 9.44228 seconden |
| ***AsNoTracking*** | 0.20309 seconden | 16.8268 seconden | 7.34436 seconden |
| ***Redis Caching*** | 0.13358 seconden | 2.37913 seconden | 0.26871 seconden |
| ***Verbetering*** | 0,06951 seconden | 14,44767 seconden | 7,07565 seconden |

Zoals te zien in tabel … is de maximale responsetijd nu met 14,4 seconden afgenomen en de gemiddelde responsetijd met 7 seconden! Dit kunnen we stellen als een behoorlijke performance winst.

### Indexing

Om te kijken of we de snelheid van de query kunnen verbeteren wil ik kijken of dit verbeterd kan worden door een index toe te voegen aan de tabel. Aangezien er al een PK staat op ID ga ik kijken of een index op longitude en latitude de query sneller kan maken.

Voor verbetering  


Na het toevoegen van een index op longitude en latitude:



Zoals te zien is er geen verbetering.

### Conclusie

Als we kijken naar de gemiddelde en maximale responsetijd, dan is de gemiddelde responsetijd met ongeveer 97,16% toegenomen, en de maximale responsetijd met ongeveer 88,11%. Hierdoor kan ik concluderen dat AsNoTracking en Redis hierbij goed geholpen hebben de performance van de applicatie te verbeteren.

97,16% = ((9.44228 - 0.26871) / 9.44228) \* 100

88,11% = ((2.37913 - 20.0492) / 20.0492) \* 100

## 4.2 Statistieken admins

Een endpoint /Listings/GetAverageNightsPerMonth en /Listings/stats/revenue-per-month om admin statistieken data op te halen voert een JOIN uit en daarnaast een where en is erg langzaam bij aanroepen. Hiervoor wil ik ook de performance verbeteren. Om de query performance te meten zal ik Miniprofiler gebruiken.

### /Listings/GetAverageNightsPerMonth

Query:

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

Voor verbetering:



Voor performance verbetering



### Indexing

Omdat ik al een PK heb op de listing\_id en date, zijn er nog 2 velden over waar het belangrijk is om een index op toe te voegen, namelijk de kolom available en minimum\_nights. Aangezien op minimum\_nights gezocht wordt en gefilterd wordt op available

Afbeelding met tekst, schermopname, Lettertype, lijn

Automatisch gegenereerde beschrijving

Na de non-clustered index toegevoegd te hebben is dit het resultaat:



Zoals te zien is er geen verbetering qua performance. Ik ben toen de query gaan inspecteren met execution plan in management studio. Hier werd aangegeven om een index te plaatsen op available en om minimum\_nights toe te voegen als include kolom in plaats van een index kolom.

Afbeelding met tekst, schermopname, Lettertype, lijn

Automatisch gegenereerde beschrijving

Afbeelding met tekst, schermopname, Lettertype, lijn

Automatisch gegenereerde beschrijving

Dit was het resultaat na toevoegen van deze index:

Zoals te zien is er nu wel een performance verbetering behaald van ongeveer 4 seconden.



Na het toevoegen van de index in figuur….(die werkende bij vorige route) is er al wat performance verbetering te zien van ongeveer 3 seconden.



## 4.4. Conclusie

**TODO**

# 5. Security

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

## 5.1. OWASP ZAP

Om security issues op te sporen zal ik met ZAP een scan uitvoeren op mijn API: https://localhost:7049/

Om de scan uit te voeren gebruik ik 1 specifieke route, omdat je met ZAP niet de base url mee kang even end at ZAP zelf de endpoints vind. Ik heb gecontroleerd of bij verschillende routes verschillende waarschuwingen egtoond warden, maar dit was niet het geval. Daarom ga ik de scan uitvoeren op ***/Listings/geoinfo.***

### Gevonden security issues

Afbeelding met tekst, schermopname, Lettertype, nummer

Automatisch gegenereerde beschrijving

### 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 waarbij ik de header ‘*X-Content-Type-Options*’. Zie figuur…

Afbeelding met tekst, schermopname, Lettertype, lijn

Automatisch gegenereerde beschrijving

Strict-Transport-Security Header Not Set

Deze header geeft aan dat het ook mogelijk is dat de gebruiker van het HTTP protocol gebruikt kan maken. Dit is natuurlijk niet altijd veilig. Daarom gaan we dit oplossen. Dit ga ik doen door bij hetzelfde codeblock als hierboven nog een header toe te voegen, genaamd ‘*Strict-Traponsport-Security*’

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

## 5.2 Package vulnerabilities

Om de beveiliging zo goed mogelijk te maken, controleert Snyk de dependencies op kwetsbaarheden in zowel de front- als backend van de applicatie.

### 5.2.1 Frontend

Om Snyk in mijn frontend applicatie te runnen, heb ik gebruik gemaakt van de extensie binnen Visual Studio Code. <https://docs.snyk.io/integrate-with-snyk/use-snyk-in-your-ide/visual-studio-code-extension>

Zoals hieronder te zien in figuur … heeft snyk in totaal 4 vulnerabilities gevonden, waarvan 1 wordt aangemerkt als een hoge vulnerability, de *nth-check* en 3 als medium, de *inflight*, *postcss* en *serialize-javascript*. Ook is te zien dat ze allemaal vanauit 1 dependency komen, namelijk *react scripts versie 5.0.1.*

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving



Om deze vulnerabilities op te lossen, moeten de nth-check, postcss en serialize-javascript dependencies geupgrade worden naar een hogere versie. Dit heb ik gedaan door de volgende code in de package.json:  
Afbeelding met tekst, Lettertype, schermopname, handschrift

Automatisch gegenereerde beschrijving

Voor de warning van inflight is helaas nog geen oplossing, zoals snyk zelf aangeeft:

Afbeelding met tekst, Lettertype, schermopname, wit

Automatisch gegenereerde beschrijving

Na alle dependencies warnings opgelost te hebben die mogelijk waren, heb ik een tweede scan uitgevoerd. Dit zijn dit de overgebleven vulnerabilities:

Afbeelding met tekst, schermopname, Lettertype

Automatisch gegenereerde beschrijving

### 5.2.2 Backend

Om Snyk in mijn backend applicatie te runnen, heb ik gebruik gemaakt van de extensie binnen Visual Studio. <https://docs.snyk.io/integrate-with-snyk/use-snyk-in-your-ide/visual-studio-extension>

Na de eerste scan te hebben uitgevoerd zijn er 3 vulnerabilities gevonden, waarvan 1 met een hoge severity, 1 medium en 1 low, zie figuur hieronder.

Afbeelding met tekst, schermopname, software, Lettertype

Automatisch gegenereerde beschrijving

Als je naar heb tabje Installed en dan show unly vulnerable aanklikt zien we ook precies die 3 dependencies:

Afbeelding met tekst, schermopname, software, Multimediasoftware

Automatisch gegenereerde beschrijving

Ik heb dit opgelost door de volgende packages te upgraden

***System.Text.RegularExpressions*** van versie *4.3.0* naar *4.3.1*

Hierdoor is de issue van System.Text.RegularExpressions opgelost.

***Azure.Identity*** van versie *1.10.3* naar *1.11.3*

Hierdoor zijn de issues van Azure.Identity en Microsoft.Identity.Client verholpen.

Na deze aanpassingen heb ik de 2e scan uitgevoerd en bevinden zich geen zwakheden meer in de packages van de backend:



## 5.2 Backend authenticatie en autorisatie

Om de data in de backend te beveiligen, is het Auth0-framework gebruikt. Door authenticatie bij de routes toe te passen moeten gebruikers eerst aangemeld zijn om toegang te krijgen tot alle routes. Bovendien kunnen alleen gebruikers met admin rechten, die beschikken over de permissie *"read:stats"* de gegevens voor de statistieken ophalen. Deze aanpak zorgt ervoor dat alleen bevoegde personen gegevens kunnen ophalen.

TODO fotos toevoegen

## 5.3 Conclusie

Door een ZAP-scan uit te voeren op de API, specifiek op de route /Listings/geoinfo, zijn er twee issues gedetecteerd. Dit waren het ontbreken van de X-Content-Type-Options en Strict-Transport-Security headers. Deze problemen zijn opgelost door de juiste headers toe te voegen aan de serverconfiguratie.

Daarnaast heb ik met behulp van Snyk de kwetsbaarheden in de dependencies van zowel de frontend als backend van de applicatie opgespoord en aangepakt. In de frontend applicatie zijn vier kwetsbaarheden gevonden en drie ervan konden worden opgelost door dependencies te upgraden. In de backend applicatie zijn drie kwetsbaarheden geïdentificeerd en allemaal opgelost door de packages te updaten.

Ook heb ik route authentication en authorization toegepast.

Al met al kunnen we concluderen dat de applicatie nu aanzienlijk verbeterd is vergeleken met de applicatie voor deze verbeteringen.