Data Freshness in Booking Support Systems

A study into the freshness of data and sample rates

Daniël Vos

Radboud Universiteit

IN·DEI·NOMINE·FELICITER

Institute for Computing Science
and Information Sciences
Radboud University
Netherlands

13-12-2021

Abstract

Accommodation booking and support platforms need to get their data from different sources. To make this possible these sources have to be coupled in a way which allows the platform to generalize the data and present it in a similar way. How this can be done is one of the things studied in this thesis. Another important aspect is that cached data is representative of the current situation in other words, the data has to be fresh. What the requirements are for this, how to ensure the freshness of data and how often data should be refreshed are other important subjects which are studied in this thesis.
The results of the study into the sample rates turned out to be inconclusive, nevertheless it is clear that a dynamic sample rate is the most effective and resource friendly way. A dynamic sample rate is the most effective if factors like month of the year and time of day are taken into account. Because there are almost no reservations made during the night and there are a lot less reservations made during the fall and winter in comparison to the spring and summer. The dynamic sample rate used in this thesis decreases the total amount of hours with wrong availabilities with more than 60%. As a result of the research limitations it is possible (but we aren't sure) that for a different type of camping maybe other periods of the year are more busy or the reservations might be evenly spread which makes a dynamic sample rate not the most effective solution.

Table of Contents

1 Introduction

1.1 Context

Glamping is camping with luxury, staying overnight in a luxury accommodation with many natural elements. In the past few years glamping keeps getting more popular in the Netherlands. In the previous year the interest in glamping has increased with almost 30%. When you are looking for a hotel holiday, you can go to booking.com, trivago.nl, etc. But there is no real alternative of such a size when you want to go glamping. In the Netherlands there are five big booking support systems for glamping, these five systems together own a market share of over 90%. These five booking support systems are: Booking Experts, Tommy Booking Support, Camping.care, Recranet and Maxxton. To be able to create a holiday booking and comparing site such like booking.com it's necessary to couple these five booking support systems with each other. This makes it possible to easily sign up your camping to the website when you use one of those systems.

1.2 Scope

Fully coupling the five big booking support systems with each other and creating a platform on which glamping holidays can be compared and booked is too big of a task for a thesis. Therefore the scope of this thesis is limited to a sub-problem of this task. In this thesis there will be research done into the freshness of the data which is going to be retrieved from the different booking support systems. For coupling the five booking support systems, accommodation data such as name, number of persons, minimum length of stay, description, etc. will be used. But for the freshness we will especially take a look at the availability and the prices of accommodations, because most of the other data doesn't change that much.

1.3 Goal

The goal of this thesis is to do research in which way the five booking support systems can be coupled and also to do this as a proof-of-concept. I'm going to solve some problems on the basis of my research.

1) We have to look at the support which the five booking support systems provide to external developers to retrieve data from their systems and to what extent a camping has a say in this.
2) We will take a look at how to integrate the multiple schemas from the different systems together.
3) We can shift the focus to data freshness, what should the sample rates be to keep the accuracy of the date high enough.

1.4 Contribution
This thesis contributes to the general research in the field of data freshness and contributes to Donselaar Groep because it helps solving their problem. This thesis is useful for Donselaar Groep because alongside the creation of this thesis there is also a working prototype of the beginning of their platform created. This prototype can later be expanded on and used for the real system. This thesis also does research into the sample rate of accommodation data. This is done on the basis of a literature study and practical research. The results of this study are also very relevant for Donselaar Groep, because they have an interest in potentially creating a booking platform which uses accommodations from as many camping's as possible. So handling the data in an efficient way is very important for them.

1.5 Research
There are three clear problems which should be solved to make sure that this thesis can be brought to a successful end.

1) Research will be done into the freshness of data, through the testing of different sample rates and measuring the corresponding accuracy of the data. However there is also research needed for the other problems.
2) Research must be done into the support which the five booking support systems provide to external developers, it is important to discover for each of those systems an appropriate way to retrieve data from them. All five of the systems look like having an API which can be used, however there must be looked at by who these API's are meant to be used and what is actually needed to get access to the data. Is this publicly available, does the booking support system have to provide access or can the camping themselves provide access to the data.
3) There is also research needed into which fields from the data are important for Donselaar Groep and how can those fields be generalized such that the used naming's of the fields can be made equal between the different systems.

1.6 Approach
From the start there will be a literature study done into the freshness of data, hereby will be looked at different aspects.

-When can we call data fresh, what is the definition according to the literature?

-How can you keep data from different sources fresh, does all the data have to be of the same freshness or can this differ?

-What is the "optimal sample rate" which can be used?

Simultaneously with this literature study, four other practical studies will be conducted sequentially. Starting with the research into what kind of support the five booking support systems provide in the form of their API's. Followed by research how the data can be generalized. After this, research will be done into which fields are important to Donselaar Groep. When this has been completed, research can be started into which names should be used for the fields of the various systems. During the literature study there will be worked on the scientific basis of this thesis and during the four practical researches there will be worked on a simple prototype of the system for Donselaar Groep.

1.7 Organization description

Donselaar Groep is a group of seven young companies. Six of these companies are active in the tent branch. These companies are:

Donselaar Tenten:

This is the specialist in the rental of temporary accommodations. They are partners with more than 100 festivals like Defqon.1, Lowlands, Mysteryland, Soenda and Awekenings. They don't only work on festivals or events, they also provide their services to companies, they deliver semi-permanent storage halls and replacement rooms.

Outstanding:

They deliver fully equipped safari tents and lodge tents internationally. These are ready to use glamping concepts which are delivered to camping's, tour operators and estates. Nowadays their glamping tents can be found in more than 50 countries and they work for big tour operators like Vacanceselect and Vacansoleil.

Villatent:

They provide exclusive glamping holidays to 40 quality camping's in Europe. They ensure that there are fully furnished Villatent's ready, which are fully equipped.

Rechargers:

Rechargers takes care of luxury glamping accommodations for festivals, companies and events.

Tent Trading:

They know all about tents and are more than ten years old. Tent Trading does everything themselves, from designing their tents to making construction calculations and from the production of their tents to the montage.

Eurostretch:

Eurostretch sells, develops and produces so called stretch tents, Bedouin tents and flex tents. They sell their products to an increasing amount of customers in Europe.

Then there is also the company Verdo Werkt, they are the link between people who are searching for a job and companies which are in need of people. They fill vacancies for more than 150 clients. The umbrella organization Donselaar Groep provides support for the seven above mentioned companies with branding, e-commerce, finance and human resources.

1.8 Research questions and sub questions

The main research question is:

-How to keep data from five big booking support systems fresh?

Sub research questions are:

-How to couple the five big booking support systems?
-Which data fields are important and how to integrate them?
-Which "sample rate" is optimal to use?

2 Theoretical Framework and Literature Study

2.1 What does data freshness mean

Data freshness is an abstract concept which doesn't have an exact definition in the literature. It is about the age of data and if it is still usable for users (Bouzeghoub, 2004). The freshness of data can be measured in different units, which unit should be used depends on the purpose of the data (Bouzeghoub, 2004). In general data freshness falls under the quality dimension (Jarke, Jeusfeld, Quix, & Vassiliadis, 1999), nevertheless it is possible to divide it into two sub dimension which belong to the quality dimension. The first sub dimension it the currency factor, this measures the time between the request of the data and the receiving of the data (Segev & Fang, 1989). The second sub dimension is the timeliness factor, which measures how often data changes or new data is created (Wang & Strong, 1996).

2.2 What are existing definitions for freshness of data

As mentioned before there are multiple different definitions for the freshness of data, these definitions are dependent on the use cases of the data. Four different definitions will be discussed here, the first one belongs to the timeliness factor, the other three belong to the currency factor (Bouzeghoub, 2004; Jarke, Jeusfeld, Quix, & Vassiliadis, 1999; Segev & Fang, 1989; Wang & Strong, 1996). The first definition for data freshness is the timeliness metric, this metric measures the extent to which the age of the data is appropriate for the goal for which the data is needed (Wang & Strong, 1996). In general this will be an estimation of the time passed since the last change to the data was done and this will be limited by the frequency of the changes to the data (Naumann, Leser, & Freytag, 2005). The second definition that is discussed is the obsolescence metric, which is about the amount of changes which have been done since the extraction time of the data. In caching systems obsolescence is often called age, which refers to the amount of changes which have been done to the data since the data was cached (Huang, Sloan, & Wolfson, 1994). The third definition is the freshness-rate metric, this one measures the percentage of data which is up-to-date. In caching systems this is often called freshness, which refers to the part of the data which is up-to-date in other words fresh (Bright & Raschid, Using latency-recency profiles for data delivery on the web., 2002). The fourth definition about the freshness of data which is discussed is the currency metric, which is about the elapsed time since the data has been changed without these changes reflected in the displayed data. In practice this is usually an estimation of the time between data extraction and data delivery. In caching systems this will often be described as age (where age is the time elapsed since the data got old (Huang, Sloan, & Wolfson, 1994)) or as recency (where recency is the time elapsed since the data was cached (Cho & Garcia-Molina, 2000)).

2.3 What will the definition of freshness of data be in this thesis

In this thesis there will be a different definition created for the freshness of data.

> *Definition: Fresh (data):*
> *Data will be called fresh if the following properties hold:*
> > *-The age of the data can't be greater than the chosen maximal age.*
> > *-The percentage of correct data should be greater than or equal to the chosen minimal freshness.*

To be able to understand the first property of the definition of freshness of data it is necessary to define what is meant with the age of the data.

> *Definition: Age (of data):*
> *The age of data is equal to the passed time since the data has been requested from the source.*

The definition of age (of data) looks like the definition of age when data freshness is interpreted as the currency metric. Nevertheless our definition of age is a bit different because we are not that much interested into the time elapsed since data got old (the age surpassed the maximal chosen age). The definition we have created for the freshness of data can be seen as a combination of the freshness-rate metric and the currency metric.

2.4 How to keep data from multiple sources fresh

Data from multiple sources can be kept fresh in a few different ways. These different ways can be classified into two main ways. Pulling and pushing, pulling the data happens on set intervals or on specified events and is done by a server requesting new data (Yates & Kaul, 2012; Pappas, Gunnarsson, Kratz, Kountouris, & Angelakis, 2015). Pushing data can happen on either set intervals or events like data changes and is done by the data source, which pushes the new data to a channel which a server is listening to (Yates & Kaul, 2012; Pappas, Gunnarsson, Kratz, Kountouris, & Angelakis, 2015). Pushing data requires the server to have multiple simultaneously open channels to which it listens, this can create queues while retrieving new data and is vulnerable to useless data changes like making a change to a specified field and a few moments later changing it back. This type of changes are often not necessary to push but will be pushed unless it's prevented in some way. A way to prevent this is to wait a set amount of time after changes are made before pushing, but when this is done, a part of the advantages of pushing the data disappears. The data is less fresh than when pushed directly and it's also more resource intensive to keep track of when to push the data. When pulling data it might happen that the data drastically changes just after it got pulled by a server. This means that those changes won't be represented until the next data pull. If this is or isn't a problem, is often dependent on the amount it changes and on the use case. In the end it's all dependent on the use case, the amount of resources, and the amount of changes which is preferred. In the system which is created during this thesis there is chosen to pull the data, this is mainly because the data sources are owned by external parties which means that pushing the data could not be implemented.

2.5 Can the freshness differ between the data

It is infeasible to request all the data from multiple sources on demand which means that caching is necessary (Ling & Mi, An optimal trade-off between content freshness and refresh cost., 2004; Yates & Kaul, 2012; Huang, Sloan, & Wolfson, 1994). Cached data should be kept fresh enough to make sure that the data can be used. When data is retrieved from multiple sources, there are a few choices which can be made. An important aspect is if all the data from the different sources should have the same freshness or should this differ between the sources (Yates & Kaul, 2012; Huang, Sloan, & Wolfson, 1994). There are some advantages if all the data has the same freshness but these are neglectable when you take the disadvantages into according. The advantages of having different sample rates for different sources is that data is not often refreshed when this isn't necessary. Data from source A might change in a timespan of a day only for one percent while data from source B might change in five minutes more than ten percent. This means that data from source A might still be usable when only refreshed once a day while data from source B is absolutely not usable at a sample rate of once a day. Data from source B should most likely be refreshed more often than once every five minutes. But refreshing data from source A at a rate higher than once every five minutes will probably lead to more than 99% of useless updates because the data changes not that often. Taking this redundancy into account is on a large scale very important to not overload the system.

2.6 How fresh must data be

It depends on the use case how fresh data should be. When looking at the data which is about accommodation availabilities, it's very important that the data is very up-to-date. Customers really don't like it when something appears to be available but actually isn't available when the customer opens it up. But more important is that it can absolutely never happen than multiple customers book the same accommodation in the same period of time (Teuber & Forbrig, 2004; Foris, Crihalmean, & Foris, 2020). The data about an accommodation doesn't have to be as up-to-date as the availability or the prices, it's not that big of a problem when for example the description or the extras are not fully up-to-date (Teuber & Forbrig, 2004; Foris, Crihalmean, & Foris, 2020).

2.7 How to keep accommodation data fresh

To keep accommodation data fresh, there are two methods which are often used, deferred freshness and immediate freshness. Deferred freshness works with delays, it collects data within an on forehand defined interval. Afterwards the data is updated in batches. Immediate freshness doesn't have any delays, it distributes changes to the data as efficiently as possible without any delays. Ideally a web-based platform should use both of these methods. To keep the almost static data of the accommodation like name, maximal number of persons, area, address, etc. fresh, using deferred freshness is sufficient. This data isn't changed very often which means that refreshing it a few times a day is enough. For the more dynamic data of the accommodations like prices and availability it is possible to use deferred freshness, but the data should be refreshed a lot more often. Nevertheless for all the data of an accommodation immediate freshness should be used when the accommodation is specifically looked at in contrast to when it is just shown on a page where multiple accommodations are shown depending on the applied filters (Schrefl, et al., 2003; Labrinidis &

Roussopoulos, Balancing performance and data freshness in web database servers., 2003).

2.8 Pre-generation of web pages

An often used method to speed up the requests of web pages is page generation. This can be done when a user request the page, when this happens the data which has to be shown on the page is retrieved from the database, the page is generated and send to the user. Another method is to pre-generate pages or parts of them, this is done before a user requests the page. This can (just like with the accommodation data) be done periodically or immediately. On top of this it is possible to cache the page in three different locations, client-side, server-side or on a proxy. Finding the most efficient way is difficult challenge, often a hybrid approach is used (Pröll, Starck, Retschitzegger, & Sighart, 1999).

2.9 What is an API

API stands for Application Programming Interface it is a collection of endpoint which make it possible for programs to communicate with each other of with parts of each other. There are different types of API's for example SOAP, REST, gRPC and GraphQL, in this thesis the focus will be on REST. An API which is made on the basis of REST is often called an RESTful API, such API's support four different methods: GET, POST, PUT and DELETE. GET is used to retrieve data from an application they should not change anything they only retrieve data. POST requests are used to add data to an application. PUT requests are used to update data which is already in an application. DELETE requests are used to remove data from an application. It is not impossible, but is highly discouraged to mix such kind of requests and their effects with each other. It is often the case that the responses from an RESTful API are in JSON which stands for JavaScript Object Notation, this name is a bit misleading because it is not only used with JavaScript (Andreo & Bosch, 2019; Fremantle, Kopecký, & Aziz, 2015; Mathijssen, Overeem, & Jansen, 2020).

3 Research Methods

3.1 Research preparations

To be able to do the required research for this thesis, there is a program and testing environment created which can retrieve the data and measure the performance. The program can retrieve data from the five different booking support systems and generalize this data. This data is saved in a database and regularly updated to ensure that the data is fresh. It is easy to extend the amount of data which is stored and increase the amount of camping's from which data is retrieved.

3.2 Coupling the booking support systems

3.2.1 Which API support do the booking support systems provide

After having done research into the five booking support systems the conclusion is that they all have usable API's. The documentation of the API's of Booking Experts, Camping.care, Maxxton and Recranet were all publicly available but the API of Tommy Booking Support isn't. Nevertheless with doing some spitting through some websites of camping's which are using Tommy Booking Support it was clear that there are multiple calls made to their API which meant that they have one. Working with the API's of Booking Experts, Camping.care and Recranet was relatively easy. Because one of the sub-companies of Donselaar Groep works with Booking Expert it was possible to get fully access to the API and retrieve real life data from it. Camping.care provides the possibility to create a free camping for a trail period of a month and during that time work with their system for free, which made it possible to add data to their system and access it through their API. Recranet did not have a free trail, fortunately they have an example organization available which makes it possible to do requests to the API and get data from them. Maxxton has an extensive API documentation available online which made it possible to write the library. Without having the documentation of Tommy Booking Support's API it was possible to write the library. The knowledge gained from working with the other API's and inspecting the requests which were made by the camping's which are working with Tommy Booking Support and some try and error were enough to succeed in writing the library.

3.2.2 Gaining access to data from the booking support systems

To gain access to the data it was necessary to have the required credentials of a camping which uses one of the booking support system. Booking Experts supports a few different ways to authenticate for the usage of their API, in the system which is created is chosen to use an API key which means that a header names "X-API-Key" is added to every request made to their API which has as value an API-key. Camping.care uses bearer tokens to authenticate requests made to their API, this means that to every request a header is added with the name "Authorization", and with the value "Bearer <key>" where key is replaced by an API-key. Recranet doesn't use an API-key only an organization id which is not passed as a header but just as a parameter called organization. Maxxton uses OAuth2 authentication, which means that before it is possible to do requests to their API a token has to be provided. To get this token it is necessary to first authenticate with a client_id and a client_secret, when doing a request to their authenticate end point a response with an access token is given. Afterwards this access token can be used for API requests. Tommy Booking Support uses a simple authentication header which value is the name of the camping together with an API key, so every request contains a header with those values.

3.3 Program description

The program which combines the data from the different booking support system is written in Python, this is something what should be changed later on because of Python's mediocre speed. The main class creates a Camping's object which contains the list of camping's each camping is created on the basis of the data which is stored in the database. These entries contain the name of the camping it's credentials and the sample rate which is used for refreshing the data of that camping. A camping contains a list of accommodations which are present in that camping. Each camping contains one of the five different camping libraries, which library is used depends on the booking support system that the camping uses. To make sure that each library contains all the needed methods, they all implement the same interface. Every time new data is received it's also inserted/updated into the database, this makes it possible to request data with the use of SQL queries. The idea is that when data has to be filtered this can be done on the side of the server and when data has to be ordered this has to be done on the side of the client. An UML-Class-Diagram of the program can be found in Appendix A.

3.4 Important data fields

The different API requests to the booking support systems return different fields of data, not all these fields are necessarily import to Donselaar Groep. After some interviews with different employees, it became clear which fields from the data are important to use and which could be ignored. The results of this study can be found in Appendix B. Not all the fields are used in this thesis, mainly because not all of the five booking support systems provided them. This won't be a problem later on when a platform for searching and booking is created because when a camping owner creates the connection between the platform and their system, they will get the possibility to add the missing fields themselves. When a camping owner decides not to add the missing fields, this will result in less appearances when search filters are used. Also for doing the main research in particular the availability is the most important aspect. It's also an important filter, but it isn't a standard data field for accommodations, which means that retrieving this data goes through a different API endpoint (this holds for all the API's). The integrated schema is shown in Appendix C.

3.5 Field names generalization

To be able to filter on different data fields, there had to be made some decisions about which names are used for the filters and which options to choose from. In the future this will probably change quite a bit, nevertheless the current state of options and names for the filters can be found in Appendix B (for the filter names) and Appendix D (for the filter options where this is applicable). The data in Appendix D was also gathered during the before mentioned interviews, but for the sake of simplicity only the final result is shown in this thesis instead of all the interviewees different takes on it.

3.6 Gathering data
While doing research in which ways it is possible to get a good overview of the availabilities of the accommodations we discovered a pretty lightweight way of getting an overview. First we gather all the id's of each accommodation type. Then we get for each accommodation type the id's of the units which are of this type, for this request we need the id's of the accommodation types. Then we loop through all the reservations and fill in a table in which we can see if an accommodation is available or not. In this way we can search for the scenarios where all the units of an accommodation type are booked at a certain moment in time and when a reservation is canceled which might have an influence on the availability of the corresponding accommodation type. In this way we can use all the data which is available in the system to get a data set as large as possible.

3.7 Research limitations

3.7.1 Retrieving the needed data
We can only use the data which is available from Villatent which uses Booking Experts. This is because we don't have access to data of real camping's which use Maxxton, Recranet or Camping.care. We do have access to the data of two of the campaigns which use Tommy Booking Support, but unfortunately their API is not well enough documented to get the needed amount of data, mainly getting reservations from their system is a problem. Nevertheless from the documentation of Maxxton and Recranet we learned that we can use the same approach as we used with Booking Experts, so looking at the reservations and from there deduce the availabilities. Almost the same approach is possible with Camping.care, the only difference is that their API endpoint which returns reservations need a start and end date. This is not really a problem because it is always possible to use the current date as start date, and as end date we can just use a date which is thousand years in the future to make sure we don't miss any reservations.

3.7.2 Generalizing to multiple camping's and support systems
Because it is only possible for us to retrieve data from Villatent which uses Booking Experts, it would be inappropriate to claim that the results can be generalized to other booking support systems or even to other camping's. To be able to say anything useful about this, more camping's should be used, and studied.

3.8 Research problems
Ideally we also cache the prices of the accommodations, unfortunately this is easier said than done. Some of the booking support systems do have price lists which can be retrieved to check if the prices have been changed but these price lists are very complex and most of them don't have this. The only other way to get the prices is to request them per accommodation type which leads to a lot of requests. These requests get at some point throttled by the booking support systems when you do too much of them. And just periodically checking them is also not airtight because camping owners can at any moment change the prices when they feel like it's necessary to stimulate the bookings. The large amount of requests and the fickleness of camping owners make it very hard to keep this data fresh.

3.9 Research setup

For the research there is only focused on the accommodations of Villatent which works with Booking Experts. First all the ids of their accommodation types are requested. Then per accommodation type id the ids of the corresponding units are requested. Afterwards all the reservations are requested and one by one the availability of the units is changed based on the reservations. After each change the program looks if the change to the availability of a unit had any impact on the availability of the accommodation type itself.

4 Results

4.1 Preparation

There were three scripts created to collect the data about the reservations and availability of the accommodations.

- The first script creates an overview of all the reservations.
- The second script checks if a reservation has any impact on the availability.
- The third script calculates the average amount of time per day that the availability of the accommodations is not fully correct.

4.1.1 Reservation script

The first script which creates an overview of all the reservations does this by requesting all the reservations from the system and for each reservation it writes the relevant fields to a database.

4.1.2 Availability script

The second script checks when the availability changes it accomplishes this by first indexing all the accommodation type id's, then it indexes the accommodation unit id's per accommodation type id. During the indexing of the accommodation unit id's it creates an rentable object for each accommodation unit. This object contains both the id's and an array for the availability where each element is a day and depending on the value 1 or 0 it is booked or not. Then it goes through all the reservations and books and cancels the units based on the reservation and checks every time if that reservation had any influence on the availability. It does this by distinguishing two important situations. The first on is: if the reservation was a booking and on one of the days in the period of the reservation all the units of that accommodation type are full, then the availability changes. The second one is: if the reservation was a cancellation and on one of the days in the period of the reservation there is only one unit left, then the availability changes. If it changed the availability it creates a record of this.

4.1.3 Calculation script

The third script calculates the average amount of time per day that the availability of the accommodations is not fully correct based on different sample rates. It does this by going through the results of the availability script and calculates for each availability change the remaining time to the next sample moment (after which the availability change is represented in the data). It keeps track of the cumulative amount of time per day that accommodation data is not correct based on the different sample rates.

4.2 Data and Discussion

4.2.1 Reservation busyness

In Figure 1 and Figure 2 the reservations and cancellations for respectively 2020 and 2021 are shown. This data is gathered with the help of the reservation script. In Table 1 the total amount of reservations and cancellations for 2020 and 2021 are shown, these numbers are also gathered from the reservation script. It is clear that in different periods of the year the average amount of reservations per month changes a lot (Figure 1, Figure 2). But also the total amount of reservations in 2020 increase with almost 250% in 2021 (Table 1). The large amount of cancellations during the first few months of 2020 can most likely be explained by the start of the Covid-19 pandemic, this can be a logical reason to cancel your reservation, unfortunately this assumption is very hard to verify. In 2020 it looks like that the amount of reservations starts rising around 16-04-2020, reaches its peak around 05-06-2020 to 29-08-2020 and start declining around 13-09-2020 (Figure 1). In 2021 it looks like that the amount of reservations starts rising around 14-02-2021, reaches its peak around 25-04-2021 to 04-07-2021 and starts declining around 16-08-2021 (Figure 2). The difference can also most likely be explained by the start of Covid-19, there were lockdowns in different countries, and people were afraid to get Covid-19. Things like these are most likely the reason for the big difference in the amount of reservations. It is important to point out that in Figure 1 and Figure 2 "confirmed" means that the reservations wasn't cancelled, and "cancelled" means that the reservation was cancelled, so the total amount of reservations that took place are represented by confirmed, you should not subtract cancelled from these reservations because there status just changes.

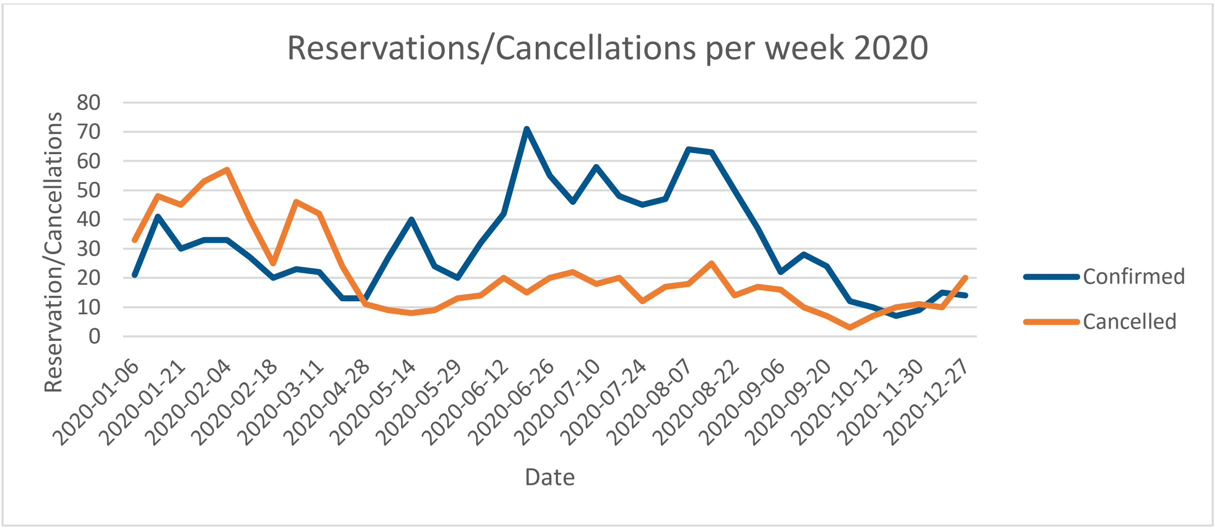


Figure 1: Reservations/Cancellations per week 2020

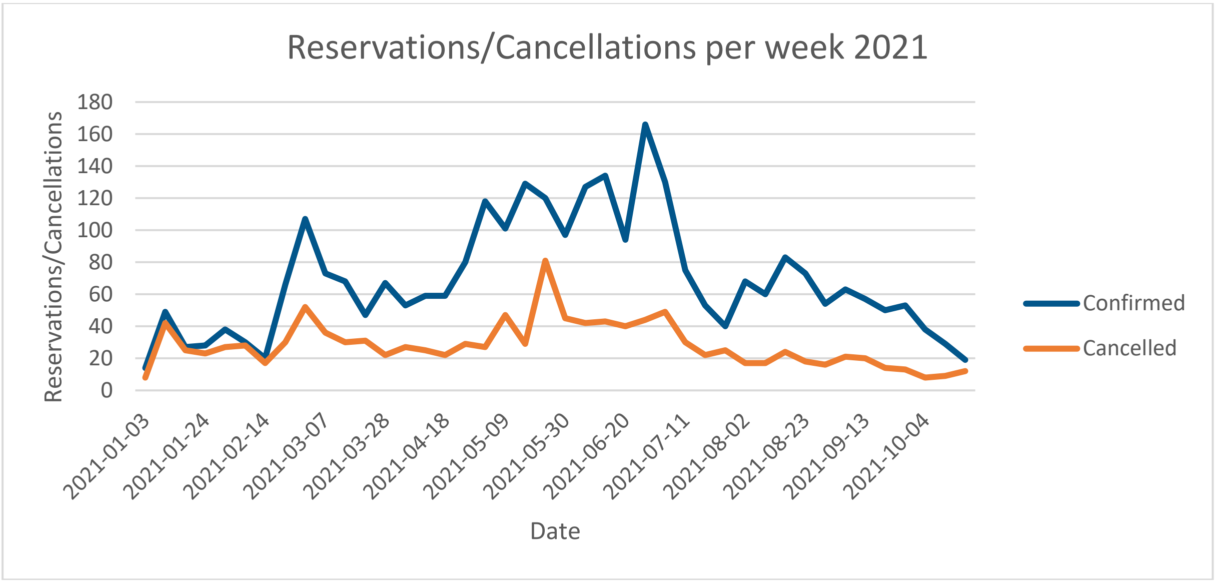


Figure 2: Reservations/Cancellations per week 2021

	Confirmed	Cancelled
2020	1186	789
2021	2916	1187

Table 1: Total amount of confirmed and cancelled reservations for 2020 and 2021

4.2.2 Occupation per day

The data for Figure 3 is also gathered from the reservations script, in the figure is show how many accommodations are occupied in both 2020 and 2021. From the data (Figure 3) it becomes clear that the accommodation units were a lot more occupied in 2021 than in 2020. The peak period is mostly the same, but there are some small peaking periods in 2021 which are not present in 2020. This is most likely due to Covid-19, which prevented people to go to the accommodations and is probably the result of the high amount of cancellations in the few months before in 2020 (Figure 1). In 2021 there are a few more busy periods which are not present in 2020. For example there is a small peak from 02-04-2021 to 04-04-2021 which corresponds with good Friday. In 2020 good Friday was during the lockdown so that is most likely the reason why there were zero accommodations occupied during that period.

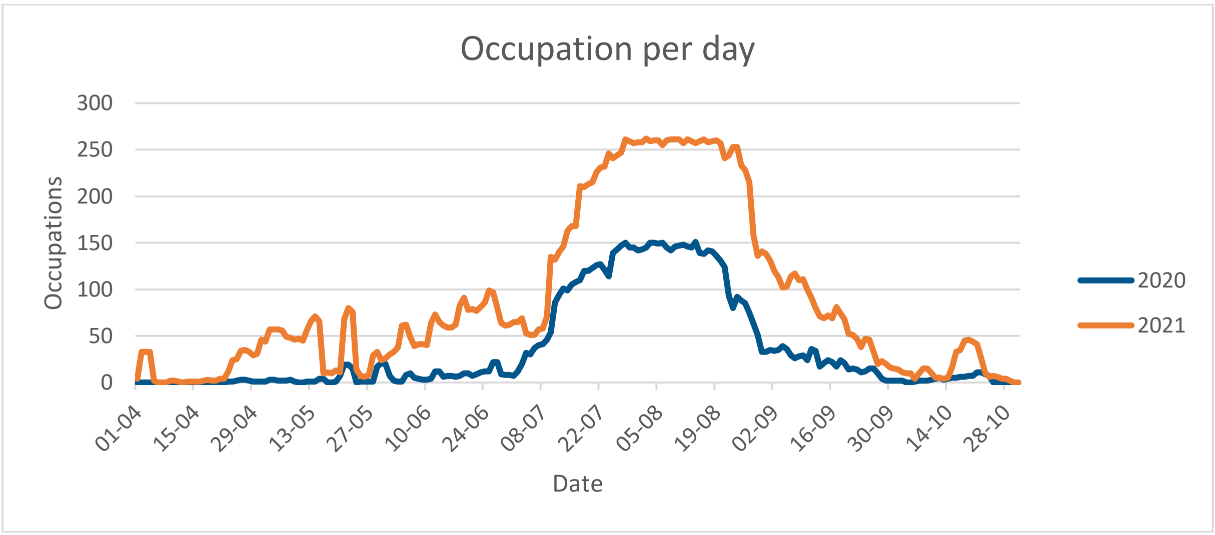


Figure 3: Occupation 2020 and 2021

4.2.3 Irregular reservations and availability changes

The changes in availability from 29-10-2019 to 24-10-2021 are shown in Figure 4, this data is gathered by the availability script. Even when there were 20 reservations in the week from 18-10-2021 until 24-10-2021 (Figure 2), the availability for only two accommodation types has been changed (Figure 4), this is because most of the accommodation types have multiple accommodation units. So it looks like that in a calm period the availability doesn't really change that much, because there are often multiple accommodation units of each accommodation type available. But in busy periods, there are a lot more reservations for example the period from 25-04-2021 until 18-07-2021 (Figure 2) is very busy. When there are for example more reservations made or cancelled, the availability changes also a lot more in that same period.

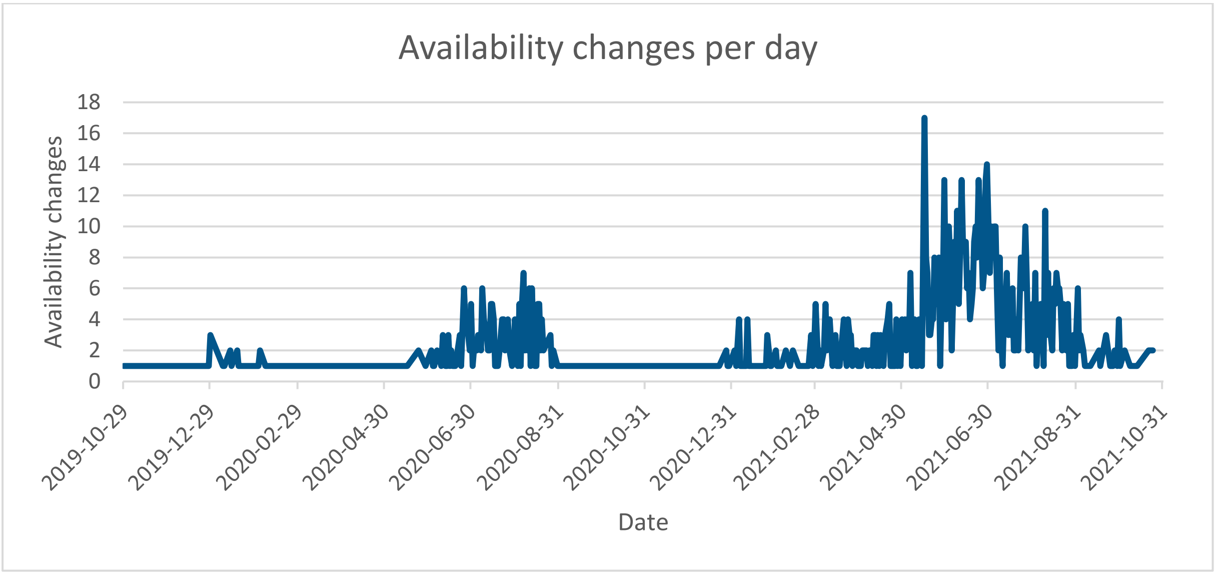


Figure 4: Availability changes per day

4.2.4 Sample rates

In Figure 5 and Figure 6 the cumulative amount of hours a day of wrong availabilities and the average percentage of wrong availabilities are shown respectively. This data is gathered from the calculation script. As pre-estimated the longer the sample rate, the less accurate the availabilities are (Figure 5, Figure 6). An at first very interesting finding are the two dips in the average amount of cumulative hours per day of wrong availabilities at 12 and at 24 hours (Figure 5). After some thinking these two dips are actually very logical and when closely looked at there is also kind of a dip at 6 hours. The reason these dips appear is that when using a sample rate of 6, 12 or 24 there is no shifting window. Which means that every day at the same times the availabilities are refreshed and because people do make reservations in somewhat the same windows of time, this results in a lower average. For example when using a sample rate of 11 hours, the window moves, the first day you refresh at 00:00, 11:00 and 22:00 the next day you refresh on 07:00 and 18:00 the day after that you refresh at 05:00 and 16:00. The higher averages are created by the refreshes which find place at 07:00 and 05:00 because there will be not that much reservation made close to these times so they create a higher average. Those dips in the graph don't occur at 2, 4 and 8 hours while these too don't create a moving window. 2 and 4 hours probably don't have a dip because refreshing at those rates is during the night just too high. 8 hours does probably not create such a dip because during the period 00:00 to 08:00 there are just not enough reservation made. It's important to mention that the average amount of cumulative hours per day of wrong availabilities Figure 5 almost always exceeds the sample rate time, this is because the time per accommodation per day is added together, that why it is called cumulative.

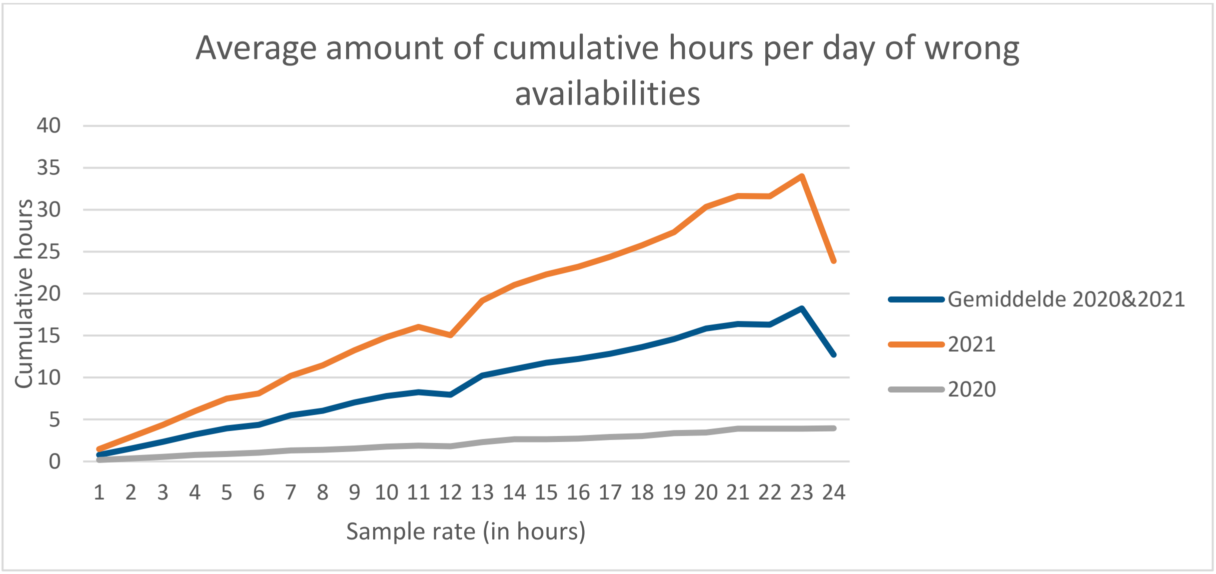


Figure 5: Average amount of cumulative hours per day of wrong availabilities; e.g. 24 cumulative hours might mean: one accommodation wrong all day, or two accommodations wrong both half a day, or four accommodations wrong all for six hours, etc.

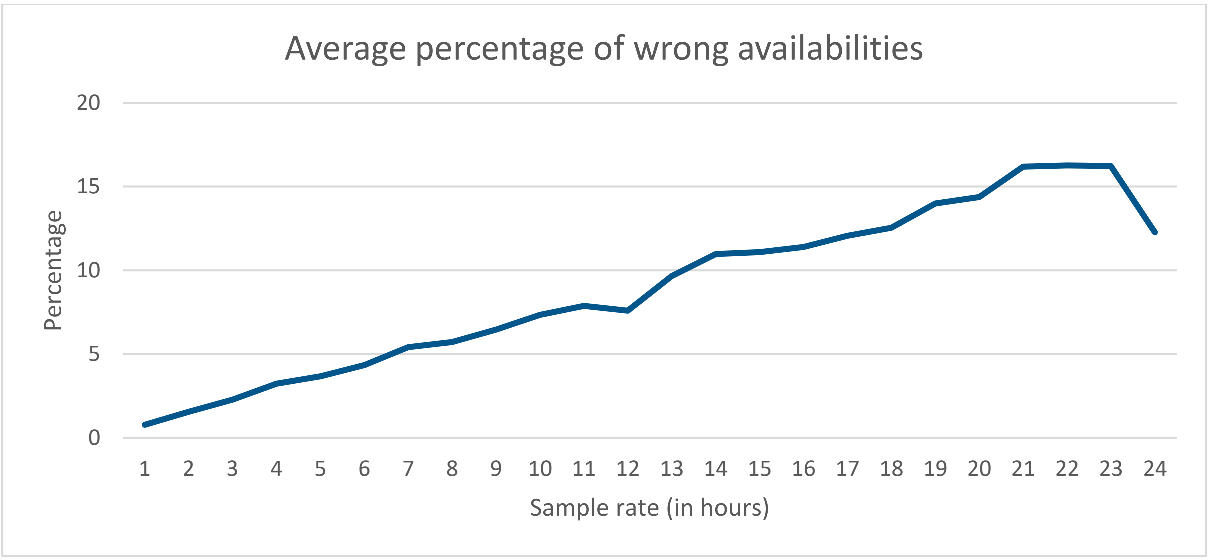


Figure 6: Average percentage of wrong availabilities

4.2.5 Time of the day

In Figure 7 the percentage of the reservations per three-hour time block is shown and in Figure 8 the percentage of the reservations per three-hour time block only including the six time blocks with the most reservations is shown. The data from both Figure 7 and Figure 8 is gathered by taking the reservations data from the reservation script, sorting it on hours and counting the occurrences. From Figure 7 it becomes clear that refreshing the data between 01:00 and 06:59 is not very useful, because only 1% of the total reservations are made between those two times. So during 07:00 and 00:59 the other 99% of reservations are made, this is a time period of 18 hours, during these hours the percentage of the reservations does vary between the 12% and the 23%. A possible refreshing strategy could be to refresh the data at 13:00, 19:00 and 01:00. Refreshing at 13:00 will update all the reservations which are made between 01:00 and 12:59 which is 31%. Refreshing at 19:00 will update all the reservations which are made between 13:00 and 18:59 which is 34%. And refreshing at 01:00 will update all the reservations which are made between 19:00 and 00:59 which is 35%. Those three refresh moments will spread the reservations pretty equally. This all suggests that making the sample rate depended on the time of the day would most likely be a way to make it more efficient.

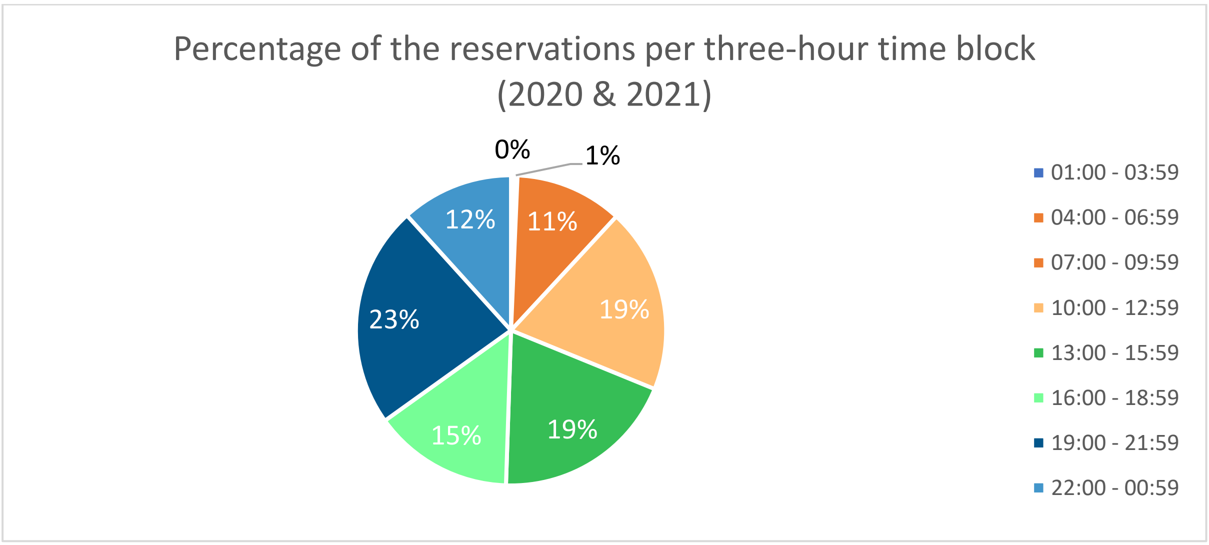


Figure 7: Percentage of the reservation per three-hour time block (2020 & 2021)

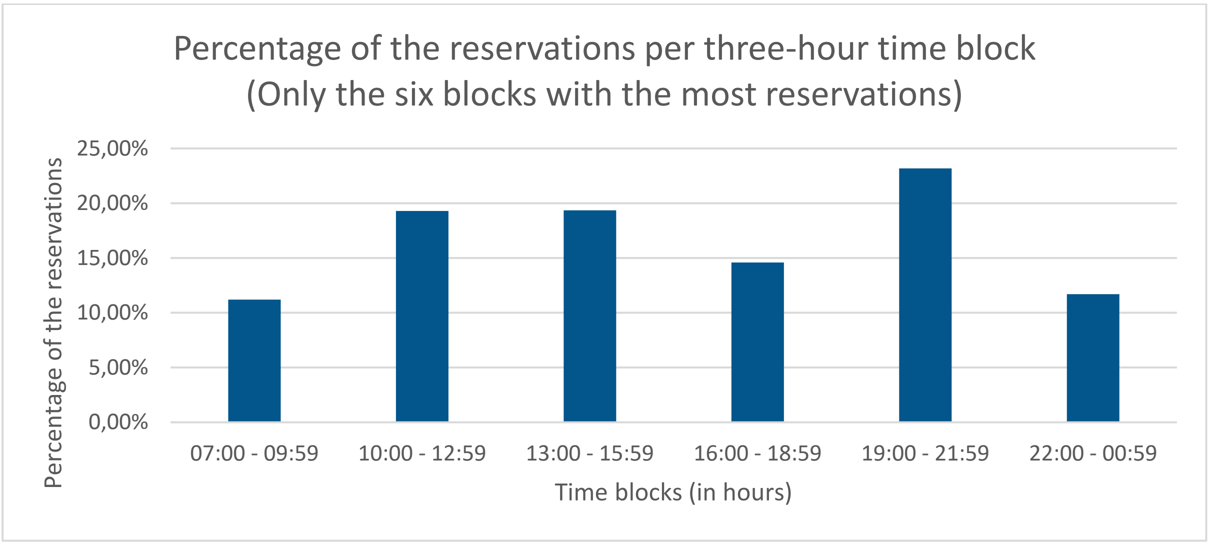


Figure 8: Percentage of the reservations per three-hour time block (Only the six blocks with the most reservations)

4.2.6 Season of the year

In Figure 9 the amount of reservations per three-month period is shown. This data is gathered from the reservation script, the data was sorted on year, month and status (confirmed or cancelled). When looking at Figure 9 it becomes clear that most of the reservations are made during the spring, the second busiest season is the summer, then the winter and finally the fall. It's important to note that both the high amount of cancellations during the winter of 2020 and the overall low amount of reservations during 2020 are most likely a consequence of Covid-19. Nevertheless it is clear that most reservations are made during the spring and the summer. Which means that to get a lower amount of wrong availabilities in those periods there are more refreshes necessary than there are necessary in the fall and winter. This all suggest that making the sample rate depended on the period of the year would most likely make it more efficient. It's important to mention that the data in Figure 9 is until 25-11-2021 and therefore does not contain all the data about November 2021 and doesn't contain any data about December 2021 at all, which means that the amount of reservations in Oct-Dec 2021 will actually be a bit higher than is shown.

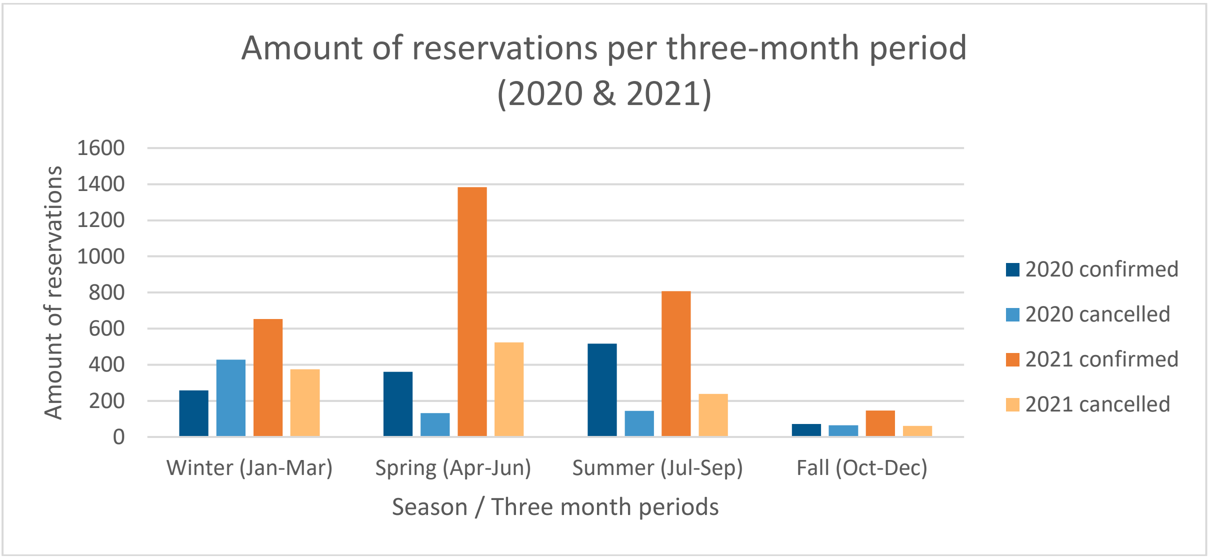


Figure 9: Amount of reservations per three-month period (2020 & 2021)

4.2.7 Uniform or dynamic sample rate

Based on Figure 7, Figure 8 and Figure 9 it becomes clear that it would be more beneficial to use a dynamic sample rate in contrast to a uniform sample rate. With dynamic sample rate we mean a sample rate based on one or multiple variables, in our case both time of the day and period of the year. With uniform sample rate we mean a sample rate which doesn't change at all. In Table 2 the total amount of hours of wrong availabilities are shown in four different situations "Dynamic time" means that the sample rate is dependent on the time of day, "Dynamic period" means that the sample rate is dependent on the period of the year. Using both dynamic time and dynamic period decreases the total amount of hours with wrong availabilities with more than 60%, without doing more refreshes in total. This is because dynamic time only shifts the refresh moments around. And dynamic period has 6 months with 2 refresh moment per day and 6 month with 4 refresh moments per day which averages on 3 refreshes a day. (In Table 3 the settings which are used in the four situations are clarified)

Total hours of wrong availabilities		Dynamic time	
		No	Yes
Dynamic period	No	266680h (100%)	197394h (74,0%)
	Yes	214480h (80,4%)	106097h (39,8%)

Table 2: Total hours of wrong availabilities in 2020 & 2021 (The percentage relative to No-No is shown in parentheses)

Dynamic time	Dynamic period	Clarification
No	No	Refreshing every 8 hours
Yes	No	Refreshing at 13:00, 19:00 and 01:00
No	Yes	During fall and winter refreshing every 12 hours, during spring and summer refreshing every 6 hours
Yes	Yes	During fall and winter refreshing every 12 hours, during spring and summer refreshing at 13:00, 17:00, 21:00 and 01:00

Table 3: Clarification about the four situations; The periods and refresh times are partially based on the data in Figure 7, Figure 8 and Figure 9 and on an on intuition based assumption about what the logic behind the moments of making reservations by customers is.

4.3 Retrospect

At first the created program checked every two minutes the availability of all accommodations, this program ran for almost two weeks. During this period we came to the conclusion that there was a much better way to accomplish the same result but better, more efficient and more resource friendly. The program described in 4.1 didn’t gather the same data as the other program would gather. Nevertheless it used existing data to get a better result in a quicker way. This made it also a lot less resource intensive to run and would also be a lot more suitable for use in a “production environment”. The reasons why we at first checked the availability every two minutes are that, we wanted to check the availability as often as possible because we did not had any knowledge yet about how often we should do this. And if it turned out to be the case that we didn’t do it often enough we had to start the whole data gathering process again from the beginning. Therefore we wanted to do this as often as possible, ideally every minute. Then it appeared to be the case that when we would do the requests every minute we would do almost exactly twice the amount of allowed requests per 15 minutes, so therefore we did it every two minutes, which allowed us to be just under the limit. The reason to change to hours instead of minutes is twofold. Firstly during the aforementioned two weeks the availability changed less than ten times. The results from the two weeks of data gathering and a quick analysis of the new method based on existing reservations quickly led to the conclusion that it would be much better to check the availability on an order of magnitude based on hours instead of minutes. During the two week period the availability changed less than ten times (averaging at less than once every 34 hours) and when looking at all availability changes together during 2020 and 2021 it’s average is less than once every 15 hours. Therefore we started looking into hours instead of minutes during the rest of the research.

5 Conclusion

5.1 Answering the research questions and sub-questions

For clarity purpose the research question and sub questions will be explicitly answered.

How to couple the five big booking support systems?

- The five booking support systems appeared not to be that hard to couple. They all had an extensive API which supported GET requests for retrieving information about the accommodations and the availabilities. When provided with the corresponding API key or OAuth2 credentials (for Maxxton) it was easy to request the data. For each API there was an library created which retrieved the necessary fields from them and renamed them such that they could be generalized in the system.

Which data fields are important and how to integrate them?

- The decisions about which of the data fields had to be retrieved, renamed and used followed from the study into which data fields where important to Donselaar Groep and how they wanted to name them. Afterwards this list was shortened for research purposes, because not all those data fields were interesting to take into account during the research. For example, it would be redundant to track the changes of the country in which an accommodation was located because this would simply not change at all.

Which "sample rate" is optimal to use?

- There is not really a way to give a unambiguously conclusion about the sample rate. It really depends on the time of year and on if it is day or night. A dynamic sample rate would probably be the most efficient and resource friendly solution. This sample rate would be based on the time of day and on which period of year it is, for example fall and winter or spring and summer look like an appropriate way to distinguish them. We did do research into a dynamic sample rate which refreshes two times a day during fall and winter and refreshes four times a day during spring and summer. The refresh moments depend on the time of day during spring and summer, it refreshes at 13:00, 17:00, 21:00 and 01:00. This sample rate decreases the total amount of hours of wrong availabilities with more than 60%.

How to keep data from five big booking support systems fresh?

- To be able to keep the data form the five big booking support systems fresh, it is important to make a clear distinction between two types of data. Static data and dynamic data. With static data is referred to data which doesn't change much or at all, data like the location of an accommodation, or the surface area for example. Dynamic data is data which changes a lot, data like availability of an accommodation, or price per night for example. For the static data a uniform sample rate of once a day would be more than sufficient. For the dynamic data a dynamic sample rate would be the most efficient choice, this sample rate should be based on the time of day and the period of the year. The dynamic sample rate which is used in our research decreases the total amount of hours of wrong availability data with more than 60%.

5.2 Discussion about & recommendations for follow-up research

-It could be very likely the case that other camping's / channels would need another type of sample rate, therefore it would be interesting to do more research into their availability changes. It can be expected that a camping which is very popular in the winter period has it's reservations peaks at another period of the year than a camping which is very popular in the summer.
-It could also be interesting to do more research into the time of day where reservations are made. The examined channel Villatent has almost all its accommodations located in Western-Europe, which increases chances of West-European citizens booking them. When taking a look at the time zone's it becomes clear that most of them live in only two time zone's UTC+0 and UTC+1. If the accommodations of Villatent where more spread around the world this would most likely increase the amount of people outside of Western-Europe to book the accommodations which could lead to real changes in the amount of reservations when looking at the time of day.
-For follow-up research it is recommended to take a deeper look into the moments when the data is refreshed, it is now partially based on the data in Figure 7 and Figure 8, and on an on intuition based assumption about what the logic behind the moments of making reservations by customers is. It might be the case that it's better to not divide the sample rate during the day uniformly.
-Another interesting point to look into is how the data is currently integrated and if there are better and more efficient ways to do this.
-For follow-up research it's also recommended to approach the problem of finding the "optimal sample rate" as a kind of train-test problem. This can be done by finding the optimal refresh moments and sample rate with the first half of the data, and then using the second half of the data to test if the found moments and rates are indeed effective. The reason why this isn't done in this thesis is because the first half of the data (data from 2020) is highly clouded by the effects of the Covid-19 pandemic, while the second half of the data (data from 2021) is a lot less affected by this. So this won't give us a clear picture of the effectiveness of approaching the problem as a train-test problem.

5.3 Limitations

During the research of this thesis it was concluded that in practice it won't be feasible to cache the prices. There are multiple reasons which make caching these infeasible. This because there are no specific endpoints in any of the API's to retrieve the prices in a usable way. Furthermore the only way to check if the prices are changed is to just request the reservation costs of a specific accommodation in a specific period. Doing this for all the accommodations in all the possible periods is just not feasible. On top of this it's also possible for camping owners to change the prices of their accommodations whenever they like which makes it also impossible to predict when the prices might change.
It also became clear that we could only use the data of Villatent because other camping's / channels did not want to provide access to their data on the ground of possible privacy issues and not wanting to spend any effort on it without being able to get something out of it. This was disappointing but did not come as a surprise because from their perspective there was indeed nothing to profit from, it would only took them time and would be a risk for them. Another limitation is that all the data is gathered during the Covid-19 pandemic which might be not indicative of data in times outside of a pandemic. It's not possible

to do any research into this limitation in this thesis, because this thesis is written during the Covid-19 pandemic.

6 Bibliography

Andreo, S., & Bosch, J. (2019). API management challenges in ecosystems. *In International Conference on Software Business* (pp. 86-93). Springer, Cham. Retrieved from https://link.springer.com/chapter/10.1007/978-3-030-33742-1_8

Baeza-Yates, R., Gionis, A., Junqueira, F. P., Murdock, V., Plachouras, V., & Silvestri, F. (2008). Design trade-offs for search engine caching. *ACM Transactions on the Web (TWEB), 2(4)*, 1-28. doi:https://doi-org.ru.idm.oclc.org/10.1145/1409220.1409223

Barish, G., & Obraczke, K. (2000). World wide web caching: Trends and techniques. *IEEE Communications magazine, 38(5)*, 178-184. doi:https://doi.org/10.1109/35.841844

Bedewy, A. M., Sun, Y., & Shroff, N. B. (2016). Optimizing data freshness, throughput, and delay in multi-server information-update systems. *In 2016 IEEE International Symposium on Information Theory (ISIT)*, (pp. 2569-2573). doi:https://doi.org/10.1109/ISIT.2016.7541763

Bouzeghoub, M. (2004). A framework for analysis of data freshness. *In Proceedings of the 2004 international workshop on Information quality in information systems*, (pp. 59-67). doi:https://doi-org.ru.idm.oclc.org/10.1145/1012453.1012464

Bright, L., & Raschid, L. (2002). Using latency-recency profiles for data delivery on the web. *In VLDB'02: Proceedings of the 28th International Conference on Very Large Databases* (pp. 550-561). Morgan Kaufmann. doi:https://doi-org.ru.idm.oclc.org/10.1016/B978-155860869-6/50055-X

Bright, L., Gal, A., & Raschid, L. (2004). Adaptive pull-based data freshness policies for diverse update patterns. Retrieved from https://drum.lib.umd.edu/bitstream/handle/1903/1334/CS-TR-4554.pdf?sequence=1&isAllowed=y

Cao, P. Z., & Beach, K. (373-388). Active cache: Caching dynamic contents on the web. *In Middleware'98*, 1998. doi:https://doi.org/10.1007/978-1-4471-1283-9_23

Ceri, S., Fraternali, P., & Paraboschi, S. (1999). Design principles for data-intensive web sites. *ACM Sigmod Record, 28(1)*, 84-89. doi:https://dl-acm-org.ru.idm.oclc.org/doi/pdf/10.1145/309844.310064

Cherkasova, L. (1998). Improving WWW proxies performance with greedy-dual-size-frequency caching policy. *Hewlett-Packard Laboratories*, 297-306. Retrieved from https://www.hpl.hp.com/techreports/98/HPL-98-69R1.pdf?jumpid=reg_R1002_USEN

Cho, J., & Garcia-Molina, H. (2000). Synchronizing a database to improve freshness. *ACM sigmod record, 29(2)*, 117-128. doi:https://doi-org.ru.idm.oclc.org/10.1145/335191.335391

Dan, A., & Sitaram, D. (1996). Generalized interval caching policy for mixed interactive and long video workloads. In Multimedia Computing and Networking 1996. *International Society for Optics and Photonics*, Vol. 2667, pp. 344-351. doi:https://doi.org/10.1117/12.235887

Dan, A., & Towsley, D. (1990). An approximate analysis of the LRU and FIFO buffer replacement schemes. *In Proceedings of the 1990 ACM SIGMETRICS conference on Measurement and modeling of computer systems*, (pp. 143-152). doi:https://doi-org.ru.idm.oclc.org/10.1145/98457.98525

Datta, A., Dutta, K., Thomas, H. M., VanderMeer, D. E., Ramamritham, K., & Fishman, D. (2001). A comparative study of alternative middle tier caching solutions to support dynamic web content acceleration. *In VLDB*, 667-670. Retrieved from http://www.vldb.org/conf/2001/P667.pdf

Ferragut, A., Rodriguez, I., & Paganini, F. (2018). Optimal timer-based caching policies for general arrival processes. *Queueing Systems, 88(3)*, 207-241. doi:https://doi.org/10.1007/s11134-017-9540-3

Foris, D., Crihalmean, N., & Foris, T. (2020). Exploring the environmental practices in hospitality through booking websites and online tourist reviews. *Sustainability, 12(24)*, (p. 10282). doi:https://doi.org/10.3390/su122410282

Fraternali, P., & Paolini, P. (1998). A conceptual model and a tool environment for developing more scalable, dynamic, and customizable web applications. *In International Conference on Extending Database Technology* (pp. 419-435). Berlin, Heidelberg: Springer. doi:https://doi.org/10.1007/BFb0101000

Fremantle, P., Kopecký, J., & Aziz, B. (2015). Web api management meets the internet of things. *In European Semantic Web Conference* (pp. 367-375). Springer, Cham. Retrieved from https://link.springer.com/chapter/10.1007/978-3-319-25639-9_49

Gertz, M., Özsu, M. T., Saake, G., & Sattler, K. U. (2004). Report on the dagstuhl seminar. *ACM SIGMOD Record, 33(1)*, 127-132. Retrieved from https://sigmodrecord.org/publications/sigmodRecord/0403/R1.report4-tamer.pdf

Gramacy, R. B., Warmuth, M. K., Brandt, S. A., & Ari, I. (2002). Adaptive caching by refetching. *In NIPS*, 1465-1472. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.14.5805&rep=rep1&type=pdf

Holmedahl, V., Smith, B., & Yang, T. (1998). Cooperative caching of dynamic content on a distributed web server. *In Proceedings. The Seventh International Symposium on High Performance Distributed Computing (Cat. No. 98TB100244)* (pp. 243-250). IEEE. doi:https://doi.org/10.1109/HPDC.1998.709978

Huang, Y., Sloan, R. H., & Wolfson, O. (1994). Divergence caching in client-server architectures. *In Proceedings of 3rd International Conference on Parallel and Distributed Information Systems* (pp. 131-139). IEEE. doi:https://doi.org/10.1109/PDIS.1994.331723

Jarke, M., Jeusfeld, M. A., Quix, C., & Vassiliadis, P. (1999). Architecture and quality in data warehouses: An extended repository approach. *Information Systems, 24(3)*, 299-253. doi:https://doi-org.ru.idm.oclc.org/10.1016/S0306-4379(99)00017-4

Labrinidis, A., & Roussopoulos, N. (2001). Update propagation strategies for improving the quality of data on the web. Retrieved from http://hdl.handle.net/1903/6236

Labrinidis, A., & Roussopoulos, N. (2003). Balancing performance and data freshness in web database servers. *In Proceedings 2003 VLDB Conference* (pp. 393-404). Morgan Kaufmann. doi:https://doi-org.ru.idm.oclc.org/10.1016/B978-012722442-8/50042-2

Labrinidis, A., & Roussopoulos, N. (2004). Exploring the tradeoff between performance and data freshness in database-driven web servers. *The VLDB Journal, 13(3)*, 240-255. doi:https://doi.org/10.1007/s00778-004-0131-7

Levy-Abegnoli, E., Iyengar, A., Song, J., & Dias, D. (1999). Design and performance of a Web server accelerator. *In IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies* (pp. 135-143). IEEE. doi:https://doi.org/10.1109/INFCOM.1999.749261

Ling, Y., & Chen, W. (2004). Measuring cache freshness by additive age. *ACM SIGOPS Operating Systems Review, 38(3)*, 12-17. doi:https://dl-acm-org.ru.idm.oclc.org/doi/pdf/10.1145/1035834.1035836

Ling, Y., & Mi, J. (2004). An optimal trade-off between content freshness and refresh cost. *Journal of applied probability, 41(3)*, 721-734. doi:https://doi.org/10.1239/jap/1091543421

Mathijssen, M., Overeem, M., & Jansen, S. (2020). Identification of practices and capabilities in API management: a systematic literature review. *arXiv preprint arXiv:2006.10481.* Retrieved from https://arxiv.org/abs/2006.10481

Naumann, F., Leser, U., & Freytag, J. C. (2005). Quality-driven integration of heterogeneous information systems. *systems. Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät II, Institut für Informatik.* Retrieved from https://edoc.hu-berlin.de/bitstream/handle/18452/3091/117.pdf?sequence=1

Pappas, N., Gunnarsson, J., Kratz, L., Kountouris, M., & Angelakis, V. (2015). Age of information of multiple sources with queue management. *In 2015 IEEE international conference on communications (ICC)* (pp. 5935-5940). IEEE. doi:https://doi.org/10.1109/ICC.2015.7249268

Pröll, B., Starck, H., Retschitzegger, W., & Sighart, H. (1999). Ready for prime time: pre-generation of web pages in TIScover. *In Proceedings of the eighth international conference on Information and knowledge management*, (pp. 63-68). doi:https://doi-org.ru.idm.oclc.org/10.1145/319950.319958

Rao, F. Y., Fang, R., Tian, Z., Lane, E., Srinivasan, H., Banks, T., & He, L. (2006). Cache mediation pattern. *roceedings of EuroPloP'06*. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.107.2273&rep=rep1&type=pdf

Röhm, U., Böhm, K., Schek, H. J., & Schuldt, H. (2002). FAS—a freshness-sensitive coordination middleware for a cluster of OLAP components. *In VLDB'02: Proceedings of the 28th International Conference on Very Large Databases* (pp. 754-765). Morgan Kaufmann. doi:https://doi-org.ru.idm.oclc.org/10.1016/B978-155860869-6/50072-X

Schrefl, M., Bernauer, M., Kapsammer, E., Pröll, B., Retschitzegger, W., & Thalhammer, T. (2003). Self-maintaining web pages. *Information Systems, 28(8)*, 1005-1036. doi:https://doi.org/10.1016/S0306-4379(03)00004-8

Segev, A., & Fang, W. (1989). Currency-based updates to distributed materialized views. Retrieved from https://escholarship.org/content/qt5zz8n5r3/qt5zz8n5r3.pdf

Sun, Y., & Cyr, B. (2019). Sampling for data freshness optimization: Non-linear age functions. *Journal of Communications and Networks*, 204-219. doi:https://doi.org/10.1109/JCN.2019.000035

Sun, Y., Uysal-Biyikoglu, E., Yates, R. D., Koksal, C. E., & Shroff, N. B. (2017). Update or wait: How to keep your data fresh. *IEEE Transactions on Information Theory, 63(11)*, 7492-7508. doi:https://doi.org/10.1109/TIT.2017.2735804

Teuber, C., & Forbrig, P. (2004). Different types of patterns for online-booking systems. *In Proceedings of the 3rd annual conference on Task models and diagrams*, (pp. 91-97). doi:https://doi.org/10.1145/1045446.1045464

Wang, J. (1999). A survey of web caching schemes for the internet. *ACM SIGCOMM Computer Communication Review, 29(5)*, 36-46. doi:https://doi-org.ru.idm.oclc.org/10.1145/505696.505701

Wang, R. Y., & Strong, D. M. (1996). Beyond accuracy: What data quality means to data consumers. *Journal of management information systems, 12(4)*, 5-33. doi:https://doi.org/10.1080/07421222.1996.11518099

Wolman, A., Voelker, M., Sharma, N., Cardwell, N., Karlin, A., & Levy, H. M. (1999). On the scale and performance of cooperative web proxy caching. *In Proceedings of the seventeenth ACM symposium on Operating systems principles*, 16-31. doi:https://doi-org.ru.idm.oclc.org/10.1145/319151.319153

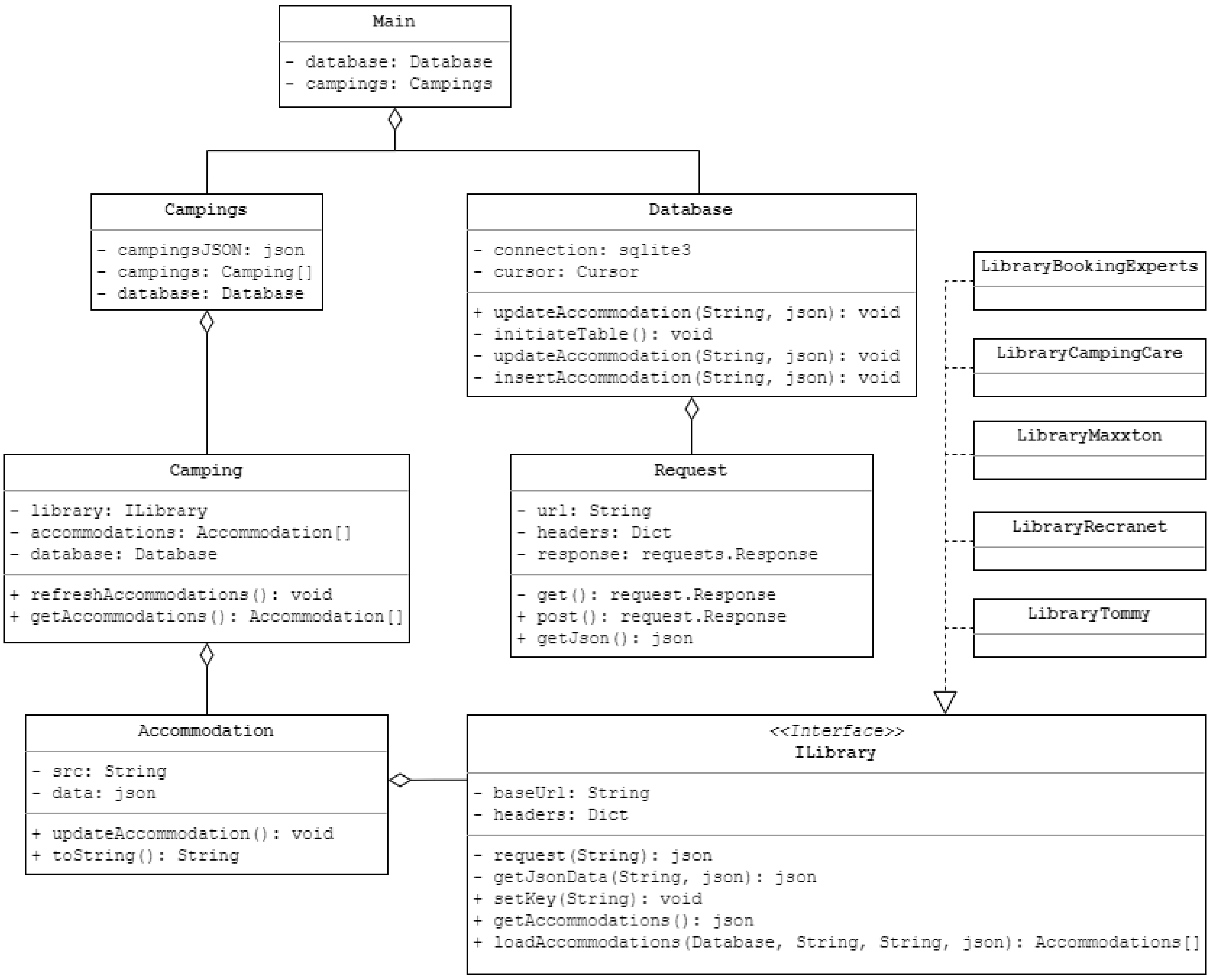
Xiong, M., Han, S., Lam, K. Y., & Chen, D. (2008). Deferrable scheduling for maintaining real-time data freshness: Algorithms, analysis, and results. *IEEE Transactions on Computers, 57(7)*, 952-964. doi:https://doi.org/10.1109/TC.2008.16

Yates, R. D., & Kaul, S. (2012). Real-time status updating: Multiple sources. (pp. 2666-2670). IEEE. doi:https://doi.org/10.1109/ISIT.2012.6284003

Zhu, H., & Yang, T. (2001). Class-based cache management for dynamic web content. *In Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)* (pp. 1215-1224). IEEE. doi:https://doi.org/10.1109/INFCOM.2001.916615

Appendix A

UML-Class-Diagram of program



Appendix B

The table below represents the result of the interviews with four different employees. Their real names are replaces with E1 up to E4 which stands for Employee 1 up to Employee 4. There is also a column named Res which stands for Result, this represents the chosen data fields which are going to be used in our research.

Data fields	E1	E2	E3	E4	Res
County	■	■	■	■	■
Region			■	■	■
Amount of persons	■	■	■	■	■
Amount of bedrooms		■	■		
Amount of bathrooms			■		
Accommodation type	■	■	■	■	■
Target Audience		■		■	■
Locality		■		■	■
Suitable for	■	■	■		■
Accommodation facilities	■	■	■	■	■
Park facilities	■	■	■	■	■
Swimming pool	■	■	■	■	■
Sport games	■	■		■	■
Park size	■			■	■
Animation		■	■	■	■
Activities environment	■	■		■	■
Ratings	■	■	■	■	■
Price	■	■	■	■	■
Glamping Vibes			■	■	■
Distance to current location	■			■	■
Searching on map	■		■	■	■

Appendix C

The table below is the integrated schema which is used. The first five columns contain the names of the data in the corresponding booking support system. The last column named Result shows the names of the data in our system.

Booking Experts	Camping.care	Maxxton	Recranet	Tommy Booking Support	Result
categories.id	accommodations.id	accommodationtypes.accotypeKindId	accommodations.id	accommodatie.id	exid
categories.attributes.name	accommodations.name	accommodationtypes.accommodationTypeKindCode	accommodations.title	accommodatie.naam	name
-	accommodations.persons_min	-	accommodations.minNumOfPersons	accommodatie.minPersonen	minPersons
categories.attributes.max_number_of_people	accommodations.persons_max	accommodationtypes.numberOfPersons	accommodations.maxNumOfPersons	accommodatie.maxPersonen	maxPersons
categories.attributes.description	accommodations.description	?	accommodations.description	accommodatie.omschrijving	description
categories.attributes.country_code	park.country	accommodationtypes.address.country.code	accommodations.country	accommodatie.land	Country
categories.attributes.city + categories.attributes.address	park.city + park.address + park.address_number	accommodationtypes.address.city + accommodationtypes.address.address1 + accommodationtypes.address.houseNumber	accommodations.locality + accommodations.address	accommodatie.plaats + accommodatie.straat + accommodatie.huisnummer	address
categories.attributes.postal_code	park.zipcode	accommodationtypes.address.zipCode	accommodations.postalCode	Accommodatie.postcode	zipcode

Appendix D

In the enumeration below the different filter options per filter are listed, these are also based on the interview mentioned in Appendix B. For clarity reasons these filter options are shown in their own appendix.

Accommodation type
  Safari tent, Yurt, Treehouse, Air lodge, Tiny house

Accommodation facilities
  Wheelchair accessible, Dishwasher, Hot tub, Garden / Fenced terrace

Park facilities
  Shop, Restaurant, Snack bar, Fresh sandwiches, Take-away meals, Bar, Disco, Etc.

Park size
  < 5 spots, 5-9 spots, 10-49 spots, 50-99 spots, 100-199 spots, 200-299 spots, 299+ spots

Activities environment
  Golf course, Skiing, Etc.

Swimming pool
  Indoor pool, Outdoor pool, Swimming paradise, Paddling pool, Subtropical swimming paradise

Sport games
  Tennis, Jeu de boules

Locality
  At the beach, Close to a city, Close to a forest, Mountain area, Etc.

Suitable for
  Disable persons, Wheelchair accessible, Pet friendly, Groups

Target audience
  Teenagers, Elderly, Romantic, Remote, Peace seekers

Glamping vibes
  Separate bedrooms for children, Barbecue, Design interior, Eco-friendly, Hammock, High speed internet, Hot tub / Whirlpool, Wood stove, Small-scale park, Luxury box spring, Luxury coffee machine, Breakfast service, View, Lots of privacy, Fireplace