Big Data Management

Project 2

Louisa Schermuly
Coralie Boucheron
Anssi Rasmus
Thomas Svensson
Mircea Murăşan
Peter Tomascik

Introduction

After discussing what influence data could have on our daily lives from a user perspective, we are moving forward with our big data journey by jumping into a more concrete way of exploiting data from a corporate point of view. Today, most data is generated by users or machines. This data is not only bulky and has a high velocity, but it is also diverse, which translates the term "Big Data". Big Data imposes new challenges that shake up the traditional way of processing data, using relational databases for example, because they are not designed for the Web scale that we mentioned before. A new data management movement has emerged, and the need for quick and efficient data processing gave birth to a new range of solution such as tools of the Hadoop framework. In order to experience those challenges as well as to realize the possibilities that Big Data offers, we were given a datasource from ITU WiFi infrastructure and were asked to put ourselves in the shoes of a consultancy team in charge of developing new services from this dataset.

To answer to this demand, we will first analyse the data and what it tells us to see what services we could get out of it. After giving an overview of our technical implementation, we will present the three different batch views we created as well as the services they provide. Finally, we will reflect on the challenges and ethical problems we can encounter during the development of such a system.

Data-driven decisionmaking

In this section we will present some theoretical background for the work we did in this report. To start with, we received the dataset from ITU, while it had already been through a process of interpretations and structuring. ITU has collected the data, made decisions about how and where to store and what parts of the data they are willing to share with us for our project. As Gitelman and Jackson (2013, p. 3) argue, data does not just exist, it has to be generated and the imagination of data entails an interpretive base. The process behind collecting and storing is not independent from interpretation: we need to classify and categorise the examined object in order to turn it into digital data. As was presented in the lecture (Week 3, 12.10.2017) all data needs to be cleaned prior analysis and re-arranged and structured in a useful way. When working with data, we do not just project the reality and turn it into digital data, we make choices that will lead into partial and limited view about the object. Thus, when we start to work with the data for our interests – by including and excluding data – we will make our interpretations of it. In that sense, the interpretations have a cumulative nature, since we make our interpretations about the data received from ITU and our interpretations are been built upon theirs. This refers to data provenance, that is the historical inputs, entities and systems that affect the way data is seen. Janitorial work requires assessing the validity of the data and the numbers representing the object: are the data representing the people in ITU connected to WiFi in a truthful way, or have the interpretations, classifying and categorising threatening the validity? We need to ask is the data representing what it is suppose to represent and is it relevant for our project.

In this project we are utilizing data from the ITU-routers to construct some services. This can be understood as a decisionmaking process in order to simplify and find a way to utilize, or frame, the data so that it adds some value when being utilized in a service. Cleaning involves excluding or including data, thereby curating and shaping it to be used for a specific purpose in a service. In a sense, we have been constructing a schema for what data to use, and what not to use. This relates to what Dourish (2017) says about the design of relational databases; the structure of the schema largely defines and constrains the conditions of viewing, editing and contributing. Meaning that the choices we do in our design of the services will create a certain view categorizing or classifying something as better or worse than something else. In order to achieve this we make up stories of this data, based on how we understand it, and from this frame it into a service. During such a process, it is important to be aware of the different decisions that are being made. This quote from Dourish understates this:
"The materialities of database technologies shape the kinds of "databasing" that can be done and imagined. Databases, then, have a remarkable power that reaches in two directions, inward toward architectures and outward toward society, systems and services" (Dourish 2017, p. 135)
The power in designing the architecture goes back to us, and the decisions we make when deciding what data to utilize and in what context, meaning we see some data more valuable than others. These decisions then affect how the service can be used, or how it is used, and how it adds value in a certain context, be it a society, system or service. Being aware of the power one has when designing a service is important, as it calls for more accountability in the

process, which might (or not) limit the possible negative implications such a system can have in the future.

Diving into data

As we point out in the previous section, making decisions is an inherent part of working with data. Therefore, in this part of the report, we will get to know the data more, to see what we can get out of it, and how we can use it.

By examining the data, we can see how many clients are connected, the quality of the connection, list of different devices connected, uptime, access points and device identifier. With the data, we can see how many clients are been connected and what is their connection quality in different parts of ITU. Moreover, the data provides us information about the device itself – is it mobile or laptop, Mac or something else – and the time period it was connected to the ITU WiFi, with an accuracy of one minute. The information provided by the data enables a wide range of possible services, which we will present later.

However, the data does not display the actual purpose of using the WiFi nor the actual traffic – is it been used for studying, working or browsing for fun. Therefore, looking the data does not show us what websites were visited. In addition, the data does not show the IP addresses or are the clients actively using WiFi or are they passively connected. The ITU WiFi connects automatically to your device if logged in before and if the WiFi connection settings are been set as automatic.

In addition, the data does not show the distance between the client and the access point, hence, the physical location of the client. It is not certain that the physical location and the access point are always in the same floor or classroom, since the client will be connected to the access point based on the quality of the connection, not the physical location. Therefore, a client in 4. floor may be connected to the access point in the 5. floor, if the connection quality is better there for some reason.

The service where we can see how many people are connected within a classroom can indicate if a course is interesting, or if the group could be fitted in a bigger/smaller classroom. The data can also be used from another perspective, given that the students are connected, one could use the data to design a service to use as a more efficient ´headcount´ in class. Giving the teacher more insights into how many people are attending this class, compared to the number of enrolled students. However, this does not give a clear picture of the reality. It might be that students are not using their computers, as the lecture is very interesting, and their focus are towards the teacher. Looking at this data, to see if a class is good or bad, might be challenging, as many people might be connected to the WiFi, but it says nothing about the quality of the class. It might be an uninspiring class, but the WiFi data tells us nothing about this. This serves as a good example, showcasing how we might have correlation, but it does not necessarily give us any insights into what actually happens. The data might not provide any deep understanding of what is going on in the university. Say that you do not know this university, but you get your hands on this data. Will you then be able to say something about the individuals? You can say something, but again, it might not provide any deep understanding about the individuals at this university. In the next paragraph, we will look into the things we can say something about with this data.

There are some stories to be told about the data, however not very detailed ones. Data can show what is the time period when the majority of students arrive and leave ITU, when they go for lunch or other breaks. This can be seen by looking when they are actively using their laptops, hence, connected to the ITU WiFi. After collecting data we could probably come up with patterns that tell us stories, such as majority of students come to ITU between 9-11 in the morning, have their lunch break between 12-14 and leave the premises between 16-17. In order to create these stories and to understand what the data say to us about the activity, you need to understand the context where the data is been collected from. Big Data always requires context to be understood. For us to understand what the data means, we need to have local competence about the processes, timetables and habits within ITU. By having the domain knowledge as ITU students, we know that Auditorium 1 and 2 are the biggest lecture rooms where the mass lectures are been held. Local competence and domain knowledge help us to create data patterns and to account for deviations. The university fall break would be a deviation in the data, since many students do not show up to lectures or group work during that time, and with our local competence we could explain the deviation. However, as we have seen with this section, the data might give different insights, depending on how you chose to frame it.

Implementation of the system

Now, we will get to know the technical side of the project and present the developed batch views that our services are been based on. In order to exploit big data sets, we need a method that solves complex problems mixing real time and batches. The Lambda architecture is a framework perfectly adapted for this type of problematics. It makes it possible to store and process large volumes of data (batch), but also provide low-latency real time data by adding a speed layer. As a first step of our project, we focused on which layers of the Lambda Architecture to use. First, the system requires a batch layer, which is responsible for storing the constantly growing master dataset (also known as storing layer). This layer holds all the massive computing functions on the data responsible for creation of the defined batch views. We will also need a service layer, which sits on top of the batch layer and provides the batch views for incoming user queries. As we only get new data once a day, there is no need for implementing a Speed layer. This would only be needed if the interval between new incoming data would be significantly smaller than the processing time of the batch computation.

Now that we have a precise idea of which layers to use, let’s focus on the best way to store the data. Considering the size of the dataset, we need to take under consideration problems that we could encounter with data storage and analysis. The storage capacity of the hard disks is getting better, but along with that, the processing time is also increasing. It then becomes necessary to parallelize the processing by storing on several hard disk drives. However, this inevitably raises the problem of reliability of the hard drives that causes the hardware failure. The solution envisaged is the duplication of data using Hadoop Distributed Filesystem (HDFS). HDFS chunks all the files into blocks, replicates and then disperses it between multiple nodes, which makes it easier to perform parallel computation. This method fits all the requirements we need to store our data and in the same way it stays simple as it is not providing useless features (e.g. random reads).

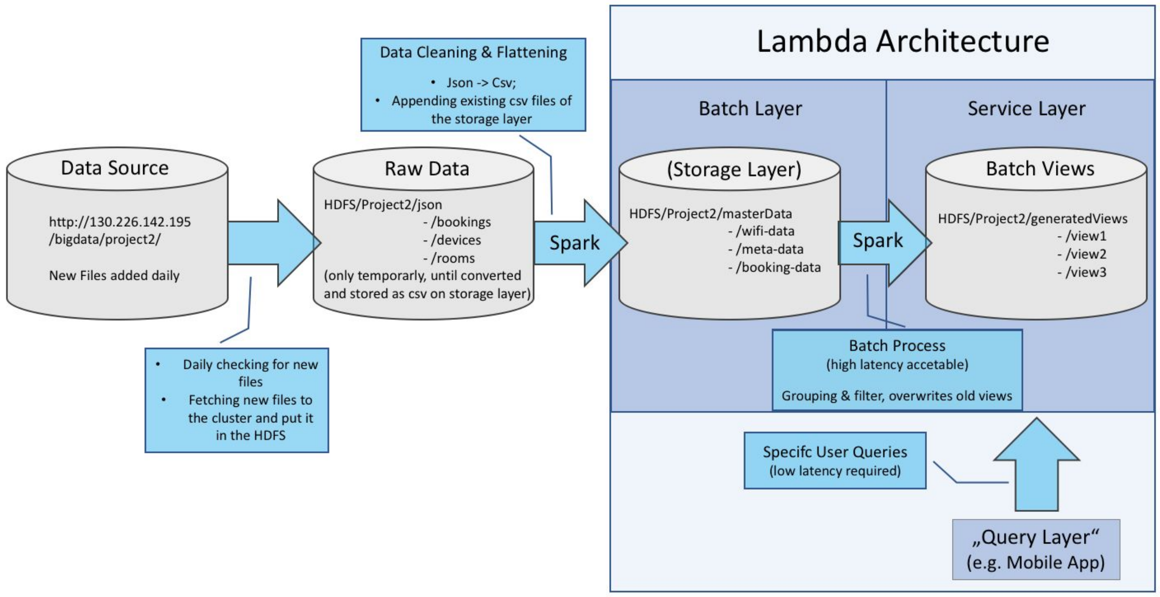
We based our choice on the requirements from the book “Big Data : principles and best practices of scalable realtime data systems” by Nathan Marz and James Warren (2015), which states, that new data should be appended efficiently and the storage should be scalable. As we receive new wifi data every day, the master data set grows very fast, and the system should handle this addition of new data without losing performance. HDFS allows increase of storage space by adding new machines. The storage should also have support for parallel processing, be tunable and take into account processing costs, because compressed data saves money but decompressing it can alter performance. Finally, the system should enforce immutability by disallowing mutable operations.

Using this procedure, we also tried to respect all the aspects the data should have. First, we kept the data as raw as possible, i.e deleting the minimum information during data preparation. Moreover, we enforced immutability of the data. In order to keep it simple and prevent human-faults we only add more data to the master dataset, instead of updating or deleting older data. Then, by keeping timestamps for each data record, the data remains eternally true. For example, the user X was connected to access point Y on a specific time TS: this data cannot become false by any event happening in the future.

We now have a more precise idea of how we are going to manage our data. However, before storing it, we need to generate a clean dataset. This prevents the system from dealing with poor or incorrect data, which saves time and resources of computation, but also avoids a wrong interpretation of the dataset.

In order to do so, we examined the data. There is a one minute interval between samples, which leaves room for a lot of potential errors. We quickly realized that there were some missing data: the first two days of wifi connections do not have a “clientOS” column and a “ssid” column. In the rest of the files, some “clientOS” fields are written as “unknown”, which can be considered as missing data, as well as some location fields that are filled with “change me” in the metadata file. In addition to that, we discovered further data quality issues in the room booking dataset. There are some inconsistent naming such as the study programs. Multiple rooms are written in one field, which generated duplicates in the dataset. For example, if a course was registered with three different classrooms, the record would appear three times in the dataset.

It is very important for our system that we have the best possible quality of data. Inside the flattening process, we perform data cleaning on the json files in order to delete any wrong entry or useless information. Since the process of data cleaning leads to putting the files on the server through HDFS, and Spark is deployed on HDFS, it made sense to use it. Spark is also fitted for repeatedly iterate over the same dataset. First, we define a schema where missing columns are introduced with a null value. We do that to prevent having errors of the kind ‘field not found’ in further usage of the data (batch view creation). In addition we wanted to have a more simple and flat structure of the wifi-data in order to store it as .cvs files in our master data set. By having the ‘did’ and ‘ts’ fields as an unique identifier for each record, the given wifi-data .json files came along with a very nested structure. By flattening the data, we introduced a new unique identifier, by adding the ‘cid’ (to ‘did’ and ‘ts’). We are aware, that by doing this, we will loose the records for those times, where no client was connected (e.g. during night). For the room booking files, we delete the duplicated rows. Then, we explode the rows that contain multiple rooms or lectureres. Note that we performed the cleaning of the room booking files because we were planning on using it for one of our batch views. However, as the project moved forward, we came up with more interesting views to implement that do not require room booking data. The fact that we don’t use those files does not mean that this cleaning is useless, as they could be useful in other applications. After cleaning, we store the flattened data in csv format on the HDFS.



Overview: Technical implementation of our big data project

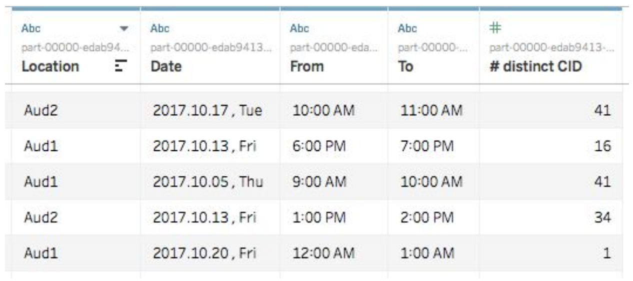
The diagram above outlines the full picture of our technical implementation. The data preparation and cleaning results in our master data set, which consists out of .csv files and is appended by new data on a daily basis. As described before this is used as a basis for the computation of our batch views. The nature of our batch algorithms is based on recomputation. This is because, two of them include a distinct cound of the client IDs, which is hard to implement in an incremental fashion. For the third view we also sticked to the recomputation approach. This approach is more human-fault tolerant, due to the fact, that failures are overwritten after the next time the batch layer runs. We know, that the recomputation algorithm offers a lower performance, but in the light of the Lambda Architecture, this is not a primary goal of the batch layer.

Batch views and services

As we previously mentioned, we can extract a lot of information from this master dataset. We reflected on what relevant use cases we could take out of it, to provide services that could be useful to different actors. When discussing our ideas about batch views, we encountered a trade-off between generality and specifity. On the one hand, we want the batch view to be as aggregated as possible to enable low-latency results to user queries. On the other hand, it should be able to not only specific questions. The discussion resulted in the three views, which will be described in the next part and whichs underlying code can be found in the appendix.

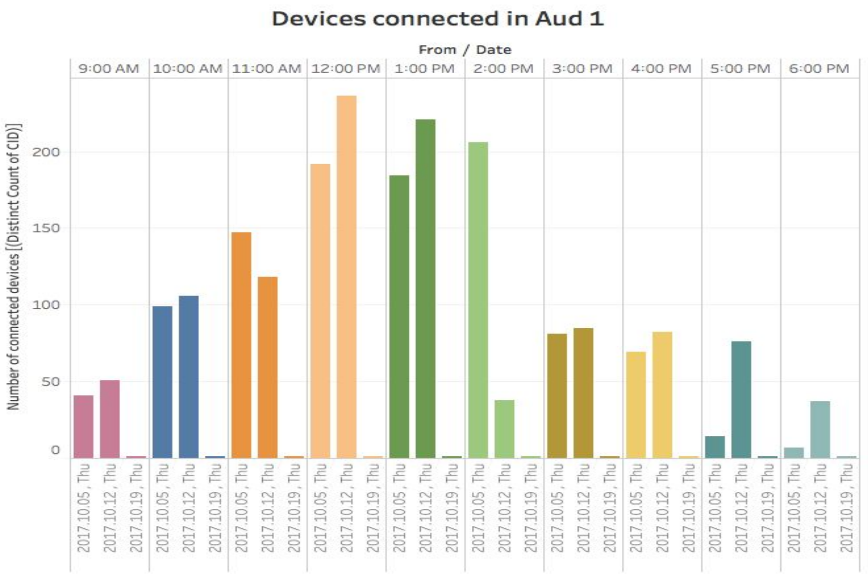
1) <u>Adapting the classroom schedule based on student connection</u>

First of all, we decided to focus on the number of connections that are made in Aud1 and Aud2 per hour everyday.



Abc	Abc	Abc	Abc	#
part-00000-edab94...	part-00000-edab9413...	part-00000-eda...	part-00000-...	part-00000-edab9413-...
Location	**Date**	**From**	**To**	**# distinct CID**
Aud2	2017.10.17 , Tue	10:00 AM	11:00 AM	41
Aud1	2017.10.13 , Fri	6:00 PM	7:00 PM	16
Aud1	2017.10.05 , Thu	9:00 AM	10:00 AM	41
Aud2	2017.10.13 , Fri	1:00 PM	2:00 PM	34
Aud1	2017.10.20 , Fri	12:00 AM	1:00 AM	1

Table of the number of distinct connections per hour in Aud1 and Aud2



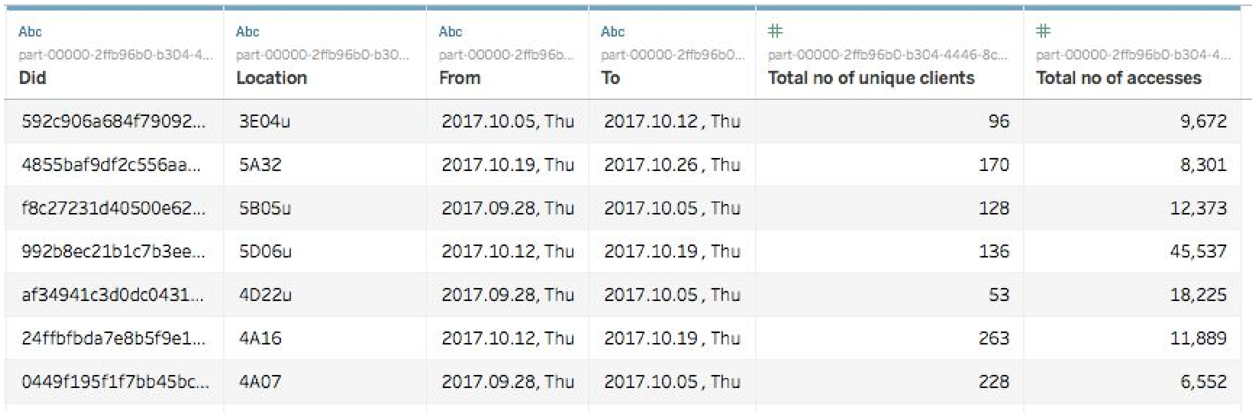
Graph of the number of distinct connections per hour in Aud1

To create this view, we merged data of different dids by their location (Aud 1 and Aud 2). We included a column with the date and the weekday, for simple filtering on the 'Query Layer' and displayed the giving the number of distinct clients ids (cids) connected during the given timeframe. We decided that is was more relevant to have a timeframe of one hour, as a lecture can be from one to three hours.

This batch view can then be used to get an overview of the occupancy of the two biggest lecture halls of ITU. As class sizes increase, those rooms are more and more requested, which results in lectures taking place late in the evening. Tracking the actual number of devices connected during the day gives insights on what room size is really needed for a given lecture. In our example, we displayed the Auditorium 1 occupancy on Thursdays. We can see for example that the class on Thursday from 3pm to 6pm is not that crowded. This leads us to the conclusion that it doesn't actually require the big room, and it could be rescheduled to an earlier time in a smaller room. As we all know, concentrating on the lecture in the late hours is really hard. Most of the students would prefer having a lecture at an earlier time in the day, since they are less tired, which makes them more focused and productive.

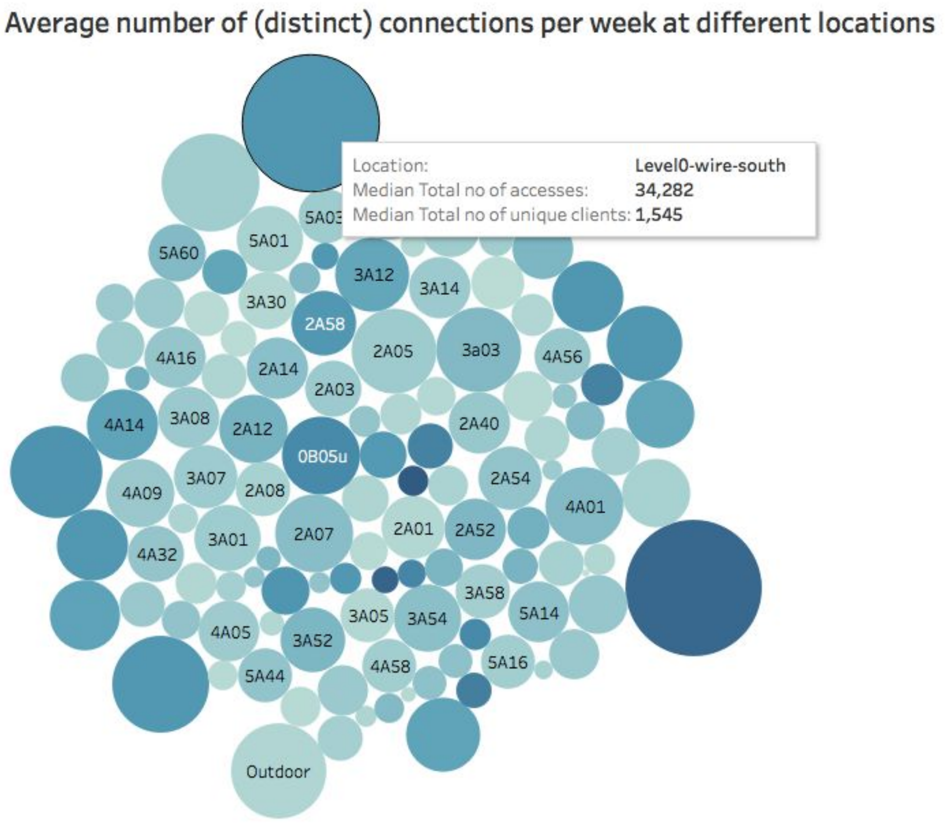
2) <u>Managing the announcements and advertisement repartition</u>

The second batch view we thought about considers the average number of connections per week at different locations.



Abc part-00000-2ffb96b0-b304-4... Did	Abc part-00000-2ffb96b0-b30... Location	Abc part-00000-2ffb96b... From	Abc part-00000-2ffb96b0... To	# part-00000-2ffb96b0-b304-4446-8c... Total no of unique clients	# part-00000-2ffb96b0-b304-4... Total no of accesses
592c906a684f79092...	3E04u	2017.10.05, Thu	2017.10.12 , Thu	96	9,672
4855baf9df2c556aa...	5A32	2017.10.19, Thu	2017.10.26 , Thu	170	8,301
f8c27231d40500e62...	5B05u	2017.09.28, Thu	2017.10.05 , Thu	128	12,373
992b8ec21b1c7b3ee...	5D06u	2017.10.12, Thu	2017.10.19 , Thu	136	45,537
af34941c3d0dc0431...	4D22u	2017.09.28, Thu	2017.10.05 , Thu	53	18,225
24ffbfbda7e8b5f9e1...	4A16	2017.10.12, Thu	2017.10.19 , Thu	263	11,889
0449f195f1f7bb45bc...	4A07	2017.09.28, Thu	2017.10.05 , Thu	228	6,552

Table of the average number of (distinct) connections per day at different locations



Location: Level0-wire-south
Median Total no of accesses: 34,282
Median Total no of unique clients: 1,545

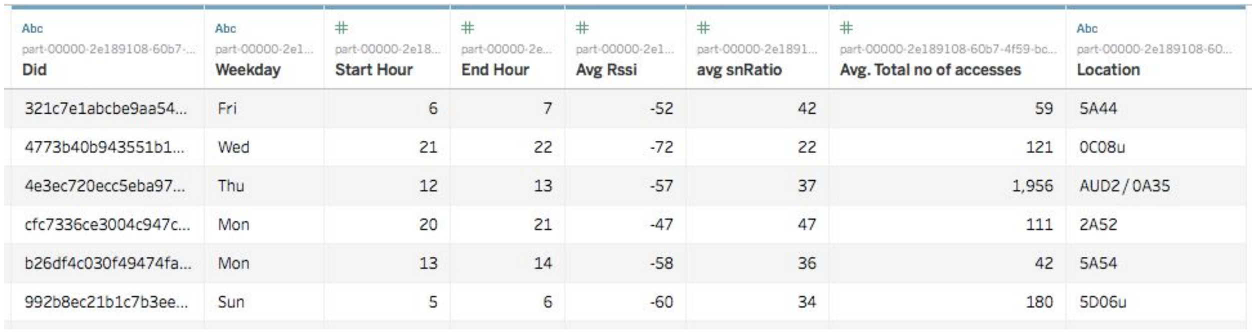
Graphical representation of the average number of (distinct) connections per day at different locations

We constructed this view by first joining did from the wifi connection files with location information from the metadata. We grouped the connections by week and calculated the number of different devices connected to an access-point (i.e counting the distinct cids) and the number of connections (i.e counting the cids). We chose to have a weekly timeframe because we estimate that it is a reasonable duration for announcements and advertisements turnover.

As shown in the graphical representation, each bubble represent a location at ITU, and the size is the average of unique clients connected to the access-point at this location. The darker the color of the bubble is, the more connections there are per week. The tooltip shown in the screenshot describes the upper big circle. The take-away here is that most people go through the basement, so important announcements (or advertising) could be placed there to reach as many people as possible.

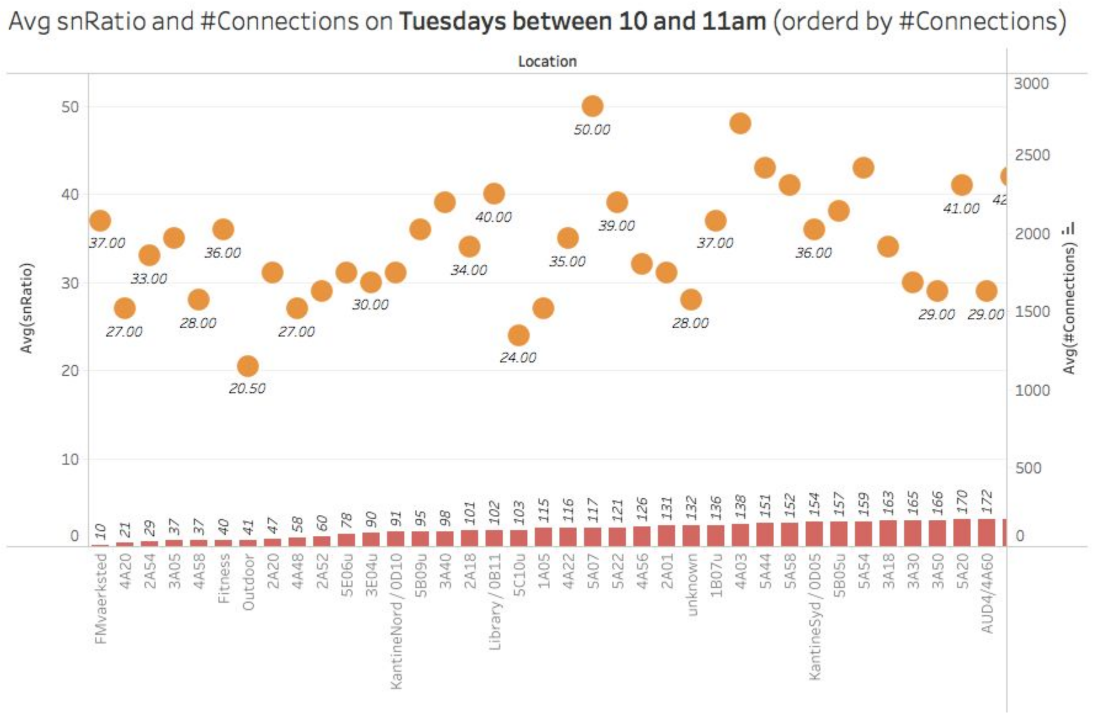
This view could be used to decide what are the strategic places to make announcements by the administration of ITU (e.g. where to put the billboard screens), but could also be used by other organizations to make advertisement, like companies putting job offers or other posters.

3) <u>Choosing a place to work based on previous wifi quality records</u>



Abc part-00000-2e189108-60b7-... Did	Abc part-00000-2e1... Weekday	# part-00000-2e18... Start Hour	# part-00000-2e... End Hour	# part-00000-2e1... Avg Rssi	# part-00000-2e1891... avg snRatio	# part-00000-2e189108-60b7-4f59-bc... Avg. Total no of accesses	Abc part-00000-2e189108-60... Location
321c7e1abcbe9aa54...	Fri	6	7	-52	42	59	5A44
4773b40b943551b1...	Wed	21	22	-72	22	121	0C08u
4e3ec720ecc5eba97...	Thu	12	13	-57	37	1,956	AUD2 / 0A35
cfc7336ce3004c947c...	Mon	20	21	-47	47	111	2A52
b26df4c030f49474fa...	Mon	13	14	-58	36	42	5A54
992b8ec21b1c7b3ee...	Sun	5	6	-60	34	180	5D06u

Table of the average wifi Received Signal Strength Indicator and Signal-to-noise Ratio per hour in different locations



In order to create this third view, we first grouped the records per device by weekday with a time window of 1 hour. We chose to make our averages for every hour because it seems to be a valid time period that people spend at one location. After that, we added three aggregated measurements : the sum of connections during the hour, the average Signal-to-noise ratio (the higher is the ratio, the better is the signal) and average Received Signal Strength Indicator (rssi) of the connections. Then, we added the location by joining our table with the metadata.

Having this view, we could imagine developing a nice interface as a mobile application for students where they could see which location has the best signal in the school. Our example shows the result of a user query by a student arriving on a Tuesday at ITU, wondering where to go to do homework. By sorting the rooms based on the number of connections, the student can get a nice overview, of possible locations with not too many connections and a good average signal-to-noise ratio.

Furthermore, this batch view could also be used for the maintenance of wifi by the IT-Department of ITU. By knowing which routers have the worst signal, the IT-Department could anticipate breakdowns or fix problems without having to wait for students to scan the QR codes on the wall to report an incident.

The Challenges of Designing the Future

As presented earlier, the main input for most of the services is the amount of connections to the different routers, allowing for measuring of network activities, amount of units connected, and the like. For ITU to provide WiFi to the students, they need certain information about the users. A WiFi solution where you do not give away any information (meaning log in credentials with your Eduroam or ITU email) would just be an open WiFi where anybody could log in, and would in principle be more un-secure. So given that the service ITU is providing is a WiFi service with more security than an open one, there is a need for some information about the users. This is necessary to deliver the service, and therefore we argue that you do not need consent for this collection of data in order to provide secure WiFi for the users.

If you decide to utilize the WiFi for guests at the ITU you have to agree with the *acceptable use policy*. First you need to register your name, your cell phone, and the person you are visiting. This will then be registered, and you will receive a text message with a temporary username and password (both for 24 hours). Then you can log in, but the interesting aspect here is that just by logging in (clicking the log-in), you accept the *acceptable use policy* (see Picture 1).

Looking into the *acceptable use policy,* we found point 4.1 particularly interesting:

"4.1 General Use and Ownership
1. While this company's network administration desires to provide a reasonable level of privacy, ***users should be aware that the data they create on the corporate systems remains the property of this company.*** *Because of the need to protect this company's network, management* ***cannot guarantee the confidentiality of information stored on any network device belonging to this company****."*

This is what you agree to when deciding to use the WiFi, stating that the data you produce when connected to the WiFi is now the property of ITU. Meaning, that the data is now theirs, and they can choose to use this data in the deliverance of services. This is how it is today, but this way of stating in an open manner that it is now their data would not be enough given the changes that will come by the Global Data Protection Regulation (GDPR) that will come into effect May 2018. So today, we argue that there is an underlying hidden consent when the user logs in with his/hers guest user, accepting that the data produced are now the property of ITU, which then means that ITU can use it in any way they want. This might deem problematic for ITU come next year, as GPDR states that *"Consent should not be a precondition of signing up to a service unless necessary for that service."* (Davis, 2017). Meaning, that if ITU are to be within the guidelines of GDPR, the part about the data *created on the corporate systems remains the property of this company*, needs to be clearly distinguishable from any other manners in a written document. It has to be stated in a clear language, and there should be an option to opt out of this (Maldoff, 2016).

Regarding the upcoming GDPR, the consent procedure for the presented services might deem problematic. In order for our services to deliver what they are suppose to, we would have to have unbundled, clear consent procedures, consistent with GDPR, for every service using the WiFi data as a provision. The consent procedures would be presented to every new WiFi user during sign up. Moreover, new consents should be asked every time a new service using the WiFi data is been released. Hereby, this has implications on how we are able to deliver the services. As a user of the WiFi might not want to use a service that allows him/her to see where the connection is the best, they need the ability to opt out. This then, means that the quality of the service for the other users, who consent to the use of their data, will be of a lesser quality. To exemplify this further, continuing with the example of connection quality, imagine that few people consent to the usage of the WiFi-data in order to use this service, then the service would have little to no value, as the data stemming from the consenting actors might come from the area with the best connection, but the service would say that this is the most crowded one. This exemplifies the challenges when it comes to consent, and usage of data.

Ethical reflection

The services are designed to deliver some value to a specific user group and they are expected to have a desired impact and affordances. However, the service functions can also be exploited by other actors for other uses. For example, the service that is built to optimize the usage of the large auditoriums might have a secondary effect; it could be used to evaluate the teachers and their lectures. ITU management could monitor the average student count in specific lectures and make conclusions on how engaging the teachers are. If the student count is decreasing significantly every week in a specific course, the management may take that as a proof that the teacher is not doing a good job. The student count statistics could be used as an evaluation metric, along with student reviews, and the like. However, that would be only speculation, since the management would not know the real reason(s) why the students are not attending the classes – possibly the teacher is providing exceptionally good slides and literature that you can easily study at home or he/she might even live stream the lectures. All in all, that is an example on how the numbers, the data presenting the object do not provide the big picture if you do not possess the domain knowledge – without understanding the local circumstances and the context it is highly questionable to make any assumptions. However, it might be tempting to see and form patterns from large amounts of data, but without contextual understanding the numbers might only increase uncertainty and lead to unvalidated decisions.

The data we use to create the services, end up as an informative object for the actor who is using the service. An informative object is always incomplete, and it is presented and shaped by the choices we have made earlier in the design process, meaning what data we choose to include and the like. However, this object is meant to create value for them. The value serves to reduce or increase uncertainty (Buckland, 1991), as where to sit, or where not to sit based on connection quality. Building on this, the concept of information as a process (Buckland, 1991), supports the notion that we can never be sure of the outcome or the usage of a certain information. To exemplify this, we can never be sure that the receiver of the information actually uses the information or understands the information in the same way we understand it, or the way we intended it to be used. Buckland (1991) looks at information as a process, not a transmission, but something that happens/changes/occurs. The process involves a sender and a receiver, and the process leads to some sort of change, which cannot be determined beforehand. Therefore, the concepts from Buckland (1991) allows us to think more broadly and become aware of the limits of our own perspectives, meaning we can strive to create a secure system, based on consent of how we foresee how it will be used, but we cannot be certain that we are safe from future implications, nor future vulnerabilities that make un-sensitive data sensitive.

Concluding remarks

Throughout this project we have been faced with the many challenges of designing services. The process started with us gaining insights and understanding of the data, and the opportunities that lie in this data. However, we came into this project with a critical perspective, meaning that we were aware of the power we have when it comes to designing a service, allowing us to choose what to include, who to include, and categorise what is valuable. As we are making these decisions, we are trying to validate them through our understanding of the context of use, the technical possibilities and the data itself. However, there are also limitations to this, as we are trying to design for a *better future,* for a lack of better words, but as with new laws and regulation coming into place next year, we saw the challenges of navigating these and still produce a service valuable to the end user.

Log

Going through our workflow of the project we have encountered many problems and challenges. With teamwork, research and help from our TA most problems have been overcome. But still, some of the issues have persisted until the end and they still have no solution.

Our first big challenge appeared from the even first day when we started looking over our data. On the big data server we could find 3 types of json files that we needed to read, understand and then use in our future data transformations. On a closer look we were able to see that to of the file types had a new file generated every day with new data about the room bookings and routers accesses. After taking a deeper look into the files, we were finally able to understand what each of attributes mean. Then we started discussing which of the attributes would be relevant for us. And one of the most difficult tasks have been to find 3 useful scenarios and decide about the 3 views that we would like to implement

One of the first decision we had to make when developing our application was where should we store our data. It was clear that we have to move data from the external resource to our server but what should we do next? We spend couple of hours discussing and researching what the

storage layer in lambda architecture actually is, how should it be used and how we have to operate on it. It was an interesting process where we learnt that there is no clear path or predefined best solution out there which we have to find. Every application is different so you need to consider all pros and cons and then create your custom design which you believe is the best fit for your custom solution. Our final process takes data from external resource and stores them on our server. We take only the latest data based on the current date. That's simply to reduce network traffic on the server. Then we move data to HDFS where Spark imports new files, cleans them, flattens them and generate csv files. These are our master data which we never edit after they were created. These generated files are then used to create Batch Views.
We are also taking care of the server by deleting resources after the entire process is finished. We discussed and agreed that there is no need to keep data on server simply because it just waste of space especially when we got out all what is needed. If there would be a potential for different data usage in the future we could simply reuse our master data. That's the major reason why these data should never be modified.

After we moved on from our initial problems we started to discuss how to automate some of the processes. It was ok to do data migration manually but this approach would obviously never work in real life applications. We agreed to use CRON on our server, what is basically a task scheduler which invokes certain bash script at certain predefined time. At the very beginning we were considering to use this bash script also for data cleaning. We had a hard time to load certain files into the Spark so we wanted to create one huge json file which would be a merge solution of all corresponding json files. Thankfully, we did not do this. We found out that we can load multiple files from scala file what helped us to determine cumbersome files. If we would combine them all into one we would never manage to find out which files are problematic.

Next task we had to tackle was data cleaning and data consistency. We were again very confused what exactly has to happen and how. Initially we thought there is no need to do it simply because data are consistent and there are no flaws. After discovering problems such as multiple data records in a record we knew that we have to properly clean our data in order to generate proper views later on. There was also a problem with missing data during first days what we discovered after trying to flatten them. This resulted in error which pointed out and ensured us in our decision. We made a decision to save generated data to csv format. We believe that this gives the data better structure for the manipulation and this format is also easier to read comparing to json.

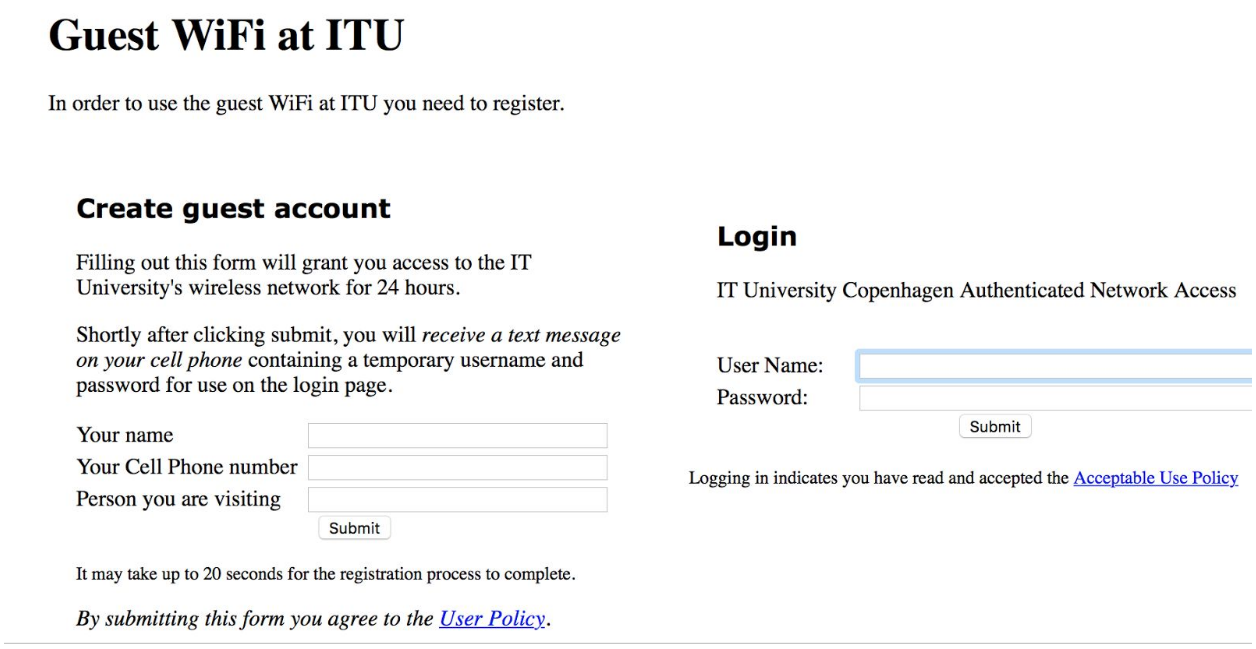
After we pretty much planned our project, we were ready to start coding. That’s when we realize that we had a lack of knowledge when it came to SPARK and SCALA. We made some research about it and read some tutorials in order to deal with them. We found out that there are actually a lot of methods and strategies to query through the datasets in order to create first our batch and then our views. So even though it seemed really hard at the beginning, we managed to filter through our data, to do operations with it, to merge some of the tables. An issue that gave us a lot of headaches was to convert the timestamp to datetime object in scala, so we can use it in to compare some times. We managed to do that in the end by converting our string to long, and only after that convert it again to datetime.

So we had to code pretty much all prepared for running on our computers. Unfortunately, we realize that our programs runs differently on our local comparing to the server. One of the issue that we weren’t able to solve (not even with the repartition) was the “Out of memory” error that we were receiving when we were trying to load some of the device jsons. We also tried to create some views with Drill-Explorer and use its syntax which seemed to be really nice, but we had some troubles with the ODBC server. But in the end we manage to find a stable version that has worked (at least with most of the files).

After we created the master data from the device and rooms jsons, we have decided to use some of the information offered in the meta.json file also. That’s when we realized that the meta.json file actually updates every day. So all the information from the days before will be constantly lost. We discussed if any of the old information from the old files could be relevant for our situation and if it needs to be stored. It seems that the deleted data was important only if the router would reset and then start all over with its upTime, or if new devices were added or removed. In the end we decided to rewrite the master data from the meta.json everyday.

Appendix

Picture 1



Guest WiFi at ITU

In order to use the guest WiFi at ITU you need to register.

Create guest account

Filling out this form will grant you access to the IT University's wireless network for 24 hours.

Shortly after clicking submit, you will *receive a text message on your cell phone* containing a temporary username and password for use on the login page.

Your name	
Your Cell Phone number	
Person you are visiting	

Submit

It may take up to 20 seconds for the registration process to complete.

By submitting this form you agree to the User Policy.

Login

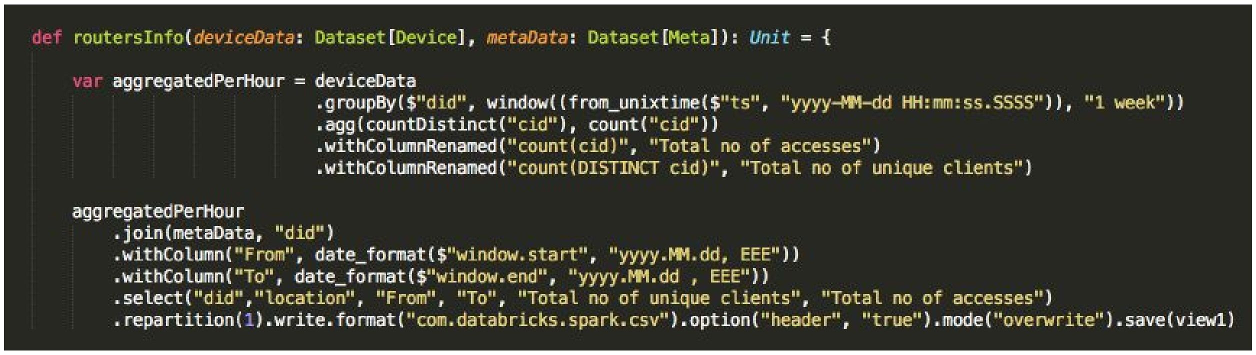
IT University Copenhagen Authenticated Network Access

User Name:	
Password:	

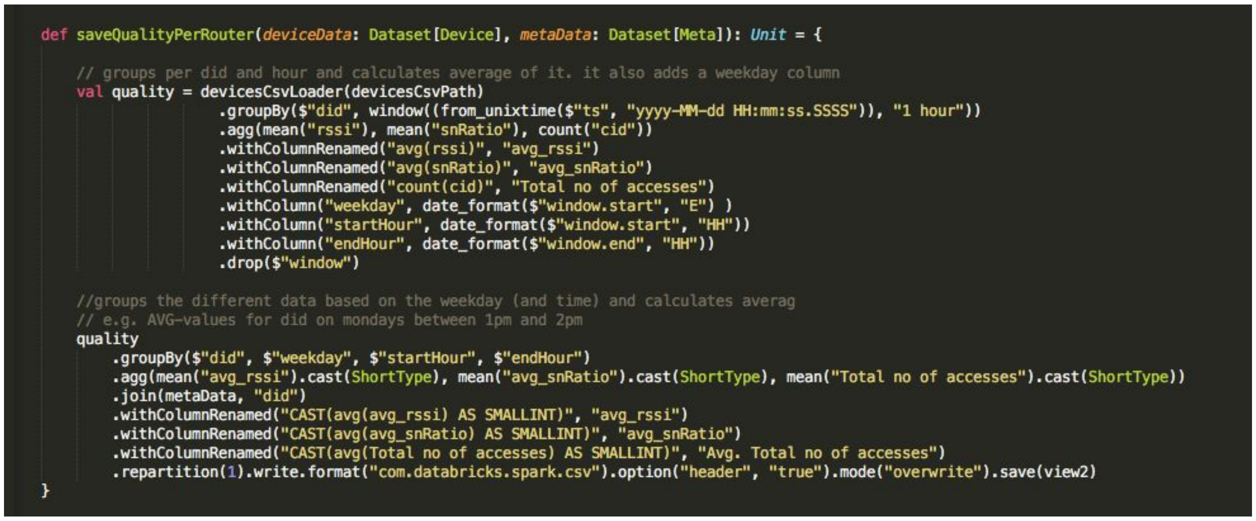
Submit

Logging in indicates you have read and accepted the Acceptable Use Policy

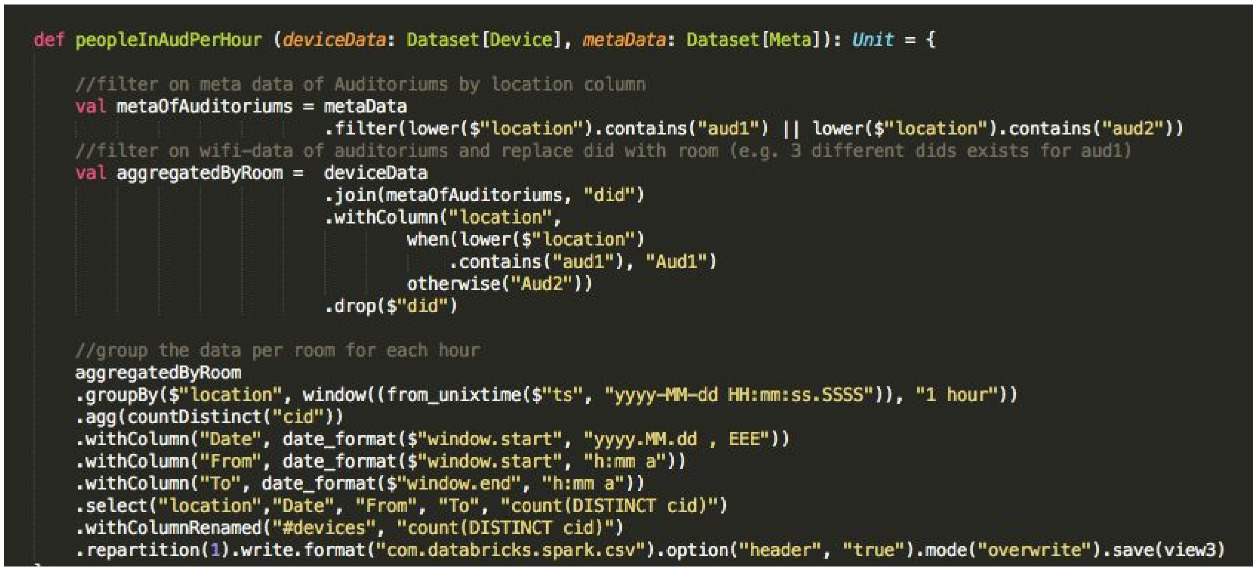
Code snippets of batch view creation:



```
def routersInfo(deviceData: Dataset[Device], metaData: Dataset[Meta]): Unit = {

    var aggregatedPerHour = deviceData
                            .groupBy($"did", window((from_unixtime($"ts", "yyyy-MM-dd HH:mm:ss.SSSS")), "1 week"))
                            .agg(countDistinct("cid"), count("cid"))
                            .withColumnRenamed("count(cid)", "Total no of accesses")
                            .withColumnRenamed("count(DISTINCT cid)", "Total no of unique clients")

    aggregatedPerHour
        .join(metaData, "did")
        .withColumn("From", date_format($"window.start", "yyyy.MM.dd, EEE"))
        .withColumn("To", date_format($"window.end", "yyyy.MM.dd , EEE"))
        .select("did","location", "From", "To", "Total no of unique clients", "Total no of accesses")
        .repartition(1).write.format("com.databricks.spark.csv").option("header", "true").mode("overwrite").save(view1)
```



```scala
def saveQualityPerRouter(deviceData: Dataset[Device], metaData: Dataset[Meta]): Unit = {

    // groups per did and hour and calculates average of it. it also adds a weekday column
    val quality = devicesCsvLoader(devicesCsvPath)
                    .groupBy($"did", window((from_unixtime($"ts", "yyyy-MM-dd HH:mm:ss.SSSS")), "1 hour"))
                    .agg(mean("rssi"), mean("snRatio"), count("cid"))
                    .withColumnRenamed("avg(rssi)", "avg_rssi")
                    .withColumnRenamed("avg(snRatio)", "avg_snRatio")
                    .withColumnRenamed("count(cid)", "Total no of accesses")
                    .withColumn("weekday", date_format($"window.start", "E") )
                    .withColumn("startHour", date_format($"window.start", "HH"))
                    .withColumn("endHour", date_format($"window.end", "HH"))
                    .drop($"window")

    //groups the different data based on the weekday (and time) and calculates averag
    // e.g. AVG-values for did on mondays between 1pm and 2pm
    quality
        .groupBy($"did", $"weekday", $"startHour", $"endHour")
        .agg(mean("avg_rssi").cast(ShortType), mean("avg_snRatio").cast(ShortType), mean("Total no of accesses").cast(ShortType))
        .join(metaData, "did")
        .withColumnRenamed("CAST(avg(avg_rssi) AS SMALLINT)", "avg_rssi")
        .withColumnRenamed("CAST(avg(avg_snRatio) AS SMALLINT)", "avg_snRatio")
        .withColumnRenamed("CAST(avg(Total no of accesses) AS SMALLINT)", "Avg. Total no of accesses")
        .repartition(1).write.format("com.databricks.spark.csv").option("header", "true").mode("overwrite").save(view2)
}
```



```scala
def peopleInAudPerHour (deviceData: Dataset[Device], metaData: Dataset[Meta]): Unit = {

    //filter on meta data of Auditoriums by location column
    val metaOfAuditoriums = metaData
                        .filter(lower($"location").contains("aud1") || lower($"location").contains("aud2"))
    //filter on wifi-data of auditoriums and replace did with room (e.g. 3 different dids exists for aud1)
    val aggregatedByRoom =  deviceData
                        .join(metaOfAuditoriums, "did")
                        .withColumn("location",
                                when(lower($"location")
                                    .contains("aud1"), "Aud1")
                                otherwise("Aud2"))
                        .drop($"did")

    //group the data per room for each hour
    aggregatedByRoom
    .groupBy($"location", window((from_unixtime($"ts", "yyyy-MM-dd HH:mm:ss.SSSS")), "1 hour"))
    .agg(countDistinct("cid"))
    .withColumn("Date", date_format($"window.start", "yyyy.MM.dd , EEE"))
    .withColumn("From", date_format($"window.start", "h:mm a"))
    .withColumn("To", date_format($"window.end", "h:mm a"))
    .select("location","Date", "From", "To", "count(DISTINCT cid)")
    .withColumnRenamed("#devices", "count(DISTINCT cid)")
    .repartition(1).write.format("com.databricks.spark.csv").option("header", "true").mode("overwrite").save(view3)
```

Sources

Davis, B. (2017). *Blog post in Econsultancy webpage July 2017:* https://www.econsultancy.com/blog/69253-gdpr-10-examples-of-best-practice-ux-for-obtaining-marketing-consent

Dourish, P. (2017). 5. *No SQL: The shifting materialities of database technologies.* In The Stuff of Bits: An essay on the materialities of information. MIT Press

Buckland, M. (1991). *Information as thing.* Journal of the American Society for Information Science (1986-1998)

Gitelman and Jackson (2013). *Raw Data is an Oxymoron - Introduction File.* Massachusetts Institute of Technology

Maldoff, G. (2016). *Blog post in iapp website January 2016:* https://iapp.org/news/a/top-10-operational-impacts-of-the-gdpr-part-3-consent/

Shklvoksi, I. (2017). *Week 3 lecture 12.10.2017.* Critical Big Data Management: Second Part of Specialisation (Autumn 2017). IT University of Copenhagen.