UiT The Arctic University of Norway

Faculty of Science and Technology
Department of Computer Science

Data collection and combination for smart nudges in physical activity

Lars Karlsen

INF-3981 Master's Thesis in Computer Science - June 2020

To my friends and family;

thank you for all the support.

"Say my name."
–Walter White

"Sometimes I'll start a sentence and I don't even know where it's going. I just hope I find it along the way."
–Michael Scott

Abstract

Today's adults and adolescents do not have enough physical activity in their life. Being active enough is crucial for a physically healthy life, but in today's modern world, this is becoming more problematic. In 2018 The World Health Organization (WHO) launched a global action plan on physical activity to reduce physical inactivity in adults and adolescents by 15% by 2030[1].

It goes to show that people are not active enough, and people's mindset and motivation towards physical activity needs to be changed, and this can be done through nudging. Nudging is about altering people's behaviour without limiting their choice. It could be to make people behave in a certain way or make it more likely for people to make a particular choice. People are different, and the sources that people are affected by varies, and therefore, the nudges need to be smart. *Smart nudging* can be defined as a digital nudge where the guidance is tailored to be relevant to the individual user and their current situation. It could be that the nudge knows that the person is old, and therefore can not go for a jog, or that a person has time for a hiking trip on Saturday, as the person has no plans for that day and the weather is going to be sunny.

In this thesis, data sources that have been found as relevant to physical activity in previous work are collected data from. Data is collected by fetching data from sensors, requests to third-party sources, fetching data from Web Scraping, and more. The data were tested to see if smart nudges could be made using the actual data from the relevant sources, and some examples of smart nudges were generated. A smart nudging system is designed using a microservice architecture to show how the data sources and their relevant data can be combined and used in a system that should influence people to have better physical activity.

Acknowledgements

I would like to thank my supervisor, Randi Karlsen, for excellent help and guidance. I am very thankful for the feedback she has given throughout this thesis, and that she had good follow-up. I have learned so much from the feedback you have given me and from working on the NUDGE project.

Thanks to everyone in the Open Distributed Systems (ODS) group, and thanks to Jacob Johnsen and Marius Mæland for good conversations and discussions about the project, and of course all the fun we have had at the office.

To my family and closest friends, thank you for believing in me and caring. I am very grateful for the support you have given me, and I could not have done it without you.

Contents

List of Figures

List of Tables

List of Abbreviations

API Application Programming Interface

CRUD Create, Read, Update, and Delete

DNT The Norwegian Trekking Association

ECG Electrocardiography

EU European Union

GDPR General Data Protection Regulation

GPS Global Positioning System

HTML HyperText Markup Language

HTTP Hypertext Transfer Protocol

IDC International Data Corporation

IoT Internet of Things

JSON JavaScript Object Notation

ODS Open Distributed Systems

REM Rapid Eye Movement

SLR Structured Litterature Review

SOA Service-Oriented Architecture

SQL Structured Query Language

TF-IDF Term Frequency-Inverse Document Frequency

UiT Arctic University of Tromsø

URL Uniform Resource Locator

WHO The World Health Organization

XML Extensible Markup Language

Glossary

MERN stack is a JavaScript stack that includes technology that is needed to develop a full-stack application. The stack consist of MongoDB, Express, React and Node.js. With this use of this stack, the development process can be done using only JavaScript.

Web Scraping is a technique to fetch data from websites by developing software to simulate web surfing and collecting the relevant information on the way.

1 Introduction

The European Union published in 2017 a survey[2] about physical activity. The survey focused on the daily exercise of people, and the results showed that 15% of Europeans who took the survey do not walk for 10 minutes at a time, *at all*, in a week. The survey also discovered that only 7% of Europeans exercise regularly(at least five times per week) [2]. Having a good physical active lifestyle have many health benefits. Some of the benefits that WHO points out is reduction in stress and anxiety, prevent obesity and heart diseases, and reduce the risk of some types of cancer, diabetes, and cardiovascular diseases[3]

1.1 Background and motivation

In today's modern world, there has been a rise in new services and products that are designed to ease everyday tasks and transportation for people. Activities such as mowing the lawn or going to the supermarket to buy groceries are created products and services for, so people do not need to do them.

In recent time, Foodora[1] and Uber Eats[2] has grown rapidly. These are businesses that pick up takeaway food from restaurants for people, and deliver it home to them, so they do not have to go there themself. The service is offered in almost

1. https://www.foodora.com
2. https://www.ubereats.com

every big city in the United States and Europe. A business in Norway, called Kolonial[3] does the same as Foodora and Uber Eats, but with groceries instead of takeaway food. In recent years electric scooters have become more and more popular in cities. These electric scooters are mass-produced by companies and put into big cities to help people get from point A to point B in an easy and fast way. Food delivery and electric scooters are some examples of services that have grown in recent years that are trying to ease everyday tasks for people. By removing or easing tasks by designing products and services, it will reduce the amount of physical activity performed by people.

People are not active enough, and today's modern world is moving towards a trend where products and services are designed to ease tasks, or make the tasks someone else's problem. Many factors affect why people are not engaging enough in physical activity. Some of these factors could be:

- Motivation.
- Ignorance.
- Time.
- Cost.
- Age.
- The feeling of being out of shape.
- Easier to live inactive because of products and services that remove many reasons for being active.

Most gyms have personal trainers that are hired to help people reach their fitness goals. The personal trainers try to find ways to motivate their users and form a workout plan that fits the user's motivational factors and goals. The personal trainers have the possibility to follow-up with their users and make adjustments depending on how the workout plan is going. Some people might need more creative and fun exercises to be more physically active, while others do not. The personal trainer can influence the motivation and progress towards the goal of being more physically active for their users. However, having a personal trainer is expensive, and it is not for everyone. What many people do when they are finished working with a personal trainer is to slowly go back to the old routines, where they lacked physical activity in their daily life. People need something that can learn what is best for them and apply this knowledge

3. https://kolonial.no/

to influence their behavior to become more physically active, just as a personal trainer, without having to think about the limitations such as cost, age, and time.

Nudging is a term about altering people's behavior and decision-making without limiting their options. *Smart nudging* is a term built on top of nudging, that takes the current situation of the user into account, and personalize the nudge towards the user. There are a lot of different ways to nudge people, and these are explained in more detail in Chapter 1.1. Smart devices such as smartphones and smartwatches have become more and more popular in recent years. These devices, throughout the years, have been filled with new hardware that has better processing power, more space in memory, more storage, and filled with more sensors[4]. Many of these sensors hold relevant information that can be used to nudge people towards being more physically active. Developing a system that can fetch data from relevant sources and use them together to create smart nudges that people can receive on smart devices through an application can help people be more physically active.

1.2 Challenges

People react differently to every situation. Many factors affect behavior and decision making, and some of these factors can not even be explained. There is no general rule or preference that can be used to affect everyone when it comes to influencing people to be more physically active, and that is a big challenge. The nudges need to take the circumstances around the user into account, as well as learning about the behavior and preferences of the user to be able to have the maximum likelihood of nudging them correctly.

As mentioned earlier, the use of smartphones and smartwatches are proliferating, and with a mobile application, these devices could be an excellent platform for nudging people to better physical health. These devices have many sensors in them, and a challenge for this master's thesis is to find out which of these sensors could be of use and how the data from these sensors can be extracted. Some of the data from these sensors can be used alone. In contrast, other data need to either be combined with data from other sources or analyzed further with data aggregation or machine learning models to get relevant information from the data.

The development of a smart nudging system can be challenging. There is an unknown amount of relevant data sources that can be of use, and the system needs to be versatile enough to affect most people. Multiple sources can be used to form smart nudges towards physical activity. These can come

from sensors, user data, or third-party sources. Some of the analysis could be done on edge. At the same time, other analytical approaches could need more computational power or more storage, and therefore need to be computed on one of the servers. However, how can these sources be combined and used as personalized nudges that should affect the behavior and interest in physical activity for a system?

1.3 The Purpose

There are many examples of nudges, and a lot of them are not personalized. Throughout this project, discussion about what sources and their data that can be applied to a personalized digital nudge was done. The nudge should guide people in the right direction towards physical activity, and help them be motivated and informed about what they can do to better their health.

To be able to personalize nudges towards each user, data from relevant sources around individual users needs to be collected. The purpose of this project is to collect data from different relevant sources and see how these sources can be used together and integrated into a smart nudging system. In the bigger picture, the purpose of this master's thesis is to give more knowledge to others when the development of such a system is taking place, so techniques on how data is fetched, and the proposal of the architecture of the smart nudge system can be learned from.

1.4 Goal

The goal of this master's thesis is to collect data from relevant information sources supporting nudges for a healthy lifestyle, in terms of physical activity. The information gained from these sources will be used to design different types of nudges depending on the situation of the user. The research question is formulated like this:

> *How can data be extracted from relevant sources and used to create smart nudges by a system to influence people to increase their physical activity?*

The main focus of this thesis was to find ways to collect information from different relevant sources and implement demonstrators that included the collection of data, and how the data could be used in a smart nudge system.

1.5 Contribution

The master's thesis contributes the ODS group and other researchers interested in personalized digital nudging with identifying what relevant data the data sources hold, how data can be extracted from multiple data sources, and how the data sources with their data can be integrated into a smart nudging system to generate relevant nudges.

1.6 Benefits and Ethics

People have different motivational factors and goals towards physical activity, and therefore, there is no specific recipe for having a healthy lifestyle. A smart nudging system that focuses on physical activity needs to benefit people who are looking to improve their lifestyle by being more physically active. It can be people that have hardly exercised before in their life, or on the other end, people with lots of exercise in their daily life, but want motivation to stay at that healthy level of physical activity. It should also benefit a wide range of ages, from elderly who tries to walk more daily, to young, energetic people who might want to do more energetic exercises. The system should be able to adapt to these factors.

There is a debate on whether or not nudging limits the autonomy and free will of people, and if it is ethical or not to nudge people towards some choice. Cass R. Sunstein[5], who is a legal scholar, wrote a paper about the ethics of nudging where he argued that nudging is ethical as long as they are transparent, and they do not limit people's choices[5]. An example of this could be that the government designs the cigarette packs with pictures of some of the side effects instead of banning convenience stores from selling cigarettes. In the blog post *On nudge, free will, and ethics*[6], Steven Senior brings up the discussion about free will, and if people do have any free will in the first place, and that there are things that influence our choice that is out of our control. Some try to influence others by having a specific goal in mind for themselves, like economic benefits, rather than steering people in the right direction to help them. S.Senior concludes by saying that nudges are essential tools to help people live healthier lives, but only if it is transparent and the goal and motivation is to improve our well-being.

To be able to nudge a user efficiently, the system needs to know its users. Not only in general about its users but tailor each nudge specifically for individual users. Relevant information about the user is therefore necessary, and the amount of information that could be relevant about the user is significant. The danger here is that personal information could be collected from the data that

is fetched about a user, and this could increase the risk of violating the privacy of the user. Also, much information could be sensitive to the public and could increase the risk of the security of individuals.

1.7 Methods and Methodologies

Anne Håkansson published in 2013 a paper called "*Portal of Research Methods and Methodologies for Research Projects and Degree Projects*"[7]. She designed a portal that helps people pick the right research methods and methodologies based on the research and the goal of the project. By following this portal, more structural research can be conducted, and irrelevant methods can be filtered away in the process.

The goal of this thesis is to collect data from different sources and looking at how this data can be used both individually and combined to design smart nudges for a system. Therefore, it is natural to approach this assignment with *qualitative research*, as the opinions and behavior of people are important, and the design of smart nudges needs to be created through the understanding of thoughts and experiences of the users of the system.

Håkansson mentions multiple research methods for research projects, and one that fits this thesis is *applied research*. This thesis tries to solve a known and practical problem, making it easier for people to be more physically active by giving more insight for how a system can be made to personalize nudges to users for a more physically active lifestyle. This thesis uses data from other relevant systems and data from existing research to make demonstrations of applications that can be applied in a real system.

I worked on a capstone project[8] last semester, where the research problem was about finding out what effects people to be less/more physically active. A *research approach* is used to find an approach to how a conclusion can be drawn. This master's thesis will make use of the *inductive approach* as the data collected from the different sources and designing of nudges are based on patterns and observation from the previous work mentioned, combined with the analyzation of the new data gathered from the relevant sources that were found from previous work. A more detailed explanation of the capstone project is done later in the thesis.

The strategy of this thesis was to conduct a *exploratory research*, as the goal was to look at how smart nudges could be created by combining and looking at the relationship between the different sources, and their data, and how this could fit into a system.

1.8 Delimitations

As mentioned earlier in this research, the platform that was kept in mind throughout this project was smart devices, because of the rapid growth. This research was limited to serving clients who use the system from wearable smart devices, like smartphones and smartwatches, as these are the most common ones, and includes sensors which are relevant for physical activity. The data sources found in this report were limited to benefit users in Norway, as finding sources to work globally would increase the complexity significantly.

1.9 Context of Project

I have been part of the ODS group, which is a group at the Computer Science Department at the Arctic University of Tromsø (UiT) for a semester before working on the master's thesis. In this period, I researched the same domain, smart nudging in physical activity, in the form of a capstone project[8]. The research focused on identifying what sources that are relevant for nudging people towards better physical activity, and what data could be relevant to extract from these sources. The project conducted qualitative research where data was collected through interviews. The respondents were five people with different level of physical activity and lifestyle, and the goal of the interview was to collect data for what increases and decreases the motivation to be more physically active. I analyzed the data collected with the use of a thematic analysis, where different themes that affect their motivation to be more physically active was discovered.

This master's thesis is a continuation of the capstone project, and the goal for this masters's thesis was to extract data from the relevant sources found in the capstone project, and find ways to use them to motivate and affect the behavior of people towards better physical activity by integrating them into a system that creates smart nudges.

1.10 Outline

The outline of the thesis is as follows:

Chapter 2 - Technical Background
This chapter introduces and explains some of the key topics and information that is needed for this thesis. It includes what nudging, digital nudging, and smart nudging are, what sensors that are available on smartphones and smartwatches,

and information about some of the frameworks that are used in this thesis, as well as general information about physical activity.

Chapter 3 - Related Work
In Chapter 3 a Structured Litterature Review (SLR) is applied. The process of the litterateur review is explained in detail, and the articles which are part of the result are explained in more detail, as well as how they are different from this master's thesis.

Chapter 4 - Methods and Methodologies
In this chapter, the methods for how data was collected is explained, how the SLR was conducted, as well as how the data was analyzed to be able to reach the goal of the thesis is explained.

Chapter 5 - Design
This chapter explains the overall architectural design of the system and explains how data is sent through the system. The chapter also dives into the different microservices of the system, to explain how each one of the microservices can be built.

Chapter 6 - Implementation
In Chapter 6, demonstrators are implemented and explained in detail. It describes how data has been extracted, as well as how some microservices has been implemented.

Chapter 7 - Smart Nudge Generation with Obtained Data
In this chapter, data that has been extracted from relevant sources are analyzed closer to see how data can be used together to form smart nudges. Examples of smart nudges are generated using data that has been extracted, together with the result of looking at the patterns between the data.

Chapter 8 - Discussion
This chapter discusses the design choice made for the system and compares it with a monolithic architecture. This chapter also looks into how the data sources and their data is relevant for generating smart nudges, and how other sources that are not used in this thesis could be relevant too.

Chapter 9 - Conclusion
In this chapter, a summary and conclusion are made, together with some ideas on future work for this project.

2

Technical Background

This chapter covers the theory around the topics that are essential in this thesis. These are the main topics such as what physical activity is, what nudging and smart nudging is, and what frameworks and technologies that are used. Sections such as 2.1 Physical activity, 2.2 Nudge and Nudging and 2.4 Smart Devices are sections that are relevant in both this thesis, and the capstone project I conducted earlier. Therefore, these sections are written and collected from that project[8].

2.1 Physical activity

Physical activity is about moving the body. WHO[9] define physical activity as:

> *... any bodily movement produced by skeletal muscles that requires energy expenditure*[9].

A common misconception is that people think physical activity means to exercise. Exercise is only a subcategory of physical activity as there are plenty of ways to be physically active without exercising. It could be cleaning, walking to the grocery store or gardening, but also more intense activities like brisk walking, cycling, jogging, and weight training.

Physical activity can be categorized into three different types based on the intensity of the activity. It is the energy expenditure used in the activity. The different types are *light physical activity*, *moderate physical activity* and *vigorous physical activity*. Light physical activities are activities like casual walking, picking up light objects, and other activities that require minimal energy expenditure. Moderate physical activities can be jogging, brisk walking, cycling with moderate intensity, and light swimming. It is activities that get the pulse a bit up, but not too high. Lastly, vigorous physical activities are activities with high energy expenditure. It is activities like sprinting, fast cycling, and fast stair climbing. Activities that get the pulse up to an uncomfortable state with close to max effort.

Physical inactivity is a big problem in today's society. Many people are not engaging enough in physical activity in their everyday life. According to WHO, physical inactivity is the fourth leading risk factor for mortality[9], as well as the main cause of roughly 25% of breast and colon cancers[9].

2.2 Nudge and Nudging

2.2.1 Nudge in general

Richard H. Thaler and Cass R. Sunstein[10] published in 2008 a book called *Nudge: Improving Decisions about Health, Wealth, and Happiness*[10]. The book defined a nudge to be[10]:

> ··· *any aspect of the choice architecture that alter people's behavior in a predictable way without forbidding any options or significantly changing their economic incentives.*

A nudge tries to make it more likely for an individual to make a particular choice, or behave in a certain way, and there exist many different techniques for nudging. Cass R. Sunstein wrote a paper called *Nudging: A Very Short Guide* [11], where he lists up what he means is the ten most important nudges. These nudges are [11]:

1. **Default rules**: Give people a default option in their decision. There are situations where an active decision is needed, but in many cases, default rules are indispensable, because it is time-consuming and feel much like a burden to require people to choose.

2. **Simplification**: Complexity is a big problem, and it can, in many cases, confuse people when deciding. The effect of simplification is something

that is underestimated. In many nations, the benefit of essential programs about education, finance, and health are reduced because of undue complexity[11].

3. **Uses of social norms**: People often get affected by what most other people do. Giving people information about what most others are doing in a particular decision or having a certain behavior in a specific situation is powerful. Examples of this can be “most people take the bus to work" or “eight out of ten people throw their garbage in the rubbish bin."

4. **Increase in ease and convenience**: People are often resistant to change, and a big reason for this is the perceived difficulty. People are often driven towards the natural choice, and therefore, the different barriers that make a decision hard to choose need to be reduced. It could be done by giving more information about the topic, making the decision easier to visualize.

5. **Disclosure**: New or secret information about a topic could be of great importance for people when making decisions. A nudge should be for the greater good; therefore, the nudge’s supplier needs to be transparent about the information and background for doing this. It could be that the government is open about what data they collect about its users, or that all information is laid out about the economic or environmental costs surrounding driving diesel or gasoline vehicles. Hiding information that is negative or positive towards one side of the topic might make people choose the other side just because people do not know better, and information is hidden from them.

6. **Warnings, graphics, or otherwise**: If there is high risk involved in the topic of the nudge, making personal or public warnings could be highly effective. A warning is supposed to catch the attention of people, and a way to do this is to make the warning with bright and sharp colors, large font size, and bold letters.

   Graphics could also make a big difference in the decision-making for people. It could be showing the result of what happens if people do or do not do a specific action. Examples of this can be images of lungs after smoking, or video of the effects of global warming in the Arctic in the past years. In Canada, 84% of smokers stated that graphical warnings were a useful source of information[12]. In Singapore, a survey showed that 28% of smokers reported that they smoke fewer cigarettes because of the graphical warnings[13].

7. **Precommitment strategies**: Most people have specific goals in life. It

could be to stop smoking, start to save money, or engage in more activity, but because of their behavior and motivation, people often fall short of those goals. With these goals, people are already pre-committed to behave or engage in a specific action, which will make people more likely to choose according to their goals. Precommitment strategies work around the idea that people are already pre-committed to an idea or behavior, but only need a small pull in that direction to make it work.

8. **Reminders**: Many people may feel that many things are happening, or that we start to procrastinate, or forget things. Our mind has a hard time keeping control of everything, and in these situations, a simple reminder can make a big difference. Timing is of great importance as the reminder should make people act almost immediately, as people might forget later to do something about the topic. It could be to remind people to pay bills when they are home, or take medicines when they should be taken, or to put on a raincoat before going to work.

9. **Eliciting implementation intentions**: People are most likely to engage in activity if someone elicits their implementation intention[11]. It could be to ask a simple question like *"Do you care about your health?*" or *"Do you visit your parents enough?*".

10. **Informing people of the nature and consequences of their past choices**: The government and institutions might hold much information about people and their past. People might change behavior if they could obtain information about their expenditures on specific things like alcohol, health care, or even fuel.

The number and variety of nudges are growing steadily, and we see through these ten different types of nudges, the difference between some of them is pretty big. Finding the right kind of nudge is crucial in systems that depend on influencing users towards a specific goal.

2.2.2 Digital Nudge

A digital nudge does not differ theoretically from a regular nudge. A digital nudge is defined as *the use of user interface design elements to guide people's choices or influence users' inputs in online decision environments*[14].

The world is going towards a trend where a lot of services and commerce are getting digital. With the growth of these digital technologies, there come many decisions that need to be made within the digital environment. User interfaces will steer people towards a direction, and therefore, making an interface in

which the designer knows the behavioral effects are of great importance.

An example of a digital nudge is from the paper *Do defaults save lives?*, which is written by Johnson, Eric J., and Goldstein, Daniel[15], where simply changing the default option in being an organ donor from opt-in to opt-out almost doubled the percentage of people who decided to become organ donors[15].

2.2.3 Smart Nudge

As seen in Section 2.2.1 about nudges, there are a lot of essential ways nudging can be accomplished, but people react differently to everything. Therefore, some people might not be as affected by graphical warnings as others. A personal nudge will try to personalize a nudge towards each specific person, so the probability of affecting people will be as high as possible. People might need different types of nudges, and the frequency might differ as people need different amount of push to make them change behaviour. Being able to nudge people in a particular way that is tailored around people's preference and combining it with the gained knowledge of its users and the context of their situation makes the nudge a smart nudge.

A smart nudge can be defined as *digital nudging, where the guidance of user behavior is tailored to be relevant to the current situation of each user*[16]. Being able to tailor information about each user can be done by monitoring and collecting data from different sources through user-profiles and analyzing these with respect to each user's context. A nudge that is more personal and tailored towards a specific person will most likely be a better choice than giving a general nudge in most cases.

Figure 2.1 shows how to design a smart nudge. This figure illustrates eight steps that are needed when designing a smart nudge where the first step starts as *Define the goal*. The architectural idea and figure are collected from *Recommendations with a Nudge*[16].

1. **Define the goal**: To define the goal of a nudge. What is the nudge trying to achieve?

2. **Understand the users**: Understanding the users are essential, and in this step, trying to understand the users in general, with regards to decision making and behavior as well as understanding each specific user. Input can be relevant from a user profile, which reflects the user's response to previous nudges so that nudges in the future can be tuned based on user reaction and preference.

3. **Understand the situation**: Trying to understand the situation around the user. It can be if the user is home, at work, or in the daytime or at nighttime. It might be relevant for building up a nudge around a user, depending on the context.

4. **Select targeted activity**: Select a targeted activity that should be applied or suggested for the user, depending on the previous steps.

5. **Select relevant information**: Collect information needed from different sources that are relevant for designing the nudge for the user.

6. **Design smart nudge**: Design a smart nudge with the selected activity and the relevant information gained in the previous steps, and use this to design a specific type of nudge to the user.

7. **Present nudge**: When the nudge is designed, it should be presented to the user during the relevant time and form in regards to when a decision should be made.

8. **Evaluate the nudge**: In this step, the nudge is evaluated. It means to see if the nudge was effective or not. It can be done by monitoring the user, or simply by feedback by the user. If the nudge was successful, we know what an effective nudge might be towards that specific user. However, if the nudge was not successful, some of the steps might need to be modified, as the selected targeted activity or the selected relevant information.

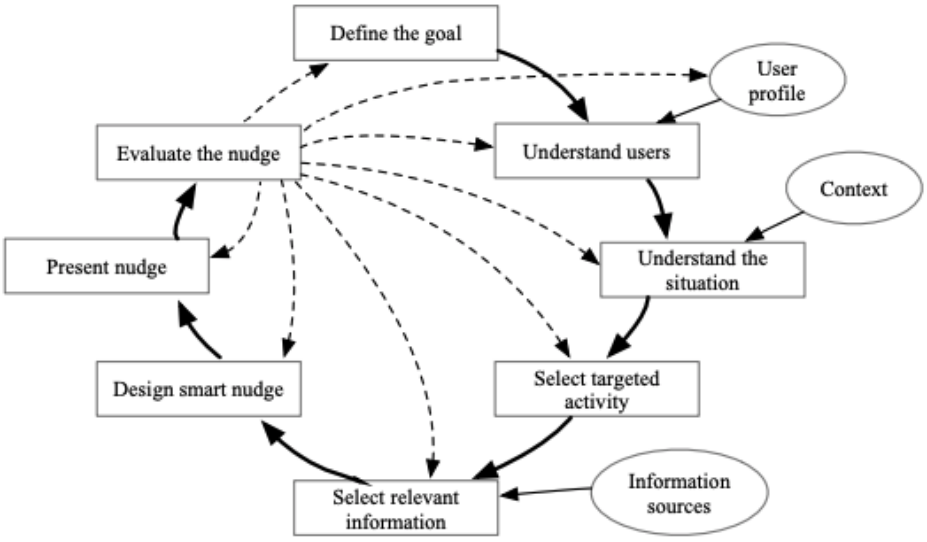


Figure 2.1: Designing process of creating a smart nudge[16].

2.3 Previous Work - Capstone Project

The ODS group is a research group at the Department of Computer Science at the Arctic University of Tromsø (UiT). The group has worked with many different projects in the domains of environment-friendly transportation, IOT, personalization, and privacy-aware computation. The research group has published multiple master's thesis and articles[16]–[19] surrounding smart nudging. The main focus for these papers have been on green transportation, and the research group is starting to do more research on how smart nudging can be applied in the health domain.

The capstone project[8] focused on finding relevant sources that could be used for a system that focuses on smart nudging in the physical health domain. In the research, a semi-structured interview was conducted. The goal of the interview was to collect data about what information people need to be more physically active, and what affects their motivation towards physical activity. The set of questions for the interview gave information about how physically active the respondent was in general, what information that was needed to engage in a specific activity, and other motivational factors. The interview was conducted on five people, where the difference in age ranged from young adults to elderly, and how physically active the respondents were, ranged from very limited to very active.

The data collected from the semi-structured interview was analyzed through the use of a thematic analysis[20]. The data from the interview was generated code for, and patterns between the different codes were discovered to find different themes that are broader topics about the things that affect people's motivation.

The result of the capstone project was that some data sources were found for the different themes that were discovered. The result can be seen in Table 2.1. The first column in the table shows the *Themes*, which represents types of information that people were affected by. These themes are the ones that were discovered through the thematic analysis. The second column, *Data Source*, are sources that data for that specific theme could be collected from. For some themes, there are many different data sources to choose from, so this table shows just those sources that suit this thesis the best in terms of simplicity of extracting data and documentation on how to do so. The last column, *Relevant data*, gives slightly more information on what data from these sources might be relevant. The data sources and relevant data was discovered after going through each theme and searching for different web applications, API's, and documentations for frameworks.

Table 2.1: Summary of what sources and relevant data that can support smart nudging for physical activity for each theme that got discovered in the capstone project.[8]

Theme	**Data source**	**Relevant data**
Conditions (weather)	Weather API & Meteorologisk Institutt's API	Fetch historical weather data, current weather data and forecast for the next days for the location of individual users
Conditions (ski)	Skisporet.no & IoT-sensor	Conditions of routes close to the user from Skisporet & Temperature and moisture from sensor
Time	EventKit & The Calendar Provider	Calendar Items
Information about physical health	World Health Organization	Pros and cons about physical health
Route suggestion	UT.no & Google Maps	Fetch information about routes close to a user from the map service & Give route suggestions from the user to the point of interest
Point of interest	Places API(Google Maps)	Relevant point of interests based on a query and geographical position
Progression	Smart devices & Health record API's	Data from motion sensors like gyroscope, accelerometer, and GPS & Step count and sleep history from health records

2.4 Smart Devices

A smart device is an electronic gadget that can connect, share, and interact with its user and other smart devices[21]. These devices can use data sent from different devices or from internal sensors to do computation. It could be a smart thermostat which receives a request to lower the temperature in a room from a user, or internal sensors which sense that the temperature is too high, and need to lower the temperature.

There exist many different types of smart devices, all from smartphones and smartwatches to smart speakers, smart doorbells, smart thermostats, and smart key chains. In this assignment, the research is limited to wearable smart devices like smartphones and smartwatches, as these devices are with people throughout the day, and therefore, can monitor people's activity better.

2.4.1 Smartphone

Regular phones are limited in the sense that it can mainly be used to make calls or send text messages. What separates a regular phone from a smartphone is the endless possibilities of usage for a smartphone. This is because smartphones have faster hardware, better storage capabilities, much more complex operating system and software that can connect to the internet and use other communication protocols like Bluetooth, Wi-Fi, and satellite navigation[22]. People can do things like measure the length of different objects with the use of augmented reality[23], and ask virtual assistants to do tasks like calculating a math problem, send money, and remember where people have parked in the past[24], [25].

Smart devices have been throughout the years filled with sensors that can be used by the software. Newer smartphones have a lot of sensors. The developer page for Android[26] categorizes the sensors into three different types. These are Motion Sensors, Position Sensors, and Environment Sensors. Table 2.2 gives an overview of all the available sensors from Android.

Table 2.2: This table shows the available sensors and type from the developer page from Android[26].

Sensor	Sensor Type
Accelerometer	Motion Sensor
Ambient Temperature	Environment Sensor
Gravity	Motion Sensor
Gyroscope	Motion Sensor
Light Level	Environment Sensor
Magnetic Field	Position Sensor
Orientation	Position Sensor
Pressure	Environment Sensor
Humidity	Environment Sensor
Device Temperature	Environment Sensor
Step Counter	Motion Sensor

1. **Accelerometer**: The accelerometer shows the acceleration(m/s^2) of the device in a 3-dimensional space(x, y, z). This is illustrated in Figure 2.2a.

2. **Ambient Temperature**: This sensor measures the ambient room temperature.

3. **Gravity**: The sensor for gravity measures the applied force of gravity(m/s^2) to the device in a 3-dimensional space.

4. **Gyroscope**: The gyroscope measures the rate of rotation around each of the three dimensions. This is calculated as $radians/s$. This is illustrated in Figure 2.2b. *Roll*, *yaw* and *pitch* are different names for the axes, which is normally used to measure the rotation of aircrafts.

5. **Light Level**: Measures the ambient light in lux.

6. **Magnetic Field**: Measures the geomagnetic field for all of the three dimensions. This can be used for creating a compass.

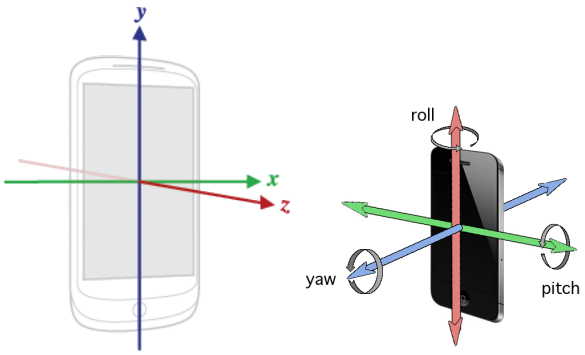
7. **Orientation**: The orientation sensor looks at the degree of rotation for each of the three dimensions of the device. This can be used to look at the device position, and which way it is angled towards.

8. **Pressure**: The Pressure sensor measures the ambient air pressure. This can be used to look at the change of air pressure(e.g., determine if people are moving up a mountain).

9. **Humidity**: Measures ambient humidity in the air.

10. **Device Temperature**: The device temperature sensor looks at the temperature of the device.

11. **Step Counter**: Measures the number of steps taken by the user of the device.



(a) Accelerometer[26] **(b)** Gyroscope[27]

Figure 2.2: Illustrations of some of the sensors available.

2.4.2 Smartwatch

A smartwatch is comparably similar to a smartphone, but it is made to be worn on the wrist. Therefore, the hardware is not as fast and might not have as much storage as in a smartphone because of the size restriction, and the number of sensors is usually lower as well. In 2019, the International Data Corporation (IDC)[28], posted an article reporting that 198.5 million units of *wearable* smart-devices are predicted to be made by the end of 2019. Smartwatches stood for approximately 44% of the units in 2018[28].

Most smartwatches offer many of the features which are typically seen in a smartphone. Smartwatches can give notifications towards activities or events, which could be notifications that are mirrored from a paired smartphone or specific notifications for the smartwatch. An example of this is the fall detection, which is supported by Apple Watch, which is the smartwatch made by Apple[29]. This feature detects whether or not the wearer has fallen, and if it detects a fall, it will notify the wearer and ask if they are okay. If the wearer does not answer, the smartwatch can alert authorities. Smartwatches also have support for applications. Some have general support for various types of applications from all sorts of domains like fitness, sleep, cooking, or parking. In contrast, other smartwatches, like Fitbit[30], have a limited set of applications that are

focused on fitness.

Most smartwatches today have sensors like accelerometer, gyroscope, ambient light sensors, and sensors for monitoring heart rate. The Apple Watch Series 4 has all of these sensors, including a pressure sensor that is used to calculate the barometric altitude, and through electrical and optical heart sensors able to monitor heart rate and even Electrocardiography (ECG) monitoring[31].

Fitbit is known for making smartwatches for monitoring physical health, and they came out with a new model called Fitbit Versa 2[1] in September 2019. This model has many features, and here are some of them[32]:

1. Monitors people while sleeping by tracking light, deep and Rapid Eye Movement (REM) sleep, and analyses the quality of the sleep to give a sleep score.

2. Monitors heart rate.

3. Users can set goals, which are notified by the watch, and can show how close users are to completing their goal.

4. Shows real-time pace and distance by using the Global Positioning System (GPS) of a paired smartphone.

5. Has multiple workouts which can be followed on-screen

6. Monitors steps taken, floors climbed and calculate calories burned

The Fitbit Versa 2 is equipped with multiple sensors. These sensors are a 3-axis accelerometer, an altimeter, an optical heart-rate tracker, an ambient light sensor, a microphone, and a vibration motor[33]. The steps taken are measured using the 3-axis accelerometer. By looking at this sensor, the device can also determine the duration, intensity, and frequency of the movement. By having the information about steps taken, the device can calculate the distance traveled as well. This is calculated by the equation:

$$Steps \cdot Stride\ Length = Distance\ Traveled$$

where *Stride Length* is based on sex and height[2].

1. https://www.fitbit.com/versa
2. `https://help.fitbit.com/articles/en_US/Help_article/1141`

2.5 General Data Protection Regulation

The European Parliament agreed upon a new primary law on the protection and privacy of the citizens in the European Union (EU), called General Data Protection Regulation (GDPR). This new law got applied in May 2018. Companies that do not comply with this law will risk getting penalties and fines. As a developer, this affects the development of new systems, as new laws have to be taken into account. Some of the most important articles and paragraphs of the regulation are listed below that might affect the development. These are all collected from the official regulation[34].

- **Right of access by the data subject**: Article 15(1) says that "*the data subject shall have the right to obtain from the controller confirmation as to whether or not personal data concerning him or her are being processed*"[34].

- **Data protection by design and by default**: Article 25 says that the controller needs to use appropriate technical measures to ensure that the personal data is being protected for the data subject.

- **Transparent information, communication and modalities for the exercise of the rights of the data subject**: In Article 12 the regulation states that the controller shall act transparent, meaning that the controller shall inform the data subject about the collection of personal data, and inform them about any associated privacy policies. Information shall be presented in a concise and transparent way.

- **Right to erasure**: Article 17 is about the right to be forgotten. This simply means that the data subject shall have the right to obligate the controller to erase personal data about the data subject without undue delay.

- **Right to object**: Article 21(1) states that "*The data subject shall have the right to object, on grounds relating to his or her particular situation, at any time to processing of personal data concerning him or her which is based on point (e) or (f) of Article 6(1), including profiling based on those provisions*"[34]. Profiling could include geographical tracking, demographic, or behavioral. This regulation states that the data subject should have the ability to opt-out of being profiled.

2.6 Microservices

Sam Newman published in 2015 a book called *Building Microservices: Designing Fine-Grained Systems*[35] where he defined that "*microservices are small autonomous services that work together*"[35]. Many monolithic systems have multiple services under the same codebase, which means that the system will grow in complexity relatively fast when adding more services. With the use of microservices, each service is an isolated service that is autonomous, which means that the services are independent of each other. Each service uses an API protocol to communicate with each other services, usually through the use of Hypertext Transfer Protocol (HTTP). The use case for microservices is when the application that is being developt is rather complex and needs to be agile. If the application is not complex and does not need to change much over time, the microservice architecture is not the right fit.

Newman mentions some of the key benefits of choosing a microservice architecture instead of a monolithic architecture[35], and these are:

1. **Technology Heterogeneity**: By splitting up each individual service, there become opportunities to have mixed technology stacks. The service can use any language and technology, as long as the service has endpoints for clients to send requests to, and the response data is sent in the right format. Therefore, the service can pick any technology or language based on the requirements and goals for that individual service. This might be a service that needs to be scalable or needs a framework for working with machine learning.

2. **Resilience**: Fault isolation is an excellent benefit of the microservice architecture. If a service fails, the rest of the system can carry on without it. This is a problem for monolithic services as if a service fails, the rest of the system will stop working as well.

3. **Scaling**: Scaling monolithic systems are often done inefficiently. It is because many monolithic systems have a specific service as a bottleneck. When scaling is needed to balance the traffic for that service, the whole system needs to be scaled, as the single service alone can not be scaled up as it is part of a bigger monolithic system. With microservices, scaling can be targeted for that individual service, as this is an autonomous server. Therefore, services that need to handle more requests, or need better hardware can run on stronger hardware, while services with less traffic can run on less powerful hardware.

4. **Ease of Deployment**: As microservices are small independent systems, development on a microservice is also independent from the other ser-

vices. When working on a monolithic codebase, changes may happen to multiple services before they are merged together into the code that is used in production. This may lead to trouble when releasing new versions of the application. In a microservice application, the service can deploy new versions independently from the other services. If bugs occur in a microservice, a simple rollback to a stable state of the service can be done, as these microservices are isolated.

5. **Organizational Alignment**: Working on large codebases can be difficult. The productivity tends to be faster with small teams working on small codebases, than the opposite. By splitting up each service to its own codebase and each its individual task, the teams can be smaller, and the codebase is minimized to only support that service.

6. **Optimizing for Replaceability**: As new and better technology arises, the need to replace old services follows. With a monolithic system, this is limited to new technology that fits just that framework and language. With the use of individual services, replacing them is relatively simple. Even removing microservices from the system can be done without too much of a complication.

2.7 Tools and Frameworks

2.7.1 React Native

React Native is a framework for making native applications for iOS and Android devices using React, a library created by Facebook[3] for building web applications in JavaScript. The framework makes it possible to develop applications using one language, with one code base, instead of implementing two separate mobile apps, for iOS and Android, using Swift and Java/Kotlin with the same logic. This is possible because the framework will make a build for both iOS and Android. Making a plain text element in HyperText Markup Language (HTML) can be done by writing `<div>Hello World</div>`. In a normal React web application, this would be valid, as HTML is the standard, but in React Native this is done by writing `<Text>Hello World</Text>`. When this code gets built, the version for iOS will have `<UIView>Hello World</UIView>`, as `UIView` is the default tag to write text in Swift, while the build for Android will have `TextView`, as it is the standard for Java/Kotlin.

3. https://reactjs.org/

2.7.2 Expo - A Toolchain for React Native

Setting up projects can be complicated sometimes. React Native has many dependencies that need to be installed before the project can be up and running. Expo is a toolchain that makes installation, development, and testing with React Native simpler. Expo sets up the project on its own without having to pre-install other dependencies. Development can be difficult when developing mobile applications if the development is done using a device simulator. Many bugs will not be discovered on a simulator, as there is a lot of different hardware that will run the application, which can make a difference. Expo has developed a mobile app that allows the developer to install a build of the application on their smartphone. It is very useful, as the application can be run on real hardware and tested there. The app can also be shared with others for review and more testing.

2.7.3 Express.js

Express.js, or simply Express, is a framework for building server-side software. Express is a framework built on top of Node.js, a runtime environment for executing JavaScript outside of the browser. Express is mainly used for hosting servers through communication over HTTP, and many modules like parsing payload, cookies, and setting up routes are included in the framework. For this thesis, Express is mainly be used to host API servers for the microservice architecture.

2.7.4 MongoDB

MongoDB is a document-based database and uses a non-structural query language, which means that it does not use tables, rows, or columns, as relational Structured Query Language (SQL) databases do. What MongoDB uses instead is a document-based model that consists of collections and documents. A document can be seen as a row in a SQL table, which contains a set of key-value pairs, and a collection can be seen as a table in the database.

What makes MongoDB unique from relational SQL databases is that the data is stored in a document, which uses a format similar to JavaScript Object Notation (JSON), and the documents in a collection do not need to contain the same keys(columns in SQL databases). Also, there is no static schema for documents. Therefore, various documents in a collection can have different fields and types, but the regular use case is that the documents in a collection follow almost the same structure. The benefit of having a dynamic schema is that if new data becomes relevant, it can be added to a collection without

any problem. MongoDB is a NoSQL database, and the database can be used in applications that need a scalable database as MongoDB can scale horizontally. MongoDB has high performance for simple queries and is flexible in the sense that new columns or fields can easily be added when needed.

To be able to manage and set up a MongoDB database in JavaScript, a library called Mongoose can be used. The library can be installed for Node.js, and can be used in every part of the database, from initializing, setting up schemas with type validations, to every Create, Read, Update, and Delete (CRUD) operation.

3

Related Work

For finding related work, a Structured Litterature Review (SLR) was conducted. Anders Kofod-Petersen wrote a paper called *How to do a Structured Literature Review in computer science*[36]. In this section, ideas from his writing are applied to be able to have a higher likelihood to find relevant work, while still limiting the search results to an amount that is possible to review.

The first step of the SLR is the *planning phase*. In this phase, it is important to identify the research question of the thesis, what sources should be used to identify research and create a search term based on the research question.

Research Question: *How can data be extracted from relevant sources and used to create smart nudges by a system to influence people to increase their physical activity?*

Sources to be searched: For this research, three sources were picked to search for related work. This is IEEE Xplore Digital Library[1], ACM Digital Library[2] and SpringerLink[3].

Search Terms: Table 3.1 shows the words that is used as part of the search term. To construct the search term, the logic is that the result should include

1. https://ieeexplore.ieee.org/
2. https://dl.acm.org/
3. https://link.springer.com/

one term from each of the groups. A formula for this can be explained as:

Should include at least one term for a specific group, G_i:

G_i = (Term 1, G_i) $\vee$ (Term 2, G_i) $\vee \cdots \vee$ (Term n, G_i), where n = number of terms in the group.

Should include a term from each of the groups:

Search Term = $G_1 \wedge G_2 \wedge \cdots \wedge G_n$, where n = number of groups.

By using these formulas, and the search terms in Table 3.1, the search term looks like this:

("Digital Nudge" $\vee$ "Persuasive system") $\wedge$ ("Data Source")

Table 3.1: Search terms for SLR.

	Group 1	**Group 2**
Term 1	Digital Nudge	Data Source
Term 2	Persuasive system	

The next step of the SLR is to select the primary studies. The search results were 8 articles from IEEE, 6 articles from ACM, and 0 articles from SpringerLink, which are 14 articles in total. To find the primary studies from these results, three stages of filtering was done. The first stage was to filter out the irrelevant articles by reading both titles and abstracts for all of the articles. The second step was to read the introduction and conclusion of the articles that still looked relevant. The last stage was to do a complete reading of the remaining articles. This resulted in finding two articles that could be relevant to the research question. These two articles are explained in more detail in the next subsections.

3.1 PC4HC: Personalized communication for health care

PC4HC is a paper that explains a communication platform for people with multimorbidity and healthcare facilities that manages the care process[37]. The system collects data from different sources, which is analyzed and used to help the doctor by enriching communication and find new knowledge through

analyzing big data. The platform also helps the patients by influencing their habits and lifestyle choices, which can positively affect their health.

The idea is that the data comes from three different places; the *patient, doctor* and *healthcare facilities*. From the *patient*, sensor data can be extracted from different types of devices. It could be smartphones, smart weights, and other devices that can extract relevant information based on the patient's health problems. Other data that comes from the patient is profile data that the user fills in upon registration, feedback from the patient, and requests. From the *doctor*, configurations and schedules are collected, as well as requests from the doctor. The *healthcare systems* sends data from dialysis machines and system configuration data.

The system will analyze the data extracted from these three sources and give reports back to the doctor, which can give the patient feedback from the reports. The system has multiple types of outputs that are given at the end to the patients. It could be that some parameter values from sensors and dialysis machines are out of range, motivational messages, feedback requests from doctors, report on trends for measurements and post-treatment. These outputs are meant to influence patients towards bettering their health and give them more knowledge about their health problems.

There are a couple of differences between this paper, and their work, compared to the work and idea of this master's thesis. First, this thesis focuses on the domain of physical activity in general. Therefore, the collected data needs to come from many different types of sources, as more activities and relevant data are needed. At the same time, PC4HC has a more specific focus of people with multimorbidity and communication between them and the doctor. Secondly, it seems like the doctor tries to influence the users *through* the use the system. In contrast, the system design for this thesis focuses more on how the system can independently affect the users.

3.2 Let them play: Experiences in the wild with a gamification and coaching system for young diabetes patients

This paper[38] explains a system that supports people with type 1 diabetes in self-management through gaming, monitoring, and coaching. This system is targeted for children and teenagers, and the game is about an investigator, which also has type 1 diabetes who tries to solve some cases. To progress in the game, the users need to control the investigator's diabetes through their

own diabetes and solve puzzles.

Some of the tasks that the investigator has to do are:

- Measure blood glucose. It has to be written down in the game, which makes the users learn more about how to diet and their glucose level. This data is stored in the system so it can be analyzed and gives notification if values are wrong, or measurements has not been given.

- Daily goal of steps taken. The game collects sensor data about the steps taken for the users, as another task is to have a daily goal of how many steps the user wants to take.

- The game has small tasks, which could be to fill out profile information or install an application that is smart to have when you have diabetes or watch an educational video.

The game has a virtual coach that tries to personalize the game experience by giving suggestions on tasks and feedback about the daily progress. When tasks are completed, points are given to the user. It could be knowledge points, experience points, or evolution points. The user can not continue the game without gaining points in these three types. As this is targeted for younger users, the use of gamification could be useful to keep the users motivated to gain knowledge about their self-management.

Both, *Let them play*, and this thesis focuses on influencing people, but *Let them play* focuses on influencing people to better their self-management with diabetes. In contrast, the master's thesis focuses on influencing people to better physical health. The main difference seems to be that the system of *Let them play* only collects data from the user, and not from any third-party sources, and does not seem to be an adaptive system that changes the goal or methods based on the behavior or environment of the users.

4

Methods and Methodologies

To answer the research question given in Section 1.4, a few methods were applied. The master's thesis used applied research, data gathering through implementation, Structured Litterature Review (SLR) and looked at the relations between the collected data to combine them into a smart nudge.

This thesis used applied research by looking at the themes and relevant data sources from previous work, which is illustrated in Table 2.1, and extracted the relevant data from these sources through implementation. Some of these data sources have API's to fetch data from, some required software to collect sensor data from devices, while others needed the use of Web Scraping to obtain the relevant information. Either way, different techniques have been used to fetch out data from these sources through the implementation of demonstrators.

Some of the sources can singularly be used to form a smart nudge or analyzed to learn more about people's behavior. In contrast, other data sources might need to be combined with other sources to create a relevant smart nudge. By collecting data from the relevant sources from Table 2.1, a more in-depth exploration was done to the data sources. These sources, and their data, needed to be combined into a system that uses the data to create smart nudges. A SLR was applied to gain more knowledge on how other systems handled data, what

data they extracted, and how others designed their system.

This master's thesis analyzed the collected data by looking at the patterns between the different sources and how they could be combined to form a more personalized nudge to individual users. In the previous work, a thematic mind map was made for each of the interviews conducted. This master's thesis created a somewhat similar map, where the different sources and what type of data that was collected from these was used to form links between the sources to create different smart nudges. The result of looking at the patterns between the data collected was to combine data that the system could give, and form smart nudges that are relevant for increasing people's physical activity.

5 Design

In this chapter, the overall architectural design for the smart nudge system is presented, and how the system can be *gdpr* compliant. Each microservices within the system is explained in more detail in their own section.

5.1 Nudge Architecture with Microservices

The overall design choice of this system was the use of a microservice architecture. Both, Figure 5.1 and 5.2, illustrated the design of the system. The figures show two different flows of how data is moved in the system, when the system receives data from the client, and when the system sends smart nudges to the client. The difference will be explained in more detail after the general architecture is explained.

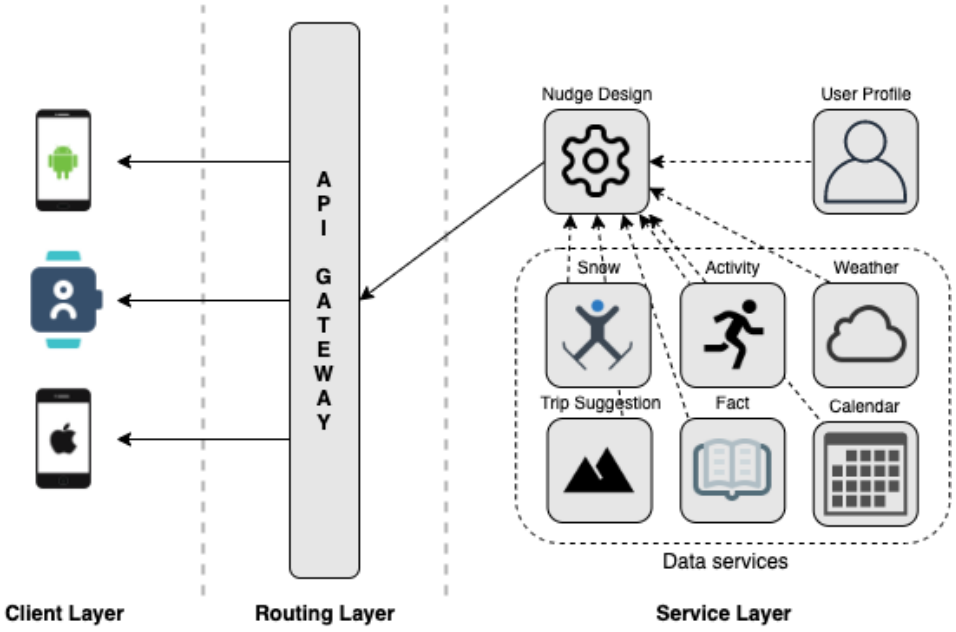


Figure 5.1: The microservice architecture of the nudge system. The figure shows how the system sends nudges to the clients. Nudges from the system is sent from the Nudge Design Service.

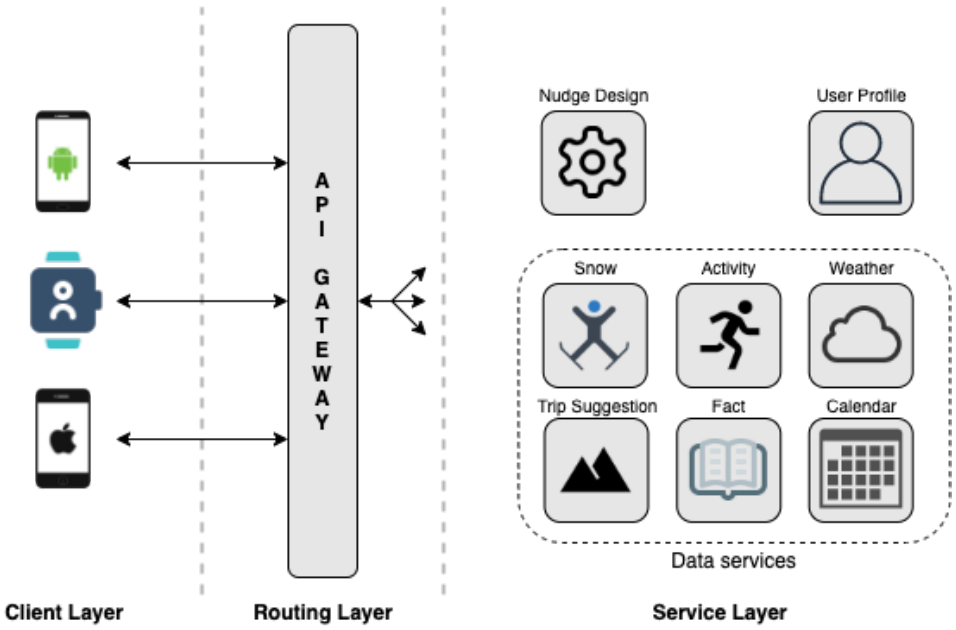


Figure 5.2: The microservice architecture of the nudge system. The figure shows how the client sends data to the system. The data is sent directly to the service that the data is meant for.

The system consists of three different layers, where the left layer is the *Client Layer*, the middle layer is the *Routing Layer*, and the right layer is the *Service Layer*. Clients can access the system through different types of devices. It can mainly be smartphones and smartwatches. These devices will access the system through a mobile application. With the use of an application, sensor data and data from other applications on the mobile device can be extracted, which the system could use to personalize the nudges. The app can be an iOS or Android mobile application, and possibly an app for smartwatches, as illustrated in the system architecture. Data from the client that are relevant for the microservices are calendar data, user input data for the user profile, and sensor data, such as data from the gyroscope, accelerometer, pedometer, and geographical location.

The Routing Layer, as the illustration in Figure 5.1 and 5.2 shows, consists only of an API gateway. The gateway is responsible for routing requests from the client to the appropriate services or the other way around. All data that is sent between the different components of the nudge system is sent in JavaScript Object Notation (JSON) format. It will make communication easier between the different components as each component knows what format the data is coming in, and how to extract the data from the object. There are three different requests that the gateway mainly needs to handle. The first is sending data from the client to the appropriate services. The second is fetching data that the user requests form the service that holds that data. The third is sending the smart nudge from the Nudge Design Service to the appropriate user.

Sensor data from a motion sensor that the client application has collected could be relevant for a service that analyses movement. The client sends the sensor data in a JSON object, which has an extra key in the object, identifying what type of data is being sent. The API gateway will have a list of data types that the gateway expects to receive, and each data type will have a list of services that are interested in that type of data. Therefore, if the API gateway receives data from the client with a data type of `'motion data'`, the API gateway will send the data to all services that are on the list of that specific data type. The same logic is applied when clients request data from the system. The client can ask the gateway for `'weather data'`, and the gateway will know which service to collect weather data from, by looking at the list.

This thesis only focuses on how this system can be used with wearable smart devices and not computers. However, the system can adapt to these by having multiple gateways, a mobile application gateway, a web application gateway, or a third-party gateway. The reason for separating between web applications and mobile applications is because the devices that run web applications or mobile applications differ in terms of hardware, network performance, and network stability. Having a third-party gateway also opens up the possibility

for other systems benefiting from the system. These gateways can also help with security and authorization, in terms of if a client should have permission to access these services or not.

The last layer in the illustration is the Service Layer. It is the collection of all the different microservices. If a service needs access to data from another service, it has to request that data from the other service instead of having access to it through a common database. Currently, the system consists of 3 different types of microservices; *Data Services*, a *Design Service* and a *User Profile Service*. *The Data Services*(i.e Fact Service, Weather Service, and Trip Suggestion Service) collects data from different sources and sends out the relevant data from it when requested. The data collected from the services can come from clients, IOT devices, or third-party sources. The *Design Service* is responsible for designing smart nudges and does this by requesting data from other services and uses these to create the smart nudge to a client. The *User Profile Service* stores general user data from the clients and analyses the trends and behavior of the users to determine what type of nudges and activities most likely affect the users.

As mentioned earlier, the system acts in two different modes depending on who sends the request, the user, or the services. Figure 5.2 shows how communication happens if a user of the system is requesting data or sending data to the system. When the user sends a request, the request goes to the API gateway and directly to the service/services that the request is meant for. Figure 5.1 illustrates how the system sends smart nudges out to the clients. There is only one service that is responsible for creating and sending out nudges to the clients, and that is the Nudge Design Service. Therefore, the Nudge Design Service collects the data needed from the other Data Services and generates the nudge, which then is sent to the API gateway that forwards the nudge to the appropriate client.

5.1.1 GDPR Compliance

With the architectural design choice suggested in Section 5.1, many different types of data will be extracted from the user. Sensor data, calendar data, user data, and maybe more as new services will become relevant as research continues. Some data will be considered sensitive, which needs to be protected. Building a system that focuses on personalizing nudges around physical activity requires some design choices around the protection of the user and its data to be compliant with the General Data Protection Regulation (GDPR). It is because much of the data that is considered relevant for the system is personal data.

In Section 2.5, articles, and paragraphs from the GDPR was mentioned. The articles and paragraphs that were mentioned in that section were the most relevant articles and paragraphs for a system developer to consider when building a system. This section will explain how this system will comply with each of these rights.

1. **Data protection by design and by default**: Some of the Data Services of the system stores data in their local database, and data that is considered personal, should be encrypted between all the different layers of the system, and only accessible for a single user and the system. It can be achievable by having the API gateway dealing with the authorization and key exchange for encryption.

   First, data can only be accessed by clients who are authorized in the form of login with a verified email address or phone number, and the data that the authorized user can access or modify is just its own data. If the authorization goes well, from the API gateway, the request gets forwarded to the right service, which will check that the data requested is for the same user as the request comes from.

   Secondly, to safely send data from the client to a service, or vice versa, the clients, the API gateway, and the microservices all need to generate a public and a private key for asymmetrical encryption. The API gateway has a list of public keys for each of the services that the system contains, and each client has the public key of the API gateway. If a client is sending data to a service, the data gets encrypted by the client using the public key of the gateway. When the gateway receives the request, it will decrypt the ciphertext, and then know what service to send it to, as the data type is included in the object, and the gateway knows what microservices that is interested in that data type. The gateway will encrypt the data with the public key of that service, and send it to the service, which will then be able to decrypt it with its own private key. The same logic goes the other way around when a service wants to send data to the client.

2. **Right of access by the data subject**: The clients have the right to receive all data that is being stored about them. The API gateway should have a route that the clients can access, that will make the gateway send a broadcast to all the services that store data, to send the data about a specific client back to the gateway. The gateway will compress the data and send it to the email address that is registered to the client, as the amount of data could be significant.

3. **Transparent information, communication, and modalities for the exercise of the rights of the data subject**: The system needs to be

transparent about how the system works, and what data that is being collected from the user. Client-side, the information about how the system works should be clear, and the users should be able to see what type of data is being extracted and how it is used.

4. **Right to erasure**: The clients have the right to erase all data that is linked to them. It can be done in the same manner as mentioned about clients receiving all data collected about them. Design an endpoint in the API gateway, which triggers the gateway to broadcast a request to all services that store data, to delete all data that is stored about a specific user. One important difference is that the services that send the data to a third-party application need to notify them as well, so they can delete data that might be stored about that user.

5. **Right to object**: Clients should have the right to choose whether or not to be profiled. It means that the users should have a simple way to decide, client-side, what data is being sent to the system. If the user does not want geographical coordinates to be sent to the system, this should be a choice the user can opt-out of.

5.2 User Profile Service

The User Profile Service is the service that is responsible for storing and updating the user profiles of the system. It is important to have user profiles, as this can help massively to personalize nudges and analyze the effect of previous nudges, to adjust nudges for the future.

The idea for this system is that the Nudge Design Service gets information from the user profile to help create nudges that are suited for the specific user that the smart nudge is meant for. The information a user profile hold is metadata, which includes activities with priority ratings, tags of what type of nudges that affect the user, health info, and other information that could affect the personalizing of nudges. The user profile will learn more about its user the longer the system is being used, as values in the user profile will follow and adjust to the user's behavior. A user might have a bad period, and the user profile has the job to adapt to this situation to set the bar lower for this period. As the nudge design process depends on the user profile, these adjustments will be reflected in the nudge.

5.3 Nudge Design Service

The Nudge Design Service is responsible for creating the smart nudges and sending them out to the users. The service is not responsible for knowing when to nudge users, but to go through the design process. Another service will activate the generation process, and this could be the Calendar Service, or the User Profile Service, which has more information on when users are available and most acceptant to a nudge. The nudge generation model that is suggested for this system is designed in the master's thesis *Designing Dynamic and Personalized Nudges*[18] by Sandor Dalecke.

Sandor Dalecke writes that a nudge can be built up of three different parts; *effect*, *content* and *incentive*. In the generation, a greeting and a goal of the nudge is set. An effect, content, or incentive is picked and iterated until a threshold of parts is collected. In his master's thesis, Sandor Dalecke gave some examples of how the effects, contents, and incentives could look like in the form of a list[18]. A small collection of this list is shown here:

Effect:

- **Commitment**: Which fits right into your schedule.
- **Loss Aversion**: Using the car instead of *goal* will cost you *difference*.

Content:

- **Weather**: The weather is *quality score* today, which is perfect for *goal*.
- **Route**: I have looked up a route for you.

Incentive:

- **Health**: *goal* helps to keep you fit and healthy.
- **Places**: You can spend some time at *places* and relax.

After the iteration is done, multiple parts of the effects, contents, or incentives are collected. They can be assembled into a nudge, which will be sent to the API gateway, which will forward the nudge to the right user. An example of how the generation process can look like is:

- **Greeting**: Hello Lars.
- **Goal**: Go for a walk.
- **Incentive**: **Places**: You should spend some time at Skihytta, Tromsø, and relax.

- **Content**: **Weather**: The weather is sunny today, which is perfect for taking a walk.
- **Content**: **Route**: I have looked up a route for you.
- **Effect**: **Commitment**: Which fits right into your schedule.
- **Result**: Hello Lars. You should spend some time at Skihytta, Tromsø, and relax. The weather is sunny today, which is perfect for taking a walk. I have looked up a route for you, which fits right into your schedule.

The Nudge Design Service will hold different types of templates that can be used to construct smart nudges. A template for nudging someone to do an activity could be one, or a template for nudging information could be another. Each template will have a set of *effects*, *contents* and *incentives* which fits that template. Some *contents* and *incentives* are dependent on data from other data services, and therefore, the Nudge Design Service has to make requests to these microservices to collect the needed data. When all the required data is collected, the smart nudge can be made and sent to the API gateway, which will forward the smart nudge to the user.

5.4 Fact Service

Some people lack knowledge of the importance of having a good physical active lifestyle, or simple ways to better their physical health. There are many positive consequences when it comes to being physically active. Some of these were mentioned in the Introduction. The Fact service stores facts about physical activity from reliable sources that can be used to spread awareness and knowledge about the consequences of physical activity. The Fact Service consists of a back-end application, which serves as a API server. This microservice is illustrated in Figure5.3.

The creation of fact data will be on the mobile application of the nudge system, and the information needs to come from reliable sources. Therefore, the creation can only be done by administrators of the system or users with special privileges, so incorrect information is not spread. The page for creating facts on the mobile application can only be accessible for these types of users. However, it can also be made as a separate application.

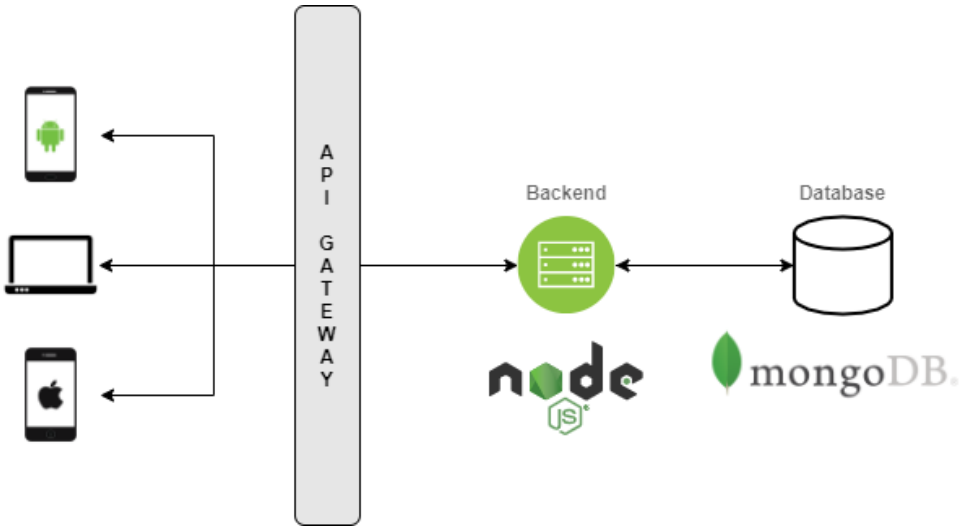


Figure 5.3: Architecture of the Fact Service. An API server that collects facts from privileged users, and stores them in a MongoDB database.

Some people are more affected by receiving positive information, while others might need negative information. By using tags to label information, and combining this with a user profile service, people can receive personalized information. For instance, this fact from WHO; *Globally, 1 in 4 adults is not active enough* [39] could be labeled as *Negative, Inactive, Statistic*, as it is statistics about the inactivity of adults.

The back-end application is made using the Express framework, and is a API server that supports adding and searching for facts. Data is stored as documents in a MongoDB database, and searching for facts is done by giving a list of tags which the back-end system will query the database for, and with the use of the Term Frequency-Inverse Document Frequency (TF-IDF) algorithm, return the most relevant facts. The user profile can hold data about what type of activities users are interested in, and learn what kind of information that affects their behavior the most. This information can then be used to get the correct type of facts to nudge people. Table 5.1 shows a suggestion of some tags that might be relevant to use both by the user profile, and the fact service to easier work together to find the best information for people. If a user has the labels *positive*, *statistic*, and *walking* in the user profile, this could be used to find facts with the same label.

Table 5.1: Suggestion of what tags a fact can have.

Theme	Tag
General	Positive Negative Neutral Active Inactive Statistic ...
Location	Indoor Outdoor ...
Activities	Walking Jogging Hiking Skiing Cycling ...
Health	Risk Reduce risk Disease Mortality Blood pressure Diabetes Heart ...

5.5 Weather Service

The Weather service is an API server that can give weather information around specific users. This is with the use of the geographical location of users. As we see in Figure 5.4, the service is tightly coupled with two different API's from Meteorologisk Institutt, as the information is collected from these sources.

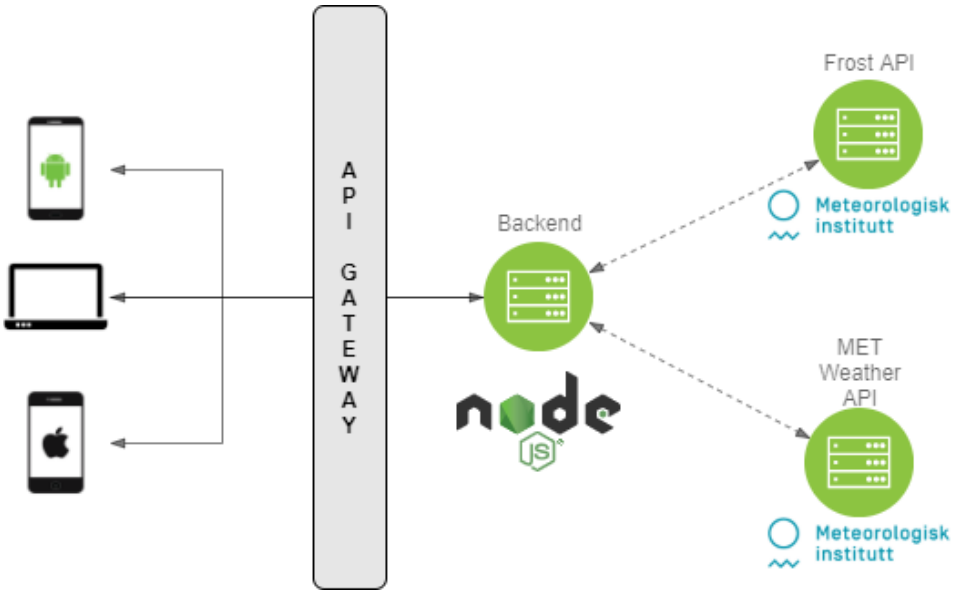


Figure 5.4: Architecture of the weather service. A back-end API server that collects data from two third-party applications from Meteorologisk Institutt.

MET Weather API [1] has lots of information for current weather and forecast for the next days. With the use of the route `/locationforecast` to their API, geographical coordinates can be added to the request. With the use of this MET Weather API, the Weather Service gives detailed weather data from the current time of the request, up to *n* days, where *n* can be a number from 0 to 10 days.

The second API that is used from Meteorologisk Institutt is Frost[2]. Frost is used to accessing historical weather data. Their API has several access points that need to be used to support historical weather data around specific users. The Weather Service needs with the use of the geographical location of a user, use this to get the ID of the closest weather sources from the route `/source`. Once the closest sources are collected, each of these sources can be fetched weather data from for a specific date through the route `/observations`. The Weather Service returns data in the form of JSON that includes the date, average temperature, and precipitation between the sources, and what sources that the data has been collected from.

1. https://api.met.no/weatherapi/
2. https://frost.met.no/index.html

5.6 Trip Suggestion Service

The Trip Suggestion Service is an API server that gives suggestions of trips dependent on the geographical location of the user. The architecture is illustrated in Figure 5.3. Clients can access the service through Hypertext Transfer Protocol (HTTP) to a Express.js back-end server with the route `/trips/:lat/:long/:distance`, where *lat* and *long* are parameters for the geographical location, and *distance* is a parameter for the radius which the service will limit the search for.

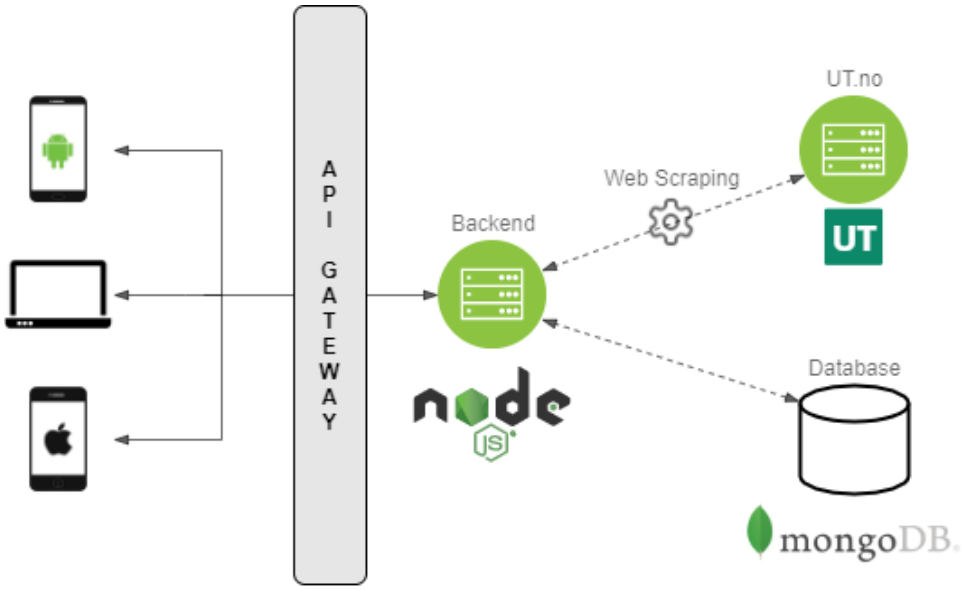


Figure 5.5: Architecture of the Trip Service. An API Server that collects data from UT.no through web scraping and stores the data in a local database.

The Trip Suggestion Service depends on having lots of data on trips all around Norway to give suggestions for users regardless of location. The Norwegian Trekking Association (DNT) has a web application called UT[3] that has a map solution, which is illustrated in Figure 5.6 with thousands of trips all across Norway, that is connected to their open API. With this API, the ID's of each trip, together with their geographical location, can be collected. The ID of each trip can be used to fetch more metadata about each trip from UT's map solution through the use of Web Scraping. This data is then stored in a MongoDB database, and when a client request trips for a user from the Trip Suggestion Service, the service will filter out the trips that are further away than the radius specified, and return the client a list of trips that are inside the radius with

3. https://ut.no/

metadata about each trip. The metadata that is included in the response is *Name of trip*, *ID*, *Description*, *Length*, *Duration* and *Type*, where *Type* can be hiking, skiing, paddling or cycling.

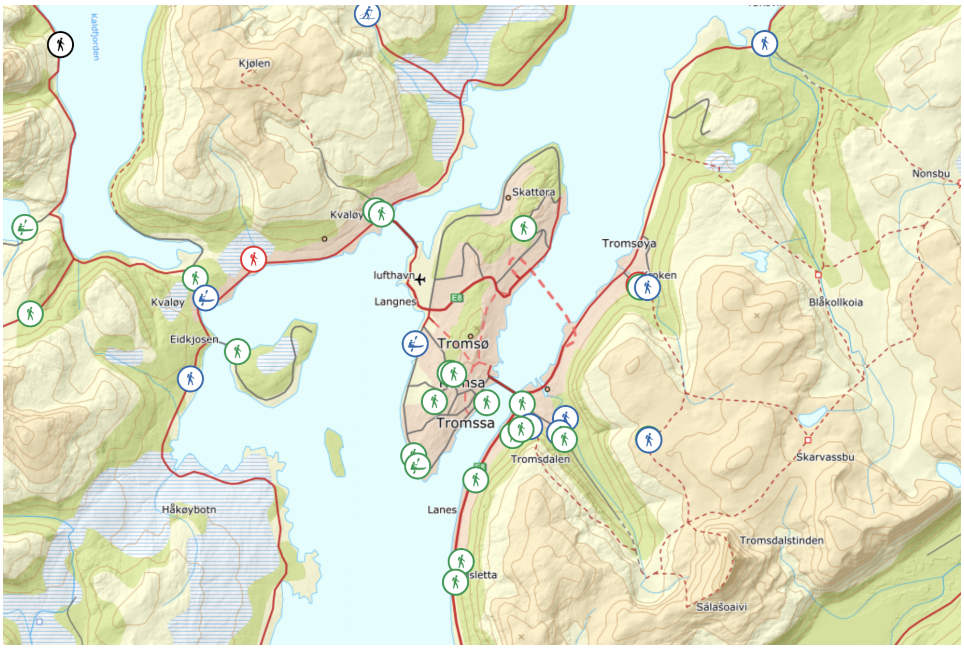


Figure 5.6: Image of how UT.no's map solution looks like. Each circle on the map shows one trip suggestion.

5.7 Calendar Service

The Calendar Service is not dependent on any third-party application for data, as the service collects calendar events from the users of the system. The architecture of the system is illustrated in Figure 5.7, where the clients send their calendar data periodically to the API gateway, which will forward the data to the back-end API server of the Calendar Service. The back-end server will find the times where the user is busy and update the user's schedule in the database.

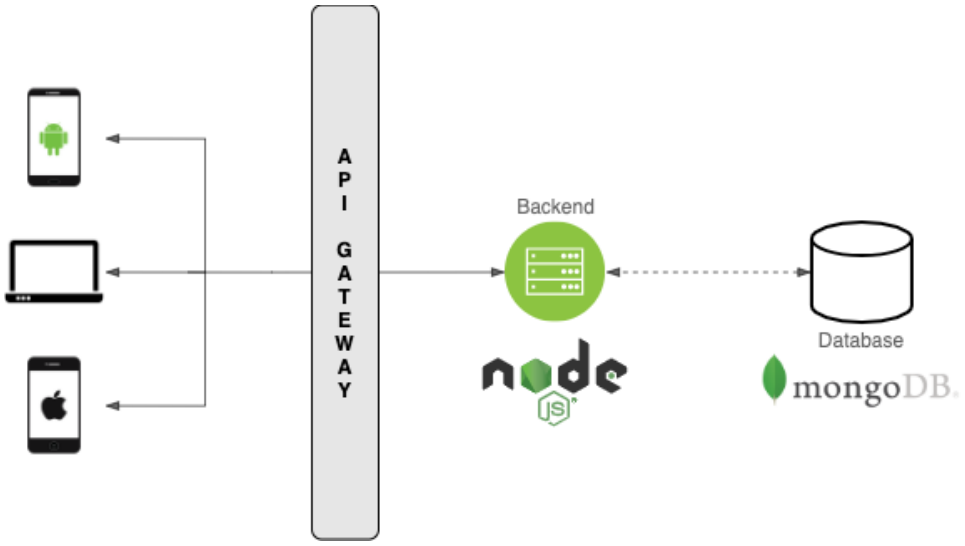


Figure 5.7: Architecture of the calendar service. Server that collects calendar data from users, and stores data about their availability.

With this service, the system can nudge users when it fits them the best. An example of this can be shown in Table 5.2. This table shows an example of how the daily schedule of a user could look like. With this information, a smart nudge could be created in the spare time between the two events for that day. Some people are more spontaneous than others, and therefore being able to not only have information about what events users have for the specific day, but the whole week could be of great interest. If the system wants to create a smart nudge to recommend a user to go on a hiking trip, some users might benefit more from having a few days to prepare, both mentally and physically, for that trip. In contrast, others could receive the nudge the same day as the system suggests the trip.

Table 5.2: Example of a calendar event for one day.

08-15	Work
15-19	-
19-21	Visiting mom and dad
21-23	Movie with Sam

5.8 Activity Service

The Activity service is responsible for storing metadata about the type of activities and amount of activity of the users. The microservice consists of a back-end

server that receives sensor data from users and analyses the data for activity detection. The architecture is illustrated in Figure 5.8. There exists both products[40] and research[41], [42] for activity detection, and the most common sensors to use is the accelerometer and gyroscope as these are motion sensors which can show patterns in rotation and acceleration in 3-dimension.

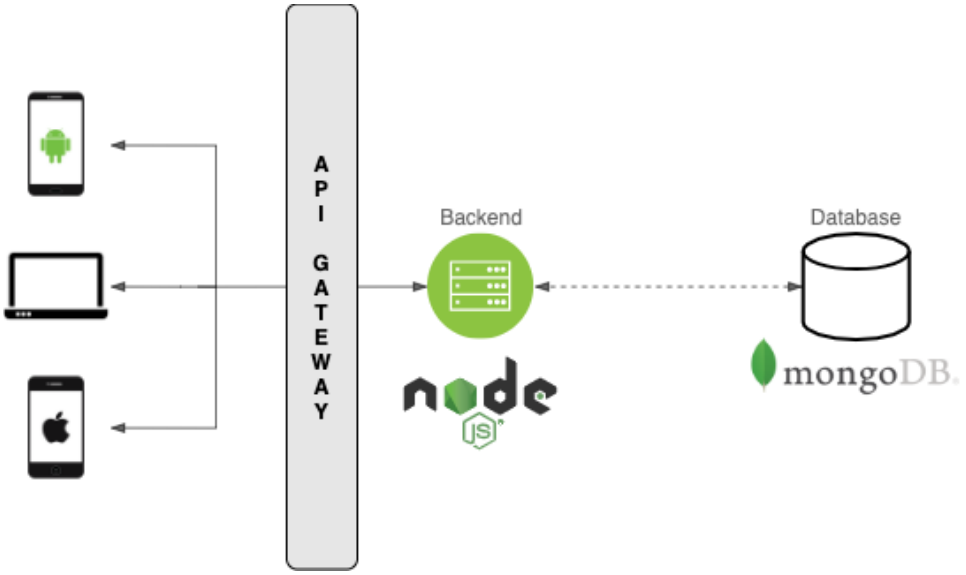


Figure 5.8: Architecture of the activity service. A back-end server that receives sensor data from users to detect the type of activity and amount.

The back-end server receives data from the accelerometer and gyroscope from the user in intervals to avoid overloading the system with requests from multiple users. The data will be split up by the server into sub-windows of even shorter intervals, which could be windows of 1 to 10 seconds, depending on what is best for predicting the activity. These windows of data will go through a machine learning classification model, which will predict the type of activity that has taken place in the window and store metadata about the kind of activities the user does and the frequency of the different types of activity. This information could be necessary for a user profile to store. What type of activities individual users use, and how often the user exercises could be used to personalize nudges towards the right activity and the right amount of activity.

To detect user activity, a classification model needs to be configured before the system is live. The classification model needs to go through a training phase where the model is being learned about the sensor data patterns through supervised data. The supervised data needs to have data from multiple different types of activities. It could be sitting, walking, running, cycling, and skiing.

This microservice also receives daily pedometer data from the users, as this data could be used to see if users have an increase or decrease in steps taken, both in the short and long run. It could be used to compare steps taken for two days in a row or compare the steps taken between weeks or months or to set a daily goal of steps to take and see how close the user is to accomplish the goal

5.9 Snow Service

The Snow Service is a service to give ski track information, and ski waxing tips. The system is illustrated in Figure 5.9, where data is collected from two different sources, IoT devices and Skisporet[4].

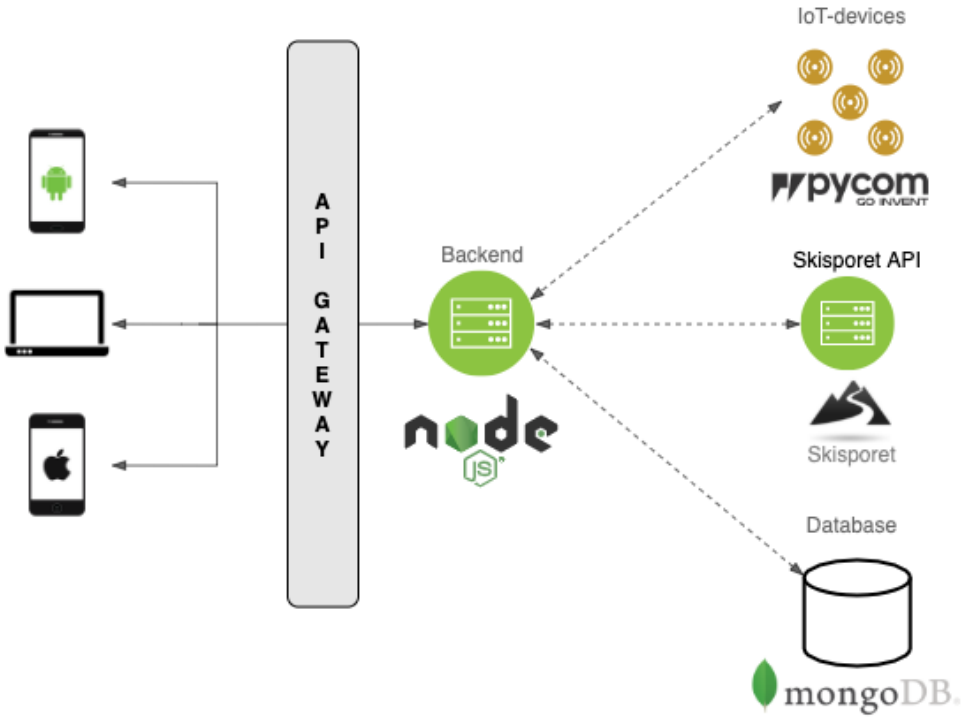


Figure 5.9: Architecture of the snow service. A back-end API server that collects data from IOT-devices, and stores it in a local database.

4. https://skisporet.no/

Skisporet has two main features that could be used by this system. The first feature is the map solution that they provide with details on ski tracks routes, with information on when the tracks last got groomed. It could be used to give users recommendations on ski tracks that are localized nearby to specific users. The data could be fetched regularly, to have the latest updates of the tracks, and store it in a local database to access the data upon requests to the system quickly.

The second feature that could be used from Skisporet is the ski waxing tips[5]. This feature lets the users give information about the conditions in the ski track; this is temperature, the graininess of the snow, humidity, and if the snow is newly fallen. With this information, the users can get ski waxing tips for those specific conditions. Håkon Wallann, which was part of the ODS group at the Arctic University of Tromsø (UiT) designed an IOT device in his master thesis[17] that measured snow height, in addition to temperature and humidity. The device can be seen in Figure 5.10. Placing these devices out in different ski tracks could give users tips on what ski wax can be used for their local ski tracks and give general recommendations of the conditions in the tracks.

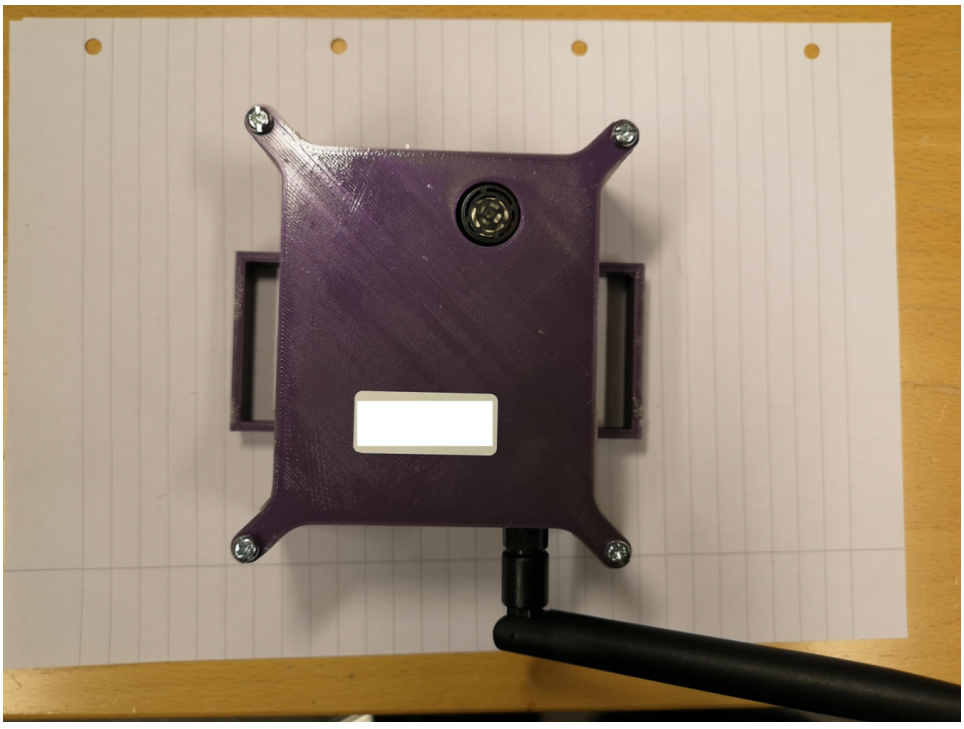


Figure 5.10: The IOT-device that was designed by Håkon Wallann to measure snow height, as well as temperature and humidity.

5. https://griptip.skisporet.no/

6

Implementation

This chapter goes through the implementation details on how data were extracted from some of the relevant sources. Some of the sections are suggestions, such as Section 6.1, 6.2, and 6.3, on how the microservices can be implemented, while the other sections only explains how data was fetched.

6.1 Fact Service

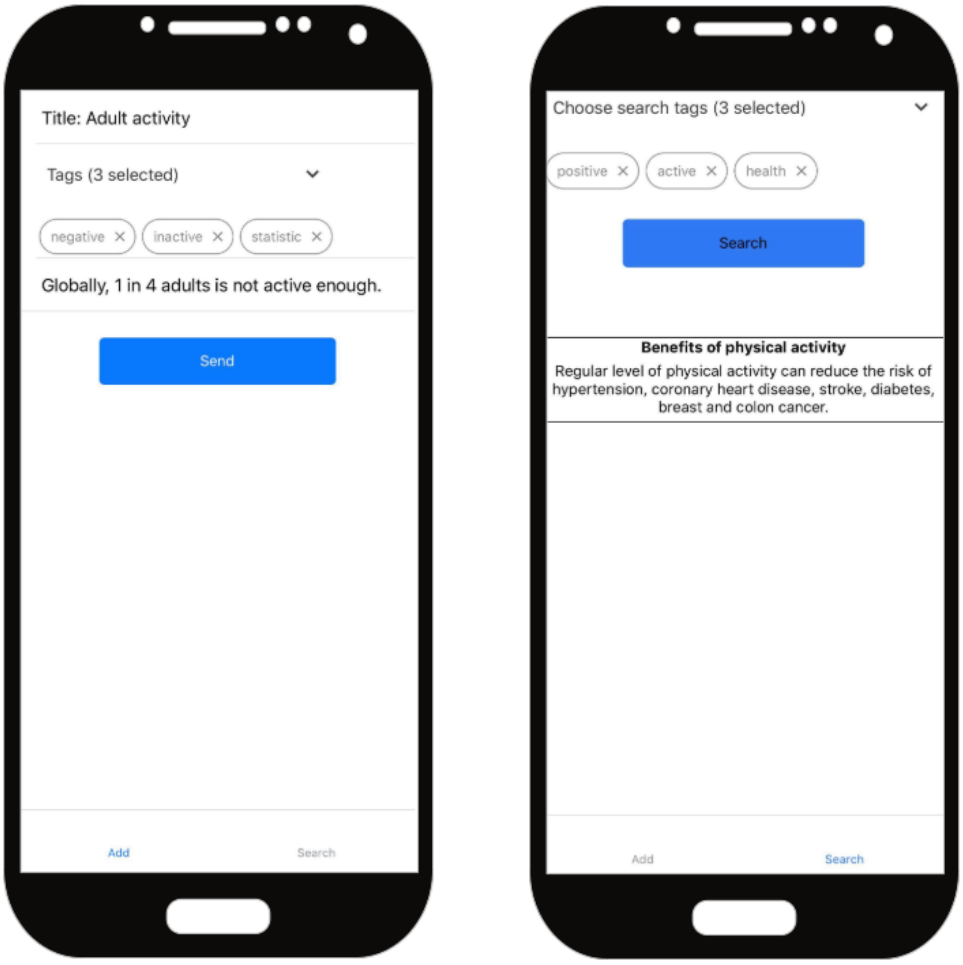
As mentioned in 5.4, the fact service is supposed to spread more knowledge around the domain of physical health. It does this by acting as a Data Service for the Nudge Design Service to fetch fact data from the back-end of the Fact Service.

For the Fact Service, A front-end application and a back-end application was implemented using the MERN stack. It means that the whole app was implemented in JavaScript.

The front-end application was implemented in React Native. It shows a demonstration of two different pages, *Add* and *Search*, that could be integrated into the nudge application used by the system's users. The front-end application for this thesis is mainly implemented to make it easier to visualize the data and test how users can search and add facts if this is decided to be added to the nudge application.

The page to add facts shows a demonstrator on how people with the right privileges can add new information to the system, which is illustrated in Figure 6.1a. A form is required to be filled out, which consists of a title, tags, and description. Once the form is filled out and the "Send" button is pressed, the title, tags, and description is packed into a JavaScript Object Notation (JSON) object, which is sent to the back-end as a POST request to `/facts`.

On the search page, shown in Figure 6.1b, the user can search for facts using tags. The tags are chosen from a list of tags. Currently, the list consists of the tags that are suggested in Table 5.1. When tags are selected, and the "Search" button is clicked, the application sends a POST request to the back-end for `/searchFact` with a JSON object that includes a list of the selected tags. The back-end response is a list of facts relevant to the chosen tags, which is then visualized on the screen of the application.



(a) The "Add" page.

(b) The "Search" page.

Figure 6.1: Screenshots of the two different pages of the React Native application taken on a Iphone Xs.

The back-end is an API server that is implemented using the Express framework, and the facts are stored in a MongoDB database, which is configured, and set up using Mongoose. Table 6.1 shows all the implemented endpoints, with a description of each of them. The Fact model consists of three attributes; a *title* of type string, *tag* of type string list, and a *description* of type string. When creating a new Fact, using a POST request to `/facts`, all of the three attributes are required, and the title also has to be unique.

When the back-end gets a POST request to `/searchFacts`, it is required that the request includes a JSON object that includes a list of tags. These tags are the search terms, and the back-end server extracts all the Facts that have

at least one of the tags in their *tag* attribute list from the database. Each of these Facts is scored using the Term Frequency-Inverse Document Frequency (TF-IDF) algorithm, which scores the Facts based on how many of the search terms are included in the tag attribute list of each Fact. After every Fact is scored, a JSON object with a list of all facts that match *at least* one tag together with their score, is sent back as a response to the request.

Table 6.1: Table that shows the end points that the back-end can receive requests from, and a description of each of the endpoints.

Request Method	Route	Description
GET	/facts	Returns a JSON object with a list of all the Facts.
POST	/facts	Adds a Fact to the database.
POST	/searchFacts	Returns a JSON object with a list of the Facts that satisfies the search term with their relevancy.

6.2 Weather Service

The Weather Service was implemented as an API server using Express. The server has two different endpoints that are listed in Table 6.2. The service can fetch both historical weather data and weather forecasts around users' geographical location by using weather data from the Norwegian Meteorological Institute. The service uses two different third-party API's from the Norwegian Meteorological Institute, which is MET Weather API[1] for the weather forecast, and Frost API[2] for historical weather data.

1. https://api.met.no/weatherapi/
2. https://frost.met.no/

Table 6.2: Table that shows the endpoints that service can receive requests from, and a description of each of the endpoints.

Request Method	Route	Description
GET	/currentweather/:lat /:long/:ndays	Returns a JSON object with a list of weather forecast for the next *ndays* for the given geographical location.
GET	/history/:lat/:long /:date	Returns a JSON object with weather data for the given historical *date* for the given geographical location.

The endpoint for weather forecast requires three different route parameters; *latitude*, *longitude*, and *the number of days* that the service should return forecast for. The service, with the use of the geographical coordinates given in the request, makes a request to MET Weather API's endpoint for location forecast, `/weatherapi/locationforecast/1.9/?lat=:lat&long=:long`. This request returns an XML object with the weather forecast for the next ten days for that location. What the Weather Service does with this data, is to transform the data into a JSON object, parse the data, and return a list of the weather forecast for the next *ndays*, ranging from 0 to 10 days. If *ndays* is 0, it returns the current weather forecast as a response. The response for *ndays*= 0 is shown in the next code listing:

Listing 6.1: The JSON response object that the Weather Service sends out for forecasts.

```
{
    "time": "2020-06-08T12:00:00Z",
    "location":{
        "_attributes":{
            "altitude": "7",
            "latitude": "69.649",
            "longitude": "18.955"
        }
    },
    "temperature": {
        "_attributes": {
            "id": "TTT",
            "unit": "celsius",
            "value": "8.3"
        }
    },
    "humidity": {
        "_attributes": {
            "unit": "percent",
            "value": "88.4"
        }
    },
    "cloudiness": {
        "_attributes": {
            "id": "NN",
            "percent": "99.1"
        }
    },
    "windProbability": {
        "_attributes": {
            "unit": "probabilitycode",
            "value": "0"
        }
    },
}
```

Extracting historical weather data was slightly more complex compared to extracting weather forecasts. The service requires three different route parameters to be able to extract historical weather data. That is, *latitude*, *longitude*, and *date*. The Norwegian Meteorological Institute has multiple different sources that mea-

sure and store weather data. When the Weather Service receives a request for historical data, it first needs to locate sources near the geographical location given. It is done by sending a request to Frost API's endpoint `/sources` where extra route parameters like `geometry=nearest(POINT(:lat :long))` and `nearestmaxcount=:n` can be applied. With the use of this, the service gets a list of up to $n = 5$ of the closest sources to the given geographical coordinates. The list of sources gets looped through, and for each source, observations are collected for the given date through the endpoint `/observation` from FROST API. This route takes additional route parameters, such as `referensetime=:date` and `elements=:listOfElements`. The list of elements is a list of what data is requested from the source. Possible elements that Frost API supports is 439 different elements[3], but currently only air temperature for that day and the sum of precipitation is collected. The sources receive measurements from different devices located at different geographical locations, so the measurements from each source may differ. As the Weather Service collects data from up to 5 different sources, the response is the average of all the sources. The response from the microservice is sent as a JSON object that looks like this:

Listing 6.2: The JSON response object that the Weather Service sends out for historical data.

```
{
    "date": "2020-02-11",
    "temperature": 0.1,
    "precipitation": 5.5,
    "numStations": 4,
    "location": {
        "lat": 69.64,
        "long": 18.95
    },
    "sources": [
        "SN90451",
        "SN90450",
        "SN90400",
        "SN90510",
        "SN90490"
    ]
}
```

3. https://frost.met.no/elementtable

6.3 Trip Suggestion Service

The Trip Suggestion Service was implemented as an API server in Express, which gives out a list of trips near a given location. As seen in Table 6.3, the route requires three parameters; *latitude*, *longitude*, and the *distance* for which to search for trips.

Table 6.3: methods

Request Method	Route	Description
GET	/trips/:lat/:long /:distance	Returns a JSON object with a list of trips that are inside a radius of *distance* kilometers from *(latitude, longitude)*. The trips include metadata about each single trip

UT.no[4] has thousands of trip suggestions, and the Trip Suggestion Service collects information about the trips through the use of Web Scraping. As this is only a demonstrator on extracting data from trips, the service web scrapes the web application on each request. In a real system, this would be done periodically, and stores in a local database, as it takes time to scrape information about every trip that the web application has.

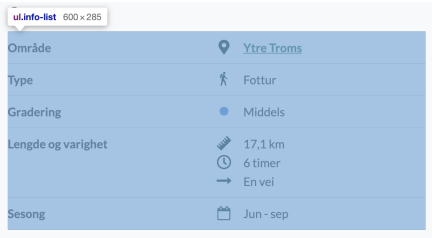
When the back-end server receives a request, the service will send a GET request to UT's open API, `/https://ut.no/api/mapcompact/trip`, that will return a list of ID's, together with the geographical coordinates for each ID. The ID's are the identification for each trip that the web application has data about.

Each trip has its own page in the web application with more information about that specific trip. The Uniform Resource Locator (URL) has the pattern `https://ut.no/turforslag/:id`, and since their open API gave a list of all the ID's, each trip's page is visited to extract relevant data from it.

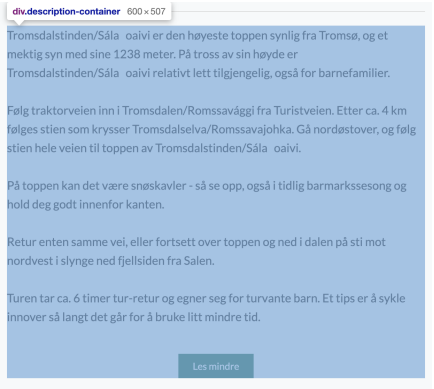
After receiving the list of ID's and the geographical location of each trip, the service loops through all the ID's, and makes a GET request to each trip's details page to get the HyperText Markup Language (HTML) for each page. As seen in Figure 6.2a, the page has a `<ul>` HTML tag with a class name *info-list* which is an unordered list that stores metadata about the trip in key-value pairs.

4. https://ut.no/

Figure 6.2b shows that the page also has a `<div>` tag with the class name *description-container*, which has a text element with description of the trip. The Trip Suggestion Service parses the HTML to find these two class names, and extracts the information in them for each ID's details page.



(a) The unordered list with metadata about a trip.



(b) The container tag with description about a trip.

Figure 6.2: Screenshots of two HTML elements that hold relevant information on the details page of a trip.

Once data about each trip is collected, The service finds out which trips are near the given geographical position. It is calculated using the Haversine Formula, a formula to measure the distance between two points in the form of latitude and longitude on a sphere. As latitude, longitude, and the distance to limit the search are required to make a request to the Trip Suggestion Service, each trip calculates the distance between the trip and the given location. The service filters out all the trips which are not inside the threshold given. The *distance* route parameter is given in kilometers and is the radius that the trips need to be inside. The service returns the client a JSON object with a list of all the trips that are inside the threshold of the given location. Each trip has key-values of *id*, *title*, *description*, *area*, *type of trip*, *difficulty*, *length*, *duration* and *season*. This is shown in the next code listing:

Listing 6.3: The JSON response object that the Trip Suggestion Service sends out. This object is a list of trip suggestions near a user.

```
[
    {
        "Id": "116168",
        "Tittel": "Rundtur Tromsdalen - Svarthammeren -
            ↪ Fjellheisen.",
        "Beskrivelse": "Turen gar etter turistveien fra
            ↪ Ishavskatedralen og innover Tromsdalen. I tett
            ↪ lovskog til a begynne med men etter hvert apner
            ↪ den seg opp og nar en tar av pa sti under
            ↪ Svarthammeren kommer en raskt hoyere opp og over
            ↪  tregrensa. Fin utsikt langs stien opp mot
            ↪ fjellheisen. Var obs. pa hoy vannstand i enkelte
            ↪  bekker og elver som ma krysses under
            ↪ varlosningen. Jeg matte ta av meg skoene og
            ↪ vasse over ei av elvene i iskaldt snosmeltevann.
            ↪  Forfriskende bare. En kan selvsagt ta turen den
            ↪  andre veien men da blir det stort sett bare
            ↪ utforbakke og det blir for lett og kjedelig.....
            ↪  En far jo en flott belonning til slutt nar en
            ↪ star over Tromso by og ser hele byen og omlandet.
            ↪  Turen kan egne seg for mountainbike ogsa.",
        "Omrade": "Ytre Troms",
        "Type": "Fottur",
        "Gradering": "Enkel",
        "Lengde": "10,3 km",
        "Varighet": "3 timer",
        "Sesong": "Mai - okt"
    },
    {...}
]
```

6.4 Sensor Data

A mobile sensor application was implemented as a demonstrator on how sensor data can be fetched out from smartphones. The demonstrator is built using the Expo toolchain, together with React Native.

As seen in Figure 6.3, the demonstrator fetches data from many different sen-

sors. The sensors that are used are the accelerometer, gyroscope, barometer, pedometer, and the location sensor. The sensors, accelerometer, gyroscope, barometer, and the pedometer can be directly accessed using the `expo-sensors` library that Expo has provided. The accelerometer and gyroscope data is fetched every 500ms and is updated on-screen in the table on the application.

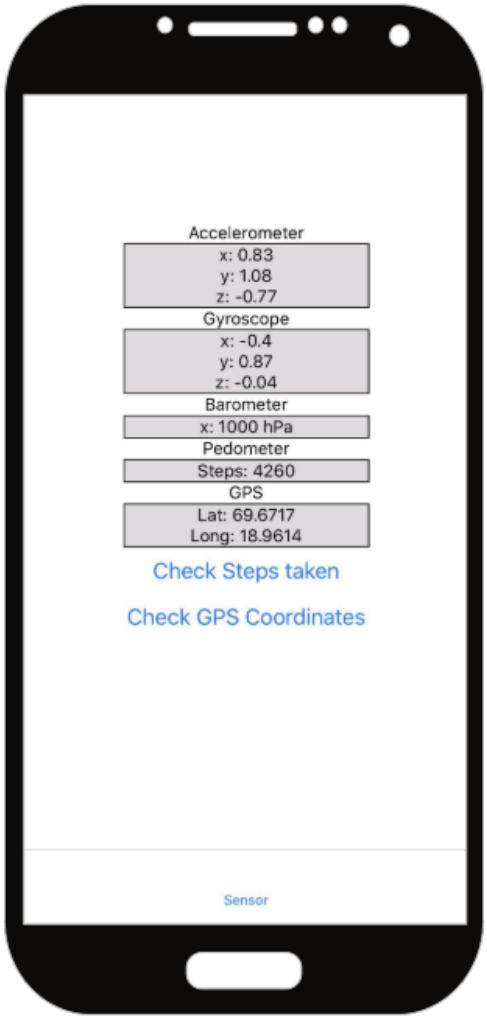


Figure 6.3: Screenshot of the mobile application made to fetch sensor data from the device. The application displays some of the sensors that was being used, together with their values, and two buttons which is used to fetch data from the pedometer and GPS.

To be able to fetch the geographical location, the application uses a library from Expo, called `expo-location`. The app has to ask for permission to use

the GPS location service before it can fetch the geographical position of the client. When clicking the button, *Check GPS Coordinates*, the system checks if permission is granted or not. If the system has been granted permission, it starts the process to fetch the client's location, and if not, it starts the process to ask for permission.

6.5 Calendar Event Data

A mobile application was implemented to demonstrate how calendar events could be extracted from smart devices. The application was built using Expo and React Native. To test the application, some events were generated and added into the ICloud calendar of the device that the application was developed on.

Figure 6.4 shows a screenshot of how the application looks like. The application consists of a button, *Collect calendar events*, and a list of calendar events that have been fetched. The application makes use of the `expo-calendar` library, and when the button is clicked, the system fetches all the different calendars the device is subscribed to. As the relevant events are stored in the ICloud calendar, all other calendars are filtered out based on the name of the calendars. The system fetches out the events from the ICloud calendar for the next 3 days and displays them in a list on the screen.

Figure 6.4: Screenshot of the mobile application made to fetch calendar events from the device. It consists of a button to fetch the events, and a list of calendar events that has been fetched.

7

Smart Nudge Generation with Obtained Data

In Chapter 5, a system architecture was designed using microservices, where many of the services collect data that is relevant for generating smart nudges. In Chapter 6, some of these Data Services were implemented as demonstrators, and demonstrators for fetching sensor data and calendar events were also made. This chapter goes through the process of generating smart nudges by looking at the opportunities from what data has been collected in this thesis, and what the Data Sources from the system design can give. The smart nudge examples will be using real data that has been fetched and will show suggestions on how the data can be used as part of the smart nudge.

7.1 Generation of smart nudges

Some of the data that has been extracted from data sources can be used alone as content in smart nudges towards physical activity. In contrast, other sources need to be combined with other data to be relevant. Figure 7.1 shows how data from different sources can be used as content in smart nudges. The figure consists of two parts, *the data collected from relevant sources* and *smart nudges*. The data from relevant sources are separated into boxes representing what type of data has been extracted. These are then color-coded to separate them

more easily. The next part shows examples of smart nudges that are generated using the data collected. The colored dots show what data types that could be used to generate that specific type of smart nudge.

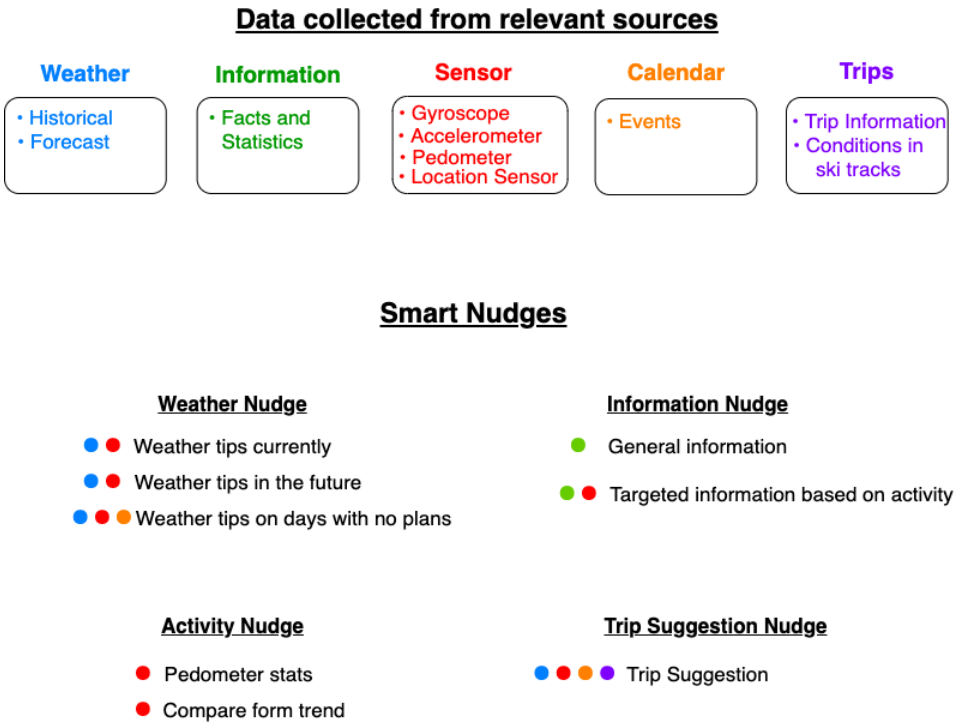


Figure 7.1: The figure consists of two parts, data collected from relevant sources and smart nudges. The data that has been collected from the relevant sources are sorted into groups in the first part of the figure, and in the second part examples of what type of smart nudges are given.

As mentioned in Section 5.3 about Nudge Design, Sandor Delecke constructed a model for how to design nudges. The model consists of combining *effects*, *incentives*, and *contents*[18]. This section used this nudge generation model to generate smart nudges based on the data that has been extracted in this thesis. Table 7.1 and 7.2 shows the generation of designing smart nudges using his model and combined with the data that has been fetched from different sources in this thesis.

Table 7.1: Table showing the generation process of creating smart nudges.

Nudge goal: Go hiking tomorrow. **Effect**: You should go hiking tomorrow. You have nothing on your schedule. **Content**: The temperature is expected to be 20°C at 12pm. **Content**: A trip up Sherpatrappa will take around 2.0h. **Incentive**: This is a perfect trip to challenge your form. **Result**: Hi Lars. You should go hiking tomorrow. You have nothing on your schedule. The temperature is expected to be 20°C at 12pm. A trip up Sherpatrappa will take around 2.0h. This is a perfect trip to challenge your form.	**Nudge goal**: Go skiing **Effect**: You should go skiing today. You have nothing on your schedule. **Content**: The weather is currently sunny, almost no wind and 2°C. **Content**: With the conditions in the track at Lysløypa, Tromsø, we suggest using a Swix VP65 ski wax. **Incentive**: Skiing is a great way to exercise. **Result**: Hello Lars. You should go skiing today. You have nothing on your schedule. The weather is currently sunny, almost no wind and 2°C. With the conditions in the track at Lysløypa, Tromsø, we suggest using a Swix VP65 ski wax. Skiing is a great way to exercise.
Nudge goal: Spread knowledge **Content**: You have been jogging a lot lately. **Content**: Did you know that regular physical activity improves bone and muscular strength? **Incentive**: Keep it up! This is good for your health. **Result**: Hi there. You have been jogging a lot lately. Did you know that regular physical activity improves bone and muscular strength? Keep it up! This is good for your health.	**Nudge goal**: Reach pedometer goal **Effect**: You are close to reaching your daily step goal. **Content**: You have currently walked 9123. You are only 877 steps away from your goal. **Incentive**: If you just walk a bit more you will reach it. **Result**: Hi there, Lars. You are close to reaching your daily step goal. You have currently walked 9123. You are only 877 steps away from your goal. If you just walk a bit more you will reach it.

Table 7.2: Table showing the generation of creating smart nudges.

Nudge goal: Improve form **Effect**: Your form has been decreasing. **Content**: You were 32% less active this week than last week. **Incentive**: You should try to be more active next week. **Result**: Hi Lars. Your form has been decreasing. You were 32% less active this week than last week. You should try to be more active next week.	**Nudge goal**: Make plans **Effect**: You should make plans for Sunday! **Content**: The forecast for Sunday is 20°C at 3pm, with no rain and almost no wind. **Content**: You have nothing planned for this day. **Incentive**: This is a great opportunity to do something fun outside. **Result**: You should make plans for Sunday! The forecast for Sunday is 20°C at 3pm, with no rain and almost no wind. You have nothing planned for this day. This is a great opportunity to do something fun outside.

As there is a common format that all the services send data in, building smart nudges becomes quite simple. The Nudge Design Service receives data in JavaScript Object Notation (JSON) format, and knows the keys of the object from the Data Services as the Nudge Design Service has information about the format from each microservice. For example, to build the smart nudge with the goal of *Go hiking tomorrow*, the data that is put in the content of the smart nudge message comes from the Weather Service and the Trip Suggestion Service. The JSON object that these microservices gives in response are shown in Listing 6.1 and 6.3. To extract the temperature and time from the Weather Service, this can be done by accessing the keys `obj.temperature._attributes.value` and `obj.time`. Extraction of the trip name and the trip duration that is needed from the Trip Suggestion Service could be done by accessing the keys `obj.tittel` and `obj.varighet`.

8 Discussion

In this chapter, discussion about the chosen architecture for the smart nudge system is made and compared to a monolithic architecture. The chapter also discusses the data sources that have been used in this thesis and how they are relevant to smart nudging, as well as discussion about the experience made throughout the project.

8.1 Microservices vs. Monolithic Architecture

The chosen architecture for the system in this thesis was a microservice architecture. The system that we at the ODS group is trying to develop will depend on many factors. Factors that will change regularly and some might be outdated fast. These factors might be less complicated frameworks to use, new relevant data sources, limitations in programming languages, and more libraries and resources used in different software ecosystems. These factors were kept in mind during the design of the system, which is why microservices was used.

Some of the microservices have a basic design, as some microservices fetch data from a third-party source and provide the client with the data that has been collected. Other microservices are more complex and benefit from specific frameworks and libraries for possibly analyzing data or building a deep learning microservice. By having the system developed as a monolithic architecture,

these options are limited to the programming language's environment and the framework chosen. Some frameworks are only available in specific programming languages, which is unfortunate. With the microservice design choice made in this thesis, each microservice can be developed with the framework and programming language that benefits the microservice the most.

People are different, and therefore, various sources might affect people differently. The choice to split each component of the system into separate microservices makes the system flexible and simpler to scale. As of now, there are designed only a few data services that are relevant for nudging people to be more active. As more research is put into this project, more knowledge of what sources affect people will be known, as well as new trends in the future will change the way people are active. With more services included, complexity becomes a problem in a monolithic architecture. The services are limited to the software ecosystem, and with these limitations, workaround needs to be made, which will make the system more complex and harder to maintain. The microservice approach will not increase as fast in complexity as the microservices can be made separately, and in their own environment, making the limitations talked about irrelevant.

Many factors affect what type of activity that is relevant. It could depend on the season of the year, what is trending, weather, etc. The lockdown due to COVID-19 affected people's motivation to go outside and many regions recommended people to stay inside. This affected the physical activity of many people. With the use of a microservice architecture, as this thesis suggests, a new service that can give suggestions to indoor activities can easily be added and prioritized when designing smart nudges to people, rather than prioritizing activities that are not relevant in that situation. The system can adapt fast as each Data Service is independent from the rest of the system, and the Data Services does not do anything without being requested to. It means that if a Data Service is not relevant anymore(i.e. microservice about skiing and snow conditions), the nudge design service could remove templates for generating smart nudges that include data from this source. It will result in no more use of that data service, as all smart nudge generation goes through the nudge design service.

In Chapter 3 about related work, two relevant papers were discovered and described. These two papers explained a system that both seemed to be monolithic. The reason why a monolithic system might fit their system better than the system designed in this thesis is that they both are limiting themselves to a particular group of people, people with diabetes, and multimorbidity. The boundaries are more evident in these papers, and the relevant sources are limited in as well.

There are some drawbacks to choosing a microservice architecture. As mi-

croservices are independent and not built into a monolithic system, it makes it a distributed system. Working with distributed systems is more complicated than a monolithic system. Having to control and set up the communication between the different microservices of the system can be hard. Having consistency in a distributed system is a challenge if it is needed while having good availability. Testing a distributed system is harder than in a monolithic system, as the microservices are independently deployable and need to be tested together with the rest of the microservices. The complexity of building a monolithic system will be low at the start of development but will increase rapidly over time when new features and services are added. With microservices, and even Service-Oriented Architecture (SOA), which is similar to microservices, but the services are more coarse-grained than fine-grained, the complexity is higher at the start of development, but will not increase as fast as monolithic architecture in the long run. This is because the main problems occur at the start of the development, which is the communication between the different microservices and the flow between the microservices. Once these problems have been solved and tested properly, new microservices can be added with less complexity as they are small components that should not affect the whole system when added.

When the Nudge Design Service is making a smart nudge, it collects data from the Data Services needed for the specific nudge. This can easily be done because of the design choices made in this thesis that the Data Services send data in the same format, JavaScript Object Notation (JSON). The JSON format is one of the most common formats and supported in most programming languages. For instance, some of the microservices, like the Weather Service, collect data from Meteorologisk Institutt in Extensible Markup Language (XML) format and need to parse the data and change the format to JSON. A middle-layer could have been made between the data services and the Nudge Design Service, to change the format to JSON, so the Data Services do not need to change the format. The microservices should be limited in what services they provide, and changing format adds unnecessary complexity to the data services.

Overall, designing the system using microservices seems to be the best fit for this project. Factors will change over time, and being able to change the system as fast and straightforward as possible to cope with the changes is of high importance, and having the right tools for the right services. The system could have been built as a monolithic system, but then the system might not be changeable in the long run without building the system from scratch again because of complexity and outdated software.

8.2 Data sources and the relevancy to smart nudges

Data has been extracted from relevant data sources that people could be affected by. In Chapter 5 and 6, about design and implementation, details were described on how the relevant data was extracted from their sources. In the previous chapter, 7.Smart Nudge Generation with Obtained Data, examples were shown on how the data could be used to generate smart nudges through the use of the Nudge Design Service that uses Sandor Dalecke's nudge generation model. The generated smart nudge examples show that the data sources that are used, and the data that they hold, support smart nudging in physical activity as the data help to enrich the nudges with relevant information that people can be motivated by.

The sources that are used in this thesis are chosen as a result of previous work I have conducted in a Capstone Project[8]. These were sources that were found after analyzing interviews, but as time was limited, the number of sources that were found was also limited. There are possibly many other sources that hold data that could be relevant to integrate into the system that could be used to generate even more enriched smart nudges towards a more active life.

In this thesis, awareness about extracting data from third-party sources have been made. Many of the data sources that provide relevant data does not have any specific end-point to collect *only* the relevant data. Usually, a lot of unnecessary data comes with it. It could be problematic if some microservices have much traffic, and need to collect a lot of unneeded data in every request to get the relevant data. It could make the whole system slow, as the data could be needed from another microservice, and having to wait. Thoughts have been made about using GraphQL[1] to extract only the relevant data from requests to third-party sources. GraphQL is a library that uses a query language that can extract certain elements from data given from API's. Instead of receiving all the data, a query can be used that will include only the relevant data.

A lot of the Data Services can be generalized as the design and implementation are somewhat similar. All of the data services serve as API servers that have end-points to give out data or store data. The data that is given out to the client is collected from an internal database which the server has access to, or collected from a third-party source. What separates the Data Services are *where* the data is collected from(third-party sources, users), *how* the data is collected(API's, Web Scraping, receiving data from user), and if the data service needs further analysis or data aggregation before storing or using the data.

1. https://graphql.org/

Knowing this can make implementation easier in the future when new data services are required.

Many of the smart nudges generated in the previous chapter were suggestions on activities as reminders to them being available for that day, as no plans were scheduled on the calendar. Other smart nudges were generated, informing them either in general about physical activity or about how their physical form has been lately. In Section 2.2.1, Cass R. Sunstein mentions what he means is the ten most important types of nudges, and a couple of these have been applied in the examples made, but more types of nudges could be created together with the collected data to form other types of smart nudges. It could be the use of social norms to inform of what others do, or how active others are compared to you, or formulate activity suggestions differently to give the user a default option.

Data has been extracted from many different data sources that have been found relevant in previous work. This data has been combined and used to generate smart nudges to show examples of how the data helps to make the nudges go from being normal nudges to smart nudges. The data sources are used as part of a microservice system to show how these sources can be used to build a system that should give smart nudges towards physical activity.

9

Conclusion

Throughout the thesis, data has been gathered from different relevant sources that hold data that can be used as a motivational factor towards physical activity. Theses sources have been integrated into a system of microservices that uses the data to build smart nudges. General Data Protection Regulation (GDPR) has been mentioned and thought about while designing the system, as privacy and protection of the user is essential. The thesis gives examples of smart nudges that the system can generate with the use of the data that has been collected together with the Nudge Design Service. To conclude, this thesis shows how data from relevant sources can be collected and used to create smart nudges for a system that focuses on physical activity, and the thesis contributes future research with identifying what relevant data the data sources hold, how the data can be extracted from these sources and how these sources with their data can be used in a smart nudging system.

For future work, more research on what data sources could be relevant for physical activity could be needed. The sources that are collected data from this thesis were discovered in the previous project I conducted, which was time-limited. Only some of the components of the system were implemented. Therefore, the rest of the components need to be implemented to test the system entirely. It includes a mobile application for the users, the API gateway, and the rest of the microservices that the system consists of. Demonstrators need to be tested on smartwatches as the demonstrators that collect data from the user is tested on smartphones only.

10

References

[1] W. H. Organization. (2018). Who launches global action plan on physical activity, [Online]. Available: https://www.who.int/news-room/detail/04-06-2018-who-launches-global-action-plan-on-physical-activity (visited on 05/15/2020).

[2] E. Union, "Sport and physical activity," Dec. 2017. DOI: 10.2766/483047.

[3] W. H. Organization. (2018). Physical activity factsheets: For the 28 european member states of the who european region, [Online]. Available: http://www.euro.who.int/en/health-topics/disease-prevention/physical-activity/data-and-statistics/physical-activity-fact-sheets/factsheets-on-health-enhancing-physical-activity-in-the-28-eu-member-states-of-the-who-european-region (visited on 02/14/2020).

[4] J. Tillu. (2018). Mobile sensors: The components that make our smartphones smarter, [Online]. Available: https://medium.com/jay-tillu/mobile-sensors-the-components-that-make-our-smartphones-smarter-4174a7a2bfc3 (visited on 02/18/2020).

[5] C. R. Sunstein, "The ethics of nudging," Nov. 2014. DOI: 10.2139/ssrn.2526341.

[6] Medium. (2017). On nudge, free will, and ethics, [Online]. Available: https://medium.com/neurowonk/on-nudge-free-will-and-ethics-ca2979bccb54 (visited on 02/15/2020).

[7] A. Håkansson, "Portal of research methods and methodologies for research projects and degree projects," in. CSREA Press U.S.A, 2013, 67–73, [ed] Hamid R. Arabnia Azita Bahrami Victor A. Clincy Leonidas Deligiannidis George Jandieri, ISBN: 1-60132-243-7. [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-136960.

[8] L. Karlsen, *Information sources for smart nudges to a more active living*. UiT - The Arctic University of Norway, 2019.

[9] W. H. Organization. (). Physical activity, [Online]. Available: https://www.who.int/dietphysicalactivity/pa/en/ (visited on 02/18/2020).

[10] R. H. Thaler and C. R. Sunstein, *Nudge: Improving Decisions about Health, Wealth, and Happiness*. Yale University Press, 2008, ISBN: 978-0-14-311526-7.

[11] C. R. Sunstein. (2014). Nudging: A very short guide, [Online]. Available: http://nrs.harvard.edu/urn-3:HUL.InstRepos:16205305 (visited on 02/15/2020).

[12] D. H. et al. (2007). Communicating risk to smokers: The impact of health warnings on cigarette packages, [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1868456/ (visited on 10/15/2019).

[13] T. F. Center. (2011). Warning labels: Essential facts, [Online]. Available: https://www.tobaccofreekids.org/assets/global/pdfs/en/WL_essential_facts_en.pdf (visited on 10/15/2019).

[14] M. Weinmann, C. Schneider, and J. v. Brocke, "Digital nudging," vol. 58, pp. 433–436, Oct. 2016. DOI: 10.1007/s12599-016-0453-1.

[15] E. J. Johnson and D. Goldstein, "Do defaults save lives?," vol. 302, no. 5649, pp. 1338–1339, 2003, ISSN: 0036-8075. DOI: 10.1126/science.1091721. [Online]. Available: https://science.sciencemag.org/content/302/5649/1338 (visited on 02/16/2020).

[16] R. Karlsen and A. Andersen. (2019). Recommendations with a nudge, [Online]. Available: https://munin.uit.no/bitstream/handle/10037/16351/article.pdf?sequence=2&isAllowed=y (visited on 02/16/2020).

[17] H. Wallann, "Roadahead — removing uncertainty in travel," Master's thesis, The Arctic University of Norway, 2019. [Online]. Available: https://munin.uit.no/handle/10037/15735.

[18] S. Dalecke, "Designing dynamic and personalized nudges," Master's thesis, University of Kaiserslautern, 2019.

[19] C. R. Crăciun, "Data management for nudged green transportation," Master's thesis, The Arctic University of Norway, 2019. [Online]. Available: https://munin.uit.no/handle/10037/15410.

[20] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative research in psychology*, vol. 3, pp. 77–101, Jan. 2006. DOI: 10.1191/1478088706qp063oa.

[21] Technopedia. (). What is a smart device? [Online]. Available: https://www.techopedia.com/definition/31463/smart-device (visited on 02/19/2020).

[22] L. Gleason. (). Wireless communications in modern mobile technologies, [Online]. Available: https://dzone.com/articles/wireless-communications-in-modern-mobile-technolog (visited on 02/19/2020).

[23] S. Etienne. (). Apple announces ios 12 with new ar features, photos improvements, and more, [Online]. Available: https://www.theverge.com/2018/6/4/17414386/ios-12-announced-features-release-date-apple-wwdc-2018 (visited on 02/19/2020).

[24] A. Inc. (). How to find your parked car with maps on your iphone, [Online]. Available: https://support.apple.com/en-us/HT207227 (visited on 02/19/2020).

[25] ——, (). Siri does more than ever. even before you ask, [Online]. Available: https://www.apple.com/siri/ (visited on 02/19/2020).

[26] G. Android. (). Sensors, [Online]. Available: https://developer.android.com/guide/topics/sensors (visited on 02/20/2020).

[27] Przemek. (). Scanning rooms with an iphone, [Online]. Available: https://www.nomtek.com/scanning-rooms-with-an-iphone/ (visited on 02/19/2020).

[28] R. M.Shirer and J. D. Corporation). (2019). Idc forecasts steady double-digit growth for wearables as new capabilities and use cases expand

the market opportunities, [Online]. Available: https://www.idc.com/getdoc.jsp?containerId=prUS44930019 (visited on 02/14/2020).

[29] A. Inc. (). Use fall detection with apple watch, [Online]. Available: https://support.apple.com/en-gb/HT208944 (visited on 02/19/2020).

[30] Fitbit. (). Works with fitbit, [Online]. Available: https://www.fitbit.com/partnership (visited on 02/20/2020).

[31] A. Inc. (). Apple watch series 4 - technical specifications, [Online]. Available: https://support.apple.com/kb/SP778?locale=en_US (visited on 02/20/2020).

[32] Fitbit. (). Fitbit versa 2 | health & fitness smartwatch, [Online]. Available: https://staticcs.fitbit.com/content/assets/help/manuals/manual_versa_2_en_US.pdf (visited on 02/20/2020).

[33] ——, (). Fitbit versa 2 user manual, [Online]. Available: https://staticcs.fitbit.com/content/assets/help/manuals/manual_versa_2_en_US.pdf (visited on 02/20/2020).

[34] (). General data protection regulation, European Commission, [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679 (visited on 03/02/2020).

[35] S. Newman, *Building Microservices: Designing Fine-Grained Systems*, 1st. O'Reilly Media, 2015, ISBN: 978-1491950357.

[36] A. Kofod-Petersen, "How to do a structured literature review in computer science," May 2015.

[37] M. Generali, M. Gazzano, and M. Dolla, "Pc4hc: Personalized communication for health care," in *2017 IEEE 3rd International Forum on Research and Technologies for Society and Industry (RTSI)*, 2017.

[38] R. Akker, R. Klaassen, K. Bul, P. Kato, G.-J. Burg, and P. Bitonto, "Let them play: Experiences in the wild with a gamification and coaching system for young diabetes patients," May 2017, pp. 409–418. DOI: 10.1145/3154862.3154931.

[39] W. H. Organization. (). Physical activity fact sheet, [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/physical-activity (visited on 03/30/2020).

[40] Fitbit. (). Works with fitbit, [Online]. Available: https://www.fitbit.com/fi/smarttrack (visited on 04/03/2020).

[41] G. Filios, S. Nikoletseas, and C. Pavlopoulou, "Efficient parameterized methods for physical activity detection using only smartphone sensors," Nov. 2015, pp. 97–104. DOI: 10.1145/2810362.2810372.

[42] A. Ghods and D. Cook. (Jul. 2019). Activity2vec: Learning adl embeddings from sensor data with a sequence-to-sequence model.