Western Norway
University of
Applied Sciences

BACHELOR'S THESIS

Visualizing smart charging of electric vehicles for support personnel

Roger Karlsen, Kristin Standal, Mads Henrik Sørbø

Faculty of Engineering and Natural Sciences
Institute of Computer Science, Electrical Technology and Natural Sciences
Information Technology

Supervisor: Volker Stolz
Submission Date: 23.05.2022

TITLE PAGE FOR PROJECT

Title: Visualizing smart charging of electric vehicles for support personnel	*Date:* 23.05.2022
Author(s): Roger Karlsen Kristin Standal Mads Henrik Sørbø	*Number of pages w/o attachments:* 33 *Number of pages with attachments:* 85
Field of study: Information Technology	*Number of discs / CDs:* 0
Contact at field of study: Volker Stolz	*Grading:* None
Remarks:	

Project Owner: Tibber	*Project Owner reference:* None
Project Owner Contact: Marcus Almgren	*Contact information:* marcus.almgren@tibber.com

Samandrag:
Målet med oppgåva var å lage eit view som ein del av ein eksisterande web-applikasjon. Denne web-applikasjonen er eit internt verktøy som blir brukt av Tibber-ansatte. Viewet må vise og visualisere den informasjonen som trengs for å feilsøke såkalla 'smart charging'-spørsmål. Kundeservice-teamet hos Tibber er hovudbrukarane av dette viewet. Hovudmålet er å gjere kundeservice-teamet meir effektivt når det kjem til kundebehandling. For å evaluere produktet har kundeservice-teamet svart på eit spørjeskjema etter at produktet har blitt sluppet. Prosjektgruppa konkluderte med at det ferdige produktet oppfyller dei gitte krava.

Summary:
The goal of the assignment was to create a view as a part of an existing web application. This web application is an internal tool used by Tibber employees. The view should display and visualize information needed to troubleshoot so-called 'smart charging' questions. The customer support team at Tibber are the main users of the view. The main goal is to make the customer support team more efficient when handling customer cases. For the evaluation of the product, the customer support team answered a survey after the release of the product. The project group concluded that the final product fulfills the requirements given.

Keywords:

Smart Charging	Microservices	Customer tooling

PREFACE

In this report, the work around the bachelor thesis “Visualizing smart charging of electric vehicles for support personnel” is described. Through this project, a product was developed for Tibber. The project is conducted by Roger Karlsen, Kristin Standal, and Mads Henrik Sørbø.

We would like to acknowledge and thank the following people in their help with this project:

- Volker Stolz, Professor in Software Engineering at HVL and thesis supervisor
- Marcus Almgren, Engineering Manager at Tibber and project owner
- Markus Persson, Product Manager at Tibber and responsible for Varys
- Trond Nordheim, Lead Backend Developer at Tibber
- Finn Michael Halvorsen, Senior Software Engineer at Tibber

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Context

Tibber customers are buying energy at the market spot price with an hourly precision, allowing them to optimize their energy consumption by seeing the current and future prices of their respective market. To help in this optimization, Tibber provides different integrations to devices such as: heaters, radiators, coolers, and chargers of electrical vehicles. These integrations aim to automate the workflow of being cost aware, avoiding the most expensive hours when possible.

When doing this optimization on electric vehicles, called smart charging, there are many other variables to consider, other than the hourly market price for energy. Tibber needs to have contact with the vehicle, understand where it is located, know if it is connected to a charger that Tibber controls, have access to the desired charging goals set by the customer, be able to read the current state of charge from the battery among others. Tibber might also start or stop the charging in order to help stabilize the grid frequency. All in all, there are many things that could possibly go wrong and when that happens, the customer support at Tibber wants to swiftly and accurately be able to help understand the root cause of potential larger issues or at least be able to assist that individual customer.

1.2 Motivation

Customer support at Tibber is using an internal tool called Varys and it is in this tool Tibber wants to improve the visualization of events around smart charging of electric vehicles. The different events can come from a multitude of different sources: logs, databases, APIs, and message queues. Getting them collected in an easy-to-understand view is both a very complex task, but also highly valuable and something that can be iterated on to improve it over time. The goal is to build something in Varys that enables non-technical support people to help owners of electrical vehicles with potential issues, combining data from multiple sources into one view.

1.3 Project owner

Marcus Almgren, who is an engineering manager at Tibber, is the owner of the project. For the smart charging view Markus Persson, a product manager at Tibber, has proposed features that they want the new view in Varys to have.

Tibber was founded in 2016 by Daniel Lindén and Edgeir Vårdal Aksnes. In the long run, Tibber wants to develop the best digital tool possible, enabling customers to take advantage of new technologies in order to reduce customers energy consumption and expenses. According to Energy Star, encouraging consumers to consume less energy benefits almost everyone, with the exception of the "dinosaurs'' of the energy industry, who stand to benefit the least (EnergyStar, n.d.).

Today, Tibber employs more than 110 people and is expanding at a rapid rate (Tibber, n.d.).

1.4 Problem description and goal

Currently, Tibber customer support personnel need to fetch, manage, and interpret data manually in various databases and systems. The types of data can be timeseries or state variables, which are fetched from the customer's electrical vehicle, smart charger, or their load balancer. Tibber wants to ease this process for the support department and wishes to aggregate the data from the various logs and state variables and present it within a single view in their in-house support application named Varys. Aggregating the data is valuable in itself, but it will also generate new easy-to-understand notification messages and states, thereby expediting the troubleshooting process.

How can the various events and data streams involved in charging an electric vehicle be visualized online in such a way that a non-technical support person can assist customers?

The first milestone is getting the view to a point where it can be tested by support personnel, as this will enable feedback to evaluate and make changes to the view to further optimize the smart charging troubleshooting.

1.5 Report structure

Chapter two shows a detailed description of the project with specifications from the project owner and how we plan to achieve this. Chapter three goes over the design process for the project. In chapter four, the solution will be showcased, and chapter five will evaluate the result and the project as a whole. Chapter six will discuss the project in general and how it was to work on the project.

2 PROJECT DESCRIPTION

2.1 Practical background

2.1.1 Previous work

Varys, the software on which this project is based in large part, is a complete and robust product built on Vue that has been extensively tested and is currently in use by the entire customer support department. Varys has all information regarding a customer, their homes, subscriptions, and registered devices.

Vue is a progressive framework for creating user interfaces, it is implemented as additional markup to HTML. The core library focuses solely on the view layer, and it is simple to use and integrate with other libraries or projects. When combined with modern tooling and supporting libraries, Vue is perfectly capable of powering intricate Single-Page Applications.

A typical view in Varys is built by a number of generic components that can be reused and extended in a number of different ways. As new components are required, they are developed on a continuous basis. The overall style, including colors, typefaces, padding, and margins, have already been decided.

Varys depends on a number of microservices. Microservices is an architecture that enables rapid, frequent, and reliable delivery of large, complex applications which goes hand in hand with agile development methods. By being as modular as possible, microservices overcome the limitations of monolithic systems. In their simplest form, they aid in the development of applications by separating them into a set of small services, each of which runs in its own process and is independently deployable, they can be monitored and scaled independently. This makes them fail-tolerant, as they can continue operating in the event of a system failure or an error in one of the system's components. These services may be written in a variety of different programming languages and utilize a variety of distinct data storage techniques.

Among the most important ones for this project is the 'Customer' service and the 'Device Orchestrator' service. Please note that throughout this report, the names of different services and attributes will not be the actual words as used in the source code. This is requested by Tibber, because knowing what certain features and attributes are named in the source code can be a back door usable by malicious attackers.

Device Orchestrator (DO)

This service orchestrates third-party device integration and provides access to a collection of device data, as well as device control when supported. It collects data such as the battery level, the location of the car, the charging speed, and the settings from the customer app.

DO determines the average charging power of the preceding charging sessions and compares it to the latest charging session's average. If the value of the most recent charging session differs significantly from the average of the previous charging sessions, it will use the average of the previous charging sessions for the machine learning calculations. If it is less than that, the most recent charging session's value will be used. DO sends this data to a machine learning service, which generates a charging schedule indicating when to begin charging.

2.1.2 Initial requirements

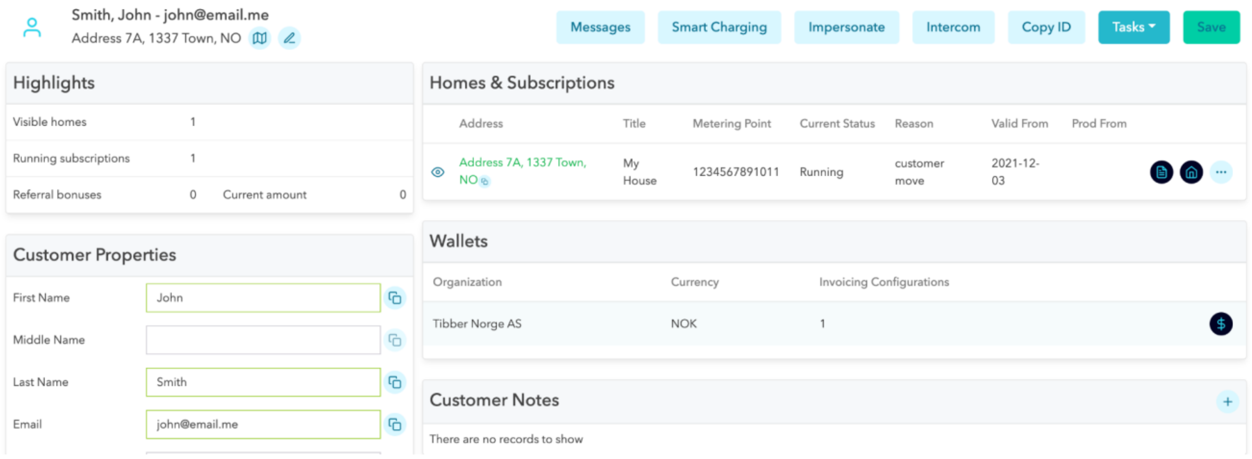
The initial demand for the view is that it should be easy for customer support personnel to quickly find errors in the configuration of cars and/or chargers. They have expressed that they want customer support personnel to stop having to read through logs manually and to have the logs and other data be visualized. That the view should only have information that is related to smart charging. This boils down to having information about cars, chargers, and load balancers. They also want there to be alerts with varying colors depending on severity that pops up at the top of the view if there are any configuration errors. The view should list all cars, chargers, and load balancers in three different tables for each registered home.

This is the minimum set of specifications put forth that is required for the view to be useful to support staff. When this is achieved, support personnel can begin testing.

2.1.3 Current methods for troubleshooting smart charging

In this section, we will present an example of how customer support currently has to figure out a scenario of a misconfigured charger and car, where both have set different departure times, which might lead to errors for when the car should be finished charging.

A customer calls in saying that their car had not finished charging at the correct time. The customer support person has to look up the customer in Varys and is then presented with the information shown in figure 2.1.



Smith, John - john@email.me
Address 7A, 1337 Town, NO

Highlights

Visible homes	1		
Running subscriptions	1		
Referral bonuses	0	Current amount	0

Customer Properties

First Name	John
Middle Name	
Last Name	Smith
Email	john@email.me

Homes & Subscriptions

Address	Title	Metering Point	Current Status	Reason	Valid From	Prod From
Address 7A, 1337 Town, NO	My House	1234567891011	Running	customer move	2021-12-03	

Wallets

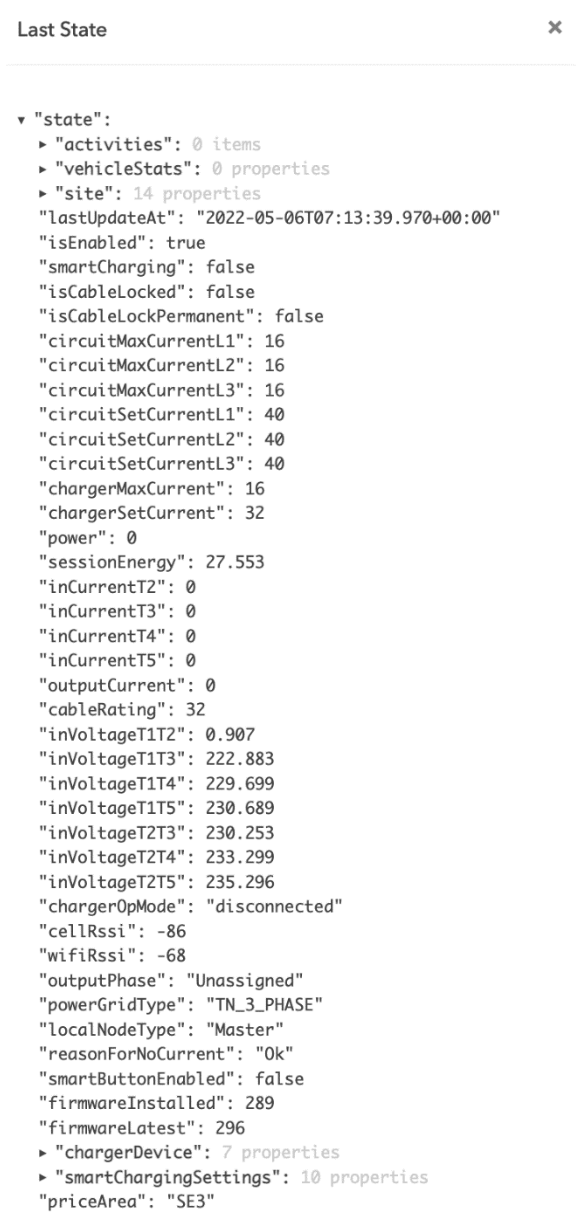
Organization	Currency	Invoicing Configurations
Tibber Norge AS	NOK	1

Customer Notes

There are no records to show

Figure 2.1 Customer look-up

From here the customer support person has to figure out which home (if there are multiple) that might have the error, then proceed to that home to continue. In the home view there is a table that lists all the different devices the customer has registered to that home; in this list the customer support person will have to find the correct device and click a button that lets one look at it as a JSON-modal (see figure 2.2). While viewing the JSON-modal, the customer support person has to navigate to the correct property, which in this example is listed under "smartChargingSettings" and then either copy the values that this device has for departure times and paste them in a document or write them down. Then, they have to exit the modal and find the corresponding charger and find the correct property that shows set departure times. After finding the correct properties they can then start comparing the values and see if they differ. In this scenario the departure times differ, and it was an answer for why the car was not fully charged at the correct time. However, if the departure times were set to the same time, it might be some other reason for why the car did not charge at the correct time and therefore the support person has to start to look for other causes. Doing these steps can take a while and one can see the value of reducing the time the customer support person has to spend navigating the different devices and finding the right properties whilst looking for a cause for the problem.


Last State ×

```
▾ "state":
  ▸ "activities": 0 items
  ▸ "vehicleStats": 0 properties
  ▸ "site": 14 properties
  "lastUpdateAt": "2022-05-06T07:13:39.970+00:00"
  "isEnabled": true
  "smartCharging": false
  "isCableLocked": false
  "isCableLockPermanent": false
  "circuitMaxCurrentL1": 16
  "circuitMaxCurrentL2": 16
  "circuitMaxCurrentL3": 16
  "circuitSetCurrentL1": 40
  "circuitSetCurrentL2": 40
  "circuitSetCurrentL3": 40
  "chargerMaxCurrent": 16
  "chargerSetCurrent": 32
  "power": 0
  "sessionEnergy": 27.553
  "inCurrentT2": 0
  "inCurrentT3": 0
  "inCurrentT4": 0
  "inCurrentT5": 0
  "outputCurrent": 0
  "cableRating": 32
  "inVoltageT1T2": 0.907
  "inVoltageT1T3": 222.883
  "inVoltageT1T4": 229.699
  "inVoltageT1T5": 230.689
  "inVoltageT2T3": 230.253
  "inVoltageT2T4": 233.299
  "inVoltageT2T5": 235.296
  "chargerOpMode": "disconnected"
  "cellRssi": -86
  "wifiRssi": -68
  "outputPhase": "Unassigned"
  "powerGridType": "TN_3_PHASE"
  "localNodeType": "Master"
  "reasonForNoCurrent": "Ok"
  "smartButtonEnabled": false
  "firmwareInstalled": 289
  "firmwareLatest": 296
  ▸ "chargerDevice": 7 properties
  ▸ "smartChargingSettings": 10 properties
  "priceArea": "SE3"
```

Figure 2.2 JSON-modal example

2.1.4 Initial solution idea

As one might infer from the example above, Tibber wants to cut the time customer support personnel use while navigating through the different views and look up data manually. They want to gather everything related to smart charging in one view, where common misconfigurations will alert the support person instantaneously. So, the initial idea is to aggregate all the various data and logs into one view and set up an alarm system that will notify of common misconfigurations as well as having all other data related to smart charging available within the view should there be a need for manual lookup.

A number of different Vue components are used to construct the various views in Varys. The new view that is being created must be consistent with the other views that already exist in Varys.

The solution for the tables that contain listings of cars and devices need to be styled similarly to the tables in other views.

From the requirement we know that we need at least five components: one for cars, one for chargers, one for load balancers, one for alerts and one for item highlights.

2.2 Constraints

Because the new view will be created within an existing application, it must be visually and functionally compatible with the existing design and layout. We do not have complete creative flexibility in developing the view but must adhere to the application's existing design. The new system is not hardware-dependent, as Varys is a web application that runs in any modern web browser.

2.3 Resources

Tibber has provided each student with appropriate hardware, MacBook Pro M1, to use in the project.

Notion is a project management and note-taking software used for task management. Notion includes kanban boards, tasks, wikis, and databases, as well as modified markdown support. The program provides a one-stop shop for taking notes, managing knowledge and data, and managing projects and tasks. Read more about Notion on their official website: https://www.notion.so/product.

Tibber uses Git and GitHub for version control and for hosting the repository online.

Metabase is an open-source, easy-to-set-up software that connects independent databases. It enables anyone to create intuitive queries without prior knowledge of SQL and to visualize data in a meaningful way. Read more about Metabase on their official website: https://www.metabase.com/product/.

The developers are free to use any integrated development environment (IDE) they prefer; however it should support intelligent code completion and ESLint. VSCode and Webstorm are two popular examples. ESLint is a static code analysis tool that identifies problematic JavaScript code patterns (ESLint, n.d.).

Docker is a platform for developing, shipping, and running applications with the help of containers. A container is a standard software unit that encapsulates code and all of its dependencies so that an application can run quickly and reliably in different computing environments. A Docker container image is a small, self-contained, executable software package that contains everything required to run an application, including code, runtime, system tools, system libraries, and settings (Docker, n.d.). Read more about Docker on their official documentation: https://docs.docker.com/get-started/overview/.

Yarn is a packet manager for your code. Yarn is used in this project to install, operate, and test both the frontend and backend of Varys on a local machine. Read more about Yarn here: https://yarnpkg.com/getting-started.

Postman is an API platform for developers to design, build and test APIs. It is not required for this project, but it is useful for testing and creating automated tests for API endpoints. Read more about Postman on the official website: https://www.postman.com/product/what-is-postman/.

Vue (also known as Vue.js) is a JavaScript framework for building user interfaces. It builds upon standard HTML, CSS, and JavaScript, and provides a component-based programming model which helps to develop both simple and complex user interfaces. Read more about Vue here: https://vuejs.org/guide/introduction.html.

Bootstrap is a popular frontend framework for developing responsive and mobile-friendly websites. It is built upon HTML, CSS and JavaScript. Bootstrap includes user interface components, layouts and JavaScript tools. Read more about Bootstrap here: https://getbootstrap.com/docs/4.6/getting-started/introduction/.

BootstrapVue is a framework based on Bootstrap v4 and is used for creating responsive websites specifically with Vue.js. It is a collection of complete components which makes it easier to use Bootstrap along with Vue. Read the BootstrapVue documentation here: https://bootstrap-vue.org/docs.

CircleCI is a continuous integration and continuous delivery platform that can be used to implement DevOps practices. Read more about DevOps in chapter 3.3 of this paper and more about CircleCI in the official documentation: https://circleci.com/docs/2.0/about-circleci/.

In addition to the software resources listed above, the employees of Tibber are also a resource for this project. There are people working in the smart charging team who are available to answer any questions regarding the smart charging functionality, as well as charger integrations and other features in Varys. The customer support team can assist because they know what common problems and questions the customers have. The product that will be developed during this project is also for the use of customer support, so their feedback is valuable.

2.4 Literature about the problem

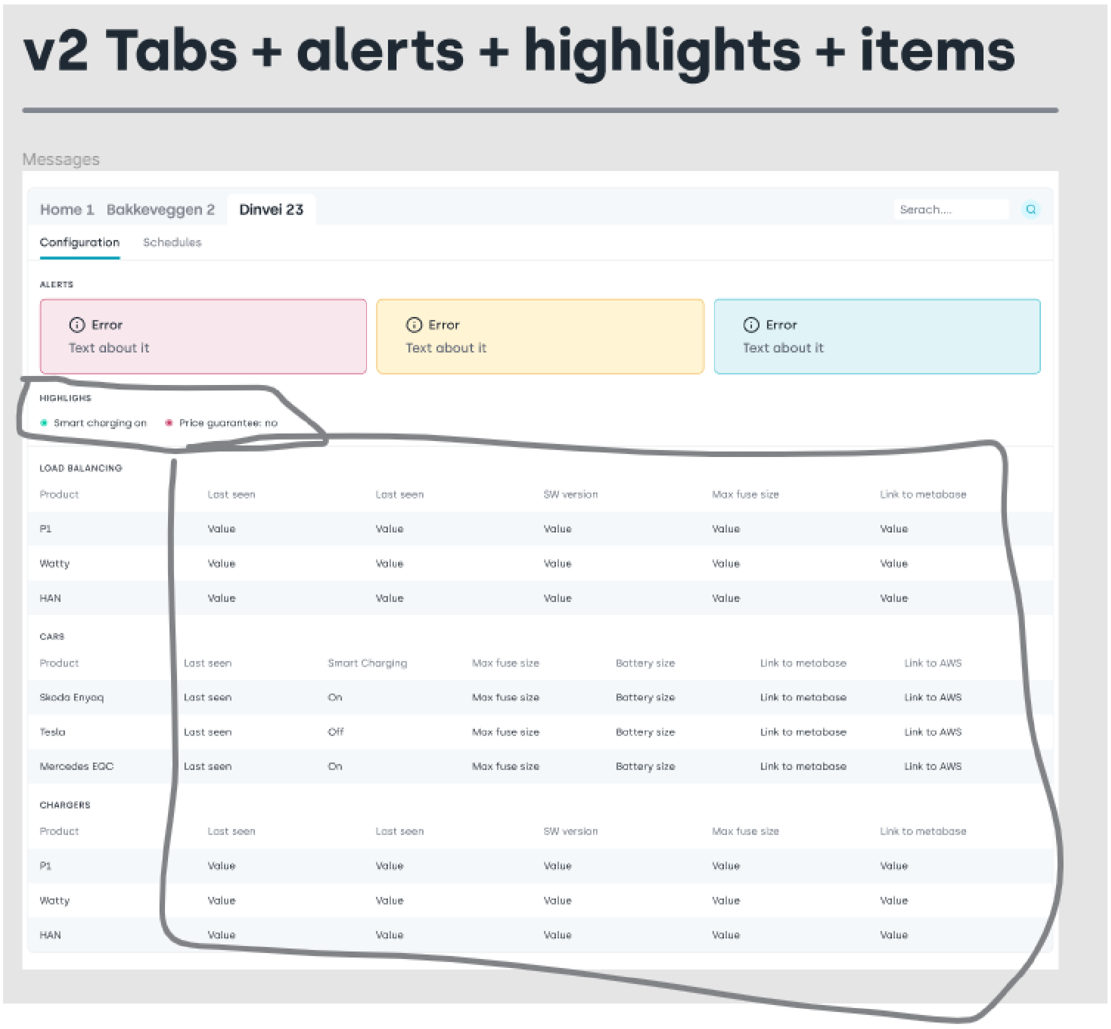
“Leveraging microservice architecture by using Docker technology” by D. Jaramillo, D. V. Nguyen and R. Smart is a paper about microservice architecture and it explains the advantages of using Docker in the implementation (Jaramillo, 2016).

Microservices: Yesterday, Today and Tomorrow is a chapter from *Present and Ulterior Software Engineering* by Dragoni, N. et al. Before describing the current state of the art in the field, this chapter analyzes the history of software architecture and the factors that led to the initial diffusion of objects and services, and then microservices. Finally, open issues and future obstacles are discussed. This poll offers an academic perspective on the problem while primarily addressing newcomers to the field. In addition, a few practical concerns are examined and a few feasible solutions are offered (Dragoni, 2017).

3 PROJECT DESIGN

3.1 Proposed solution

As the new view is to be part of an already existing web application the new view must adhere to existing design. Tibber has put forth a mock-up on how they wish the new view to look (see figure 3.1). The columns are divided into one property per column.



v2 Tabs + alerts + highlights + items

Messages

Home 1 Bakkeveggen 2 Dinvei 23 Serach....

Configuration Schedules

ALERTS

Error
Text about it

Error
Text about it

Error
Text about it

HIGHLIGHS

● Smart charging on ● Price guarantee: no

LOAD BALANCING

Product	Last seen	Last seen	SW version	Max fuse size	Link to metabase
P1	Value	Value	Value	Value	Value
Watty	Value	Value	Value	Value	Value
HAN	Value	Value	Value	Value	Value

CARS

Product	Last seen	Smart Charging	Max fuse size	Battery size	Link to metabase	Link to AWS
Skoda Enyaq	Last seen	On	Max fuse size	Battery size	Link to metabase	Link to AWS
Tesla	Last seen	Off	Max fuse size	Battery size	Link to metabase	Link to AWS
Mercedes EQC	Last seen	On	Max fuse size	Battery size	Link to metabase	Link to AWS

CHARGERS

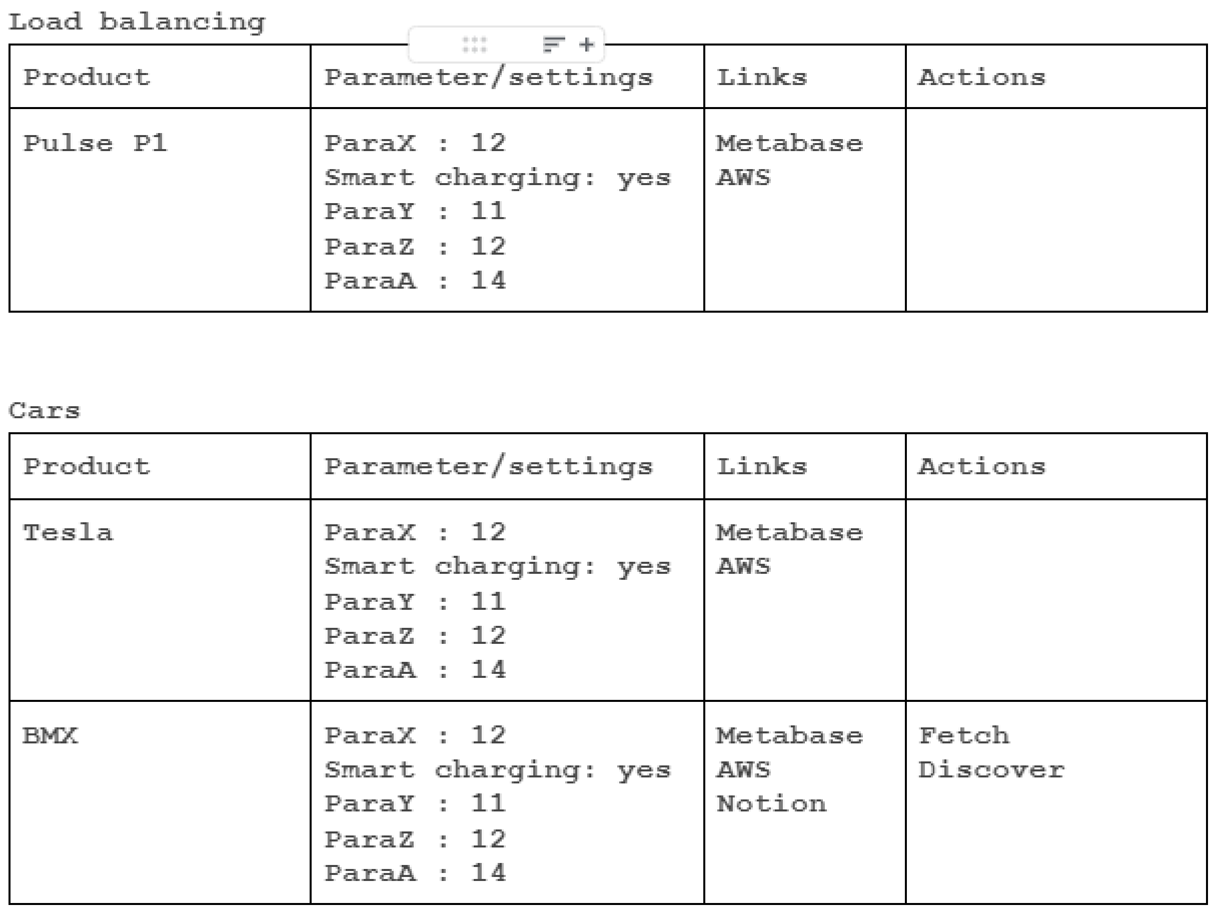
Product	Last seen	Last seen	SW version	Max fuse size	Link to metabase
P1	Value	Value	Value	Value	Value
Watty	Value	Value	Value	Value	Value
HAN	Value	Value	Value	Value	Value

Figure 3.1 Mock-up received from Tibber

The view has the following (from top to bottom):

- Tabs for the customer's home(s)
- Tabs for "Configuration" and "Schedules"
- An alerts section with alert popups

- A highlights section for a quick glance value if the customer is eligible for smart charging and eligible for a "price guarantee"
- A table that lists the customer's load balancer(s)
- A table that lists the customer's car(s)
- A table that lists the customer's charger(s)
-



Load balancing

Product	Parameter/settings	Links	Actions
Pulse P1	ParaX : 12 Smart charging: yes ParaY : 11 ParaZ : 12 ParaA : 14	Metabase AWS	

Cars

Product	Parameter/settings	Links	Actions
Tesla	ParaX : 12 Smart charging: yes ParaY : 11 ParaZ : 12 ParaA : 14	Metabase AWS	
BMX	ParaX : 12 Smart charging: yes ParaY : 11 ParaZ : 12 ParaA : 14	Metabase AWS Notion	Fetch Discover

Figure 3.2 An alternative mock-up.

As well as supplying a mock-up for how Tibber wished the view to look they also have provided a table sketch (see figure 3.2) of how the tables might be arranged, should the option of having one property per column make the view have too many columns if the number of properties shown gets too high.

The multi-column proposal has the inherent advantage of presenting far more fields and actions. The disadvantage is that the rows will be taller as a result. This, however, is irrelevant because the rows will remain very few.
In this scenario, the alternate approach for presenting the table is ideal, as we need to display a large number of properties and actions per row.

3.2 Project methods

3.2.1 Development methods

Several different development methods will be explained in a short and simple manner below. Afterwards, the selected method will be expanded upon, and it will be explained why this method was chosen.

Scrum is an iterative and incremental framework for developing, delivering, and sustaining products in complex environments. It is designed for teams of ten or fewer members, with the work broken down into goals that can be completed within one iteration, called a sprint. A sprint is usually two weeks long. Scrum also includes daily stand-up meetings, which should not last more than 15 minutes. At the end of a sprint, there is also a sprint review which demonstrates the work done to stakeholders to elicit feedback. A scrum team consists of a product owner, a scrum master, and developers. Each role has certain responsibilities. Read more about the Scrum methodology here: https://www.scrum.org/resources/what-is-scrum.

Extreme programming (XP) is an agile process and advocates frequent releases and short development cycles (Wells, 2013). Other elements of extreme programming are extensive code reviews, unit testing of all code and code simplicity. XP is all about the customer's satisfaction and when using this methodology, the team should be able to quickly respond to changes in the requirements at any time in the development cycle.

The Waterfall methodology is a developmental process that flows like a waterfall. It is sequential and each phase is completely finished before moving onto the next phase (Workfront, n.d.). The five stages of the waterfall model are requirements, design, implementation, testing/verification, and deployment. This methodology depends on the belief that all project requirements are understood upfront.

The word DevOps is a combination of the words development and operations. DevOps is a methodology that describes an iterative development process, and it can be visualized as an infinite loop with these phases: plan, code, build, test, release, deploy, operate, monitor and plan (Courtemanche, n.d.). DevOps environments typically implement CI/CD (continuous integration and deployment/delivery), with an emphasis on task automation.

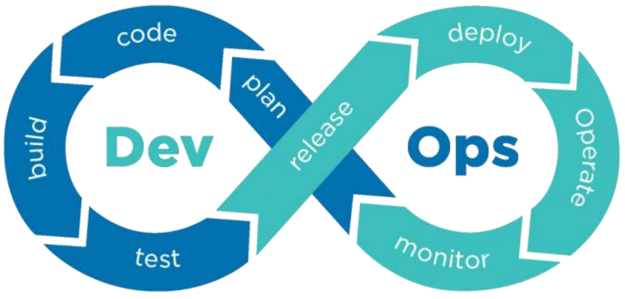


Figure 3.3 DevOps visualized as an infinite loop. Reprinted from Understanding DevOps, In BairesDev (n.d.) Retrieved May 16, 2022, from http://www.bairesdev.com/devops/

Rapid application development (RAD) is both a general term and the name for James Martin's method of rapid development. In general, RAD is an agile development process that puts less emphasis on planning and focuses more on user feedback and adaptive software development (OutSystems, n.d.). RAD is especially well suited for developing software that is driven by user interface requirements. The James Martin approach to the RAD method is divided into four phases: define requirements, make a prototype, absorb feedback, and lastly, finalize the product.

The decision to choose an agile and iterative methodology is based on the nature of the software that is being developed and the usual work style at Tibber. Before starting the development, we receive a Kanban board from Tibber with the wanted features in a loosely prioritized order. However, some of the wanted features are not clearly defined and changes need to be implemented during the process. Therefore, we start with creating a minimum viable product (MVP). This enables us to test the software and explore ideas early on while also receiving feedback from customer service representatives. Once the MVP is in place, iterations on the product continue based on user feedback, adding the nice-to-have features and polish. Several iterations could take place on the same day. Even if a feature is deemed complete, it is still possible to go back and change or improve it.

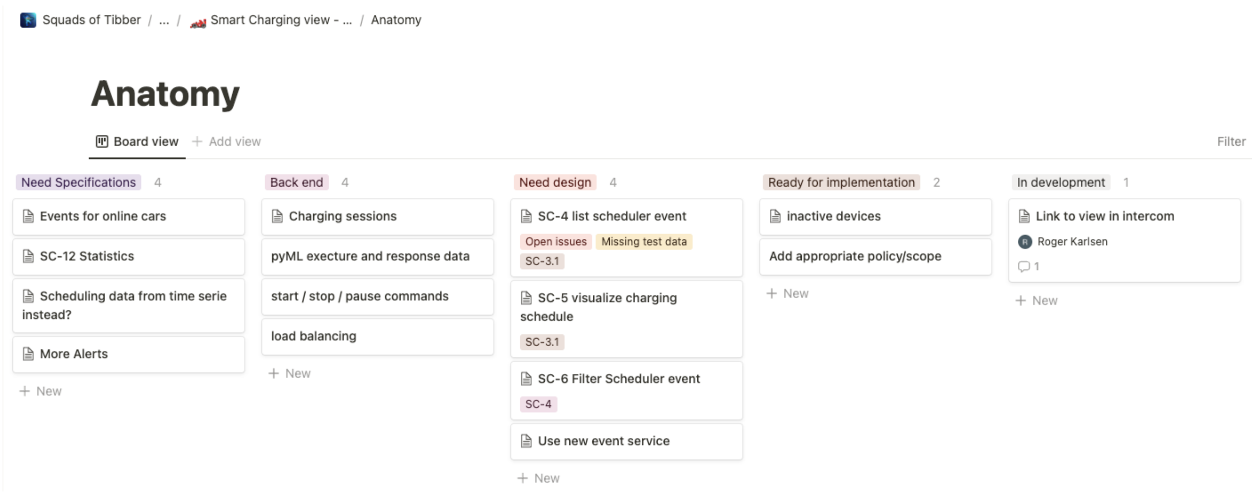
While developing we had progress meetings with Tibber every other Friday to discuss current progress and if there are any challenges in implementation. The time sheets and weekly status reports are presented in the project handbook (see attachment 3).

3.2.2 Project plan

To help keep track of tasks and to give a rough estimate about where development should be in a certain week, a Gantt chart was developed (see attachment 3). This chart was developed with two types of activity: development and report. Tasks that are related to development are marked with a blue color, whilst tasks related to the report are marked in green. By following this diagram, one should easily be able to see if they are behind schedule and in that case be able to mitigate further delays by being proactive and making changes before the time cost becomes too high.

Tibber has also made a Kanban board (see figure 3.4) where they have listed different tasks and wishes for the view. A Kanban board is a type of organizational tool that is used to visualize the various tasks that are needed to get a project to completion. It is usually divided into various lanes where one can sort the tasks in whatever way one sees fit. It is often used in agile development to help prevent bottlenecks by balancing demands with available capacity.

Development consists of continuous iterations (CI/CD), individual effort and daily/weekly sync depending on the task. Initially, the goal is to create a basic skeleton view based on the UI sketch created by the product owner. The data in the various columns does not have to be populated; what is critical is to have an accurate depiction of how the view will appear. Following approval of the view, the job of populating the tables with data can begin.


Squads of Tibber / ... / Smart Charging view - ... / Anatomy

Anatomy

Board view + Add view Filter

Need Specifications 4
- Events for online cars
- SC-12 Statistics
- Scheduling data from time serie instead?
- More Alerts
- + New

Back end 4
- Charging sessions
- pyML execture and response data
- start / stop / pause commands
- load balancing
- + New

Need design 4
- SC-4 list scheduler event — Open issues, Missing test data, SC-3.1
- SC-5 visualize charging schedule — SC-3.1
- SC-6 Filter Scheduler event — SC-4
- Use new event service
- + New

Ready for implementation 2
- inactive devices
- Add appropriate policy/scope
- + New

In development 1
- Link to view in intercom — Roger Karlsen — 1
- + New

Figure 3.4 Kanban board

Apart from the initial work with the skeleton, the most challenging problems are solved first to avoid major refactoring and restructuring down the road. Therefore, navigation/tabs functionality, as well as fetching/caching data, must work correctly and efficiently together. For example, unnecessary calls to endpoints should be avoided where possible and allow for parallel fetching for faster response time. Following that, iterate over this foundation to ensure that all requirements are met.

3.2.3 Risk analysis

In the risk analysis of the project a template of a qualitative risk analysis was used. In this template risks that are thought to have a chance of happening are listed. Then they are assigned with a value of what is thought to be the probability of it happening, on a scale from 1 to 5. Afterwards it is assigned a value of the consequence this risk would entail. And in the end, you multiply the values of the probability and consequence together and get the risk product. Which gives an indicator of the consequence this risk has for the project. See table 3.1 for more detailed information about the risk analysis.

Risk	Cause	Probability	Consequence	Risk product	Mitigation
Application fails to meet requirements	Misunderstanding of requirements. Unable to code required features	Low (2)	Very high (5)	10	Iterative development cycle (agile) where work is presented often
Inaccurate estimations	Unfamiliarity with "proper" software. Under or over estimations might occur which can skew timeframes and deadlines	Medium (3)	Medium (3)	9	Adjust the Gantt-diagram as necessary
Change of scope	Change of scope, be it from going in a different direction than thought, both from the developer's side or from a directive of the stakeholders	Very low (1)	High (4)	4	Adhere as closely as possible to the original goals set
Failing to meet stakeholder expectations	Skills / knowledge in the tools not up the standard expected of the students	High (4)	High (4)	16	Crash courses on for example Udemy or ask for help.
Low quality of code	No uniformity or previous experience in working on a big project together. Code might suffer as a consequence.	Medium (3)	High (4)	12	Review pull-requests together and comment on bad code / improvements
Low productivity	Being stuck in a task and uncertain on how to proceed, and the more time passes the more anxious one gets, leading to a vicious circle. Low interest in the project	Medium (3)	Very high (5)	15	Open and honest communication between students. Reserve judgment
Lack of ownership	No one takes responsibility ensuring that deadlines and milestones are met and celebrated	Very high (4)	Medium (3)	12	Assign tasks to individual students in the Gantt-diagram and follow up each other.
Getting Corona	Pandemic	High (4)	Very high (5)	20	Follow guidelines from FHI

Table 3.1 Risk analysis.

3.3 Evaluation plan

As soon as the minimum viable product is achieved the application will be released to the customer support department. From that point, there are two types of evaluation that will take place.

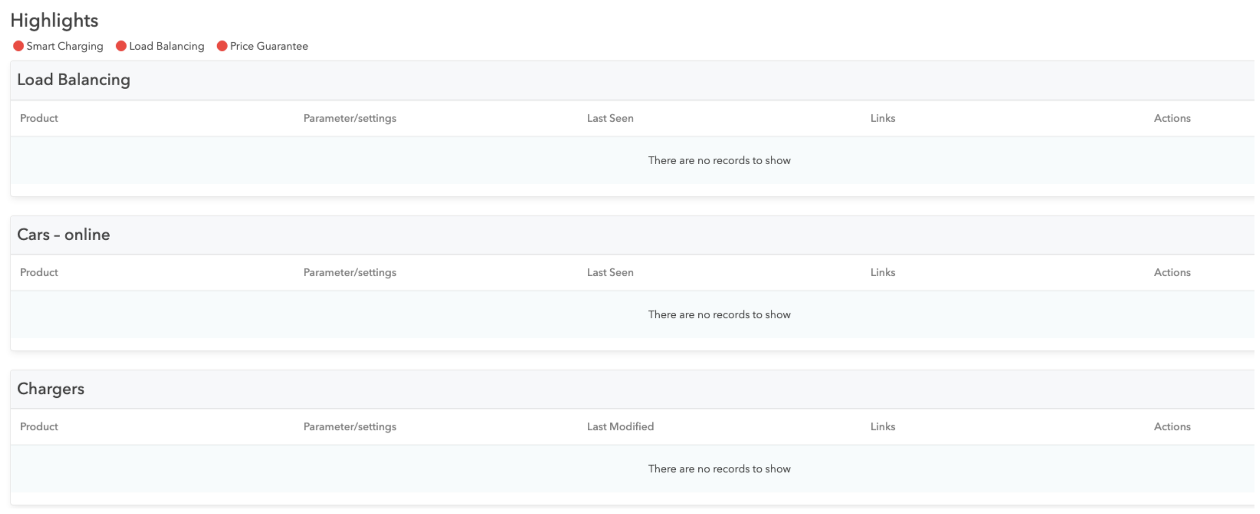
The first type of evaluation will be a simple statistical system which will save metadata of the visit by the support personnel. A basic definition of metadata is that metadata are data that describe other data (Dippo, 2000). The plan is to collect information about which customer was visited (and which house if the customer has multiple homes), which customer support user visited, and how many and what type of errors were present at the time of visiting. This can then be used by the managers to track how the new view is being used and assess which types of errors are more common.

Tibber already has in place a database for storing usage or statistics from other Varys views and actions. Markus Persson, the product manager at Tibber responsible for Varys, made this when he became responsible for Varys. The system was made to try to give an overview over what was done via Varys. To be able to improve workflows and the robustness of the system they concluded that they must have some data from the system to be able to understand how it is being used. This database is a relational PostgreSQL database. The data will be stored in an already existing table used to track usage throughout Varys. The data will be stored as JSON object literals.

The second type of evaluation is user feedback. As soon as the product is given to support personnel a page with the option to request changes or features will be posted in Notion. Here they will be able to give feedback on what's working, what's not working and if there are any bugs. After the view has been available to the support department for a while, they will receive a questionnaire containing questions regarding the new view. That they can answer on a rating scale from "Strongly Disagree" to "Strongly Agree". This will help by giving an indicator on how the product is received and how useful it is to the support personnel.

4 Solution

As Varys is made in Vue.js, it is natural to continue using Vue. We started with setting up the various components we were going to need, the index being the most obvious. As explained in the "Previous work" section, Vue pages are put together by components, so therefore it is natural to sketch out what kind of components we need for the index view. For the minimum viable product for the index, we knew we needed at least five components: one for the car table, one for the charger table, one for the load balancing table, one for the alerts and one for the highlight section.


Highlights

● Smart Charging ● Load Balancing ● Price Guarantee

Load Balancing

Product	Parameter/settings	Last Seen	Links	Actions
There are no records to show				

Cars - online

Product	Parameter/settings	Last Seen	Links	Actions
There are no records to show				

Chargers

Product	Parameter/settings	Last Modified	Links	Actions
There are no records to show				

Figure 4.1 The initial three empty tables for the view.

Before we started filling the page with real data, a skeleton had to be put in place first. This means getting the three different tables up and filling them with dummy data. While filling them with dummy data we quickly saw that the proposed solution about having one column per property would cause too much clutter. We opted then for the solution of having multiple properties in a shared column. After the tables were in place, a highlights component was made that will at one point in the future show customer support if the customer is eligible for Smart Charging and if the customer is eligible for something Tibber calls "price guarantee".

The alerts section could not be started before getting data into the view. After the three main tables were in place it was time to start fetching data. This is accomplished by retrieving data from the Device Orchestrator service. We can query and retrieve the majority of the data we require through this service.

Conveniently, the DO API is built on the ASP.NET framework, which means it comes pre-loaded with swagger. Swagger generates documentation for all API endpoints automatically, which simplifies the process of getting an overview. Additionally, it is exportable (JSON) and can be imported into Postman, which simplifies testing.

Testing is accomplished by making calls to relevant endpoints with the required query parameters and/or JSON data and assessing whether the response/data returned is as expected/sufficient.

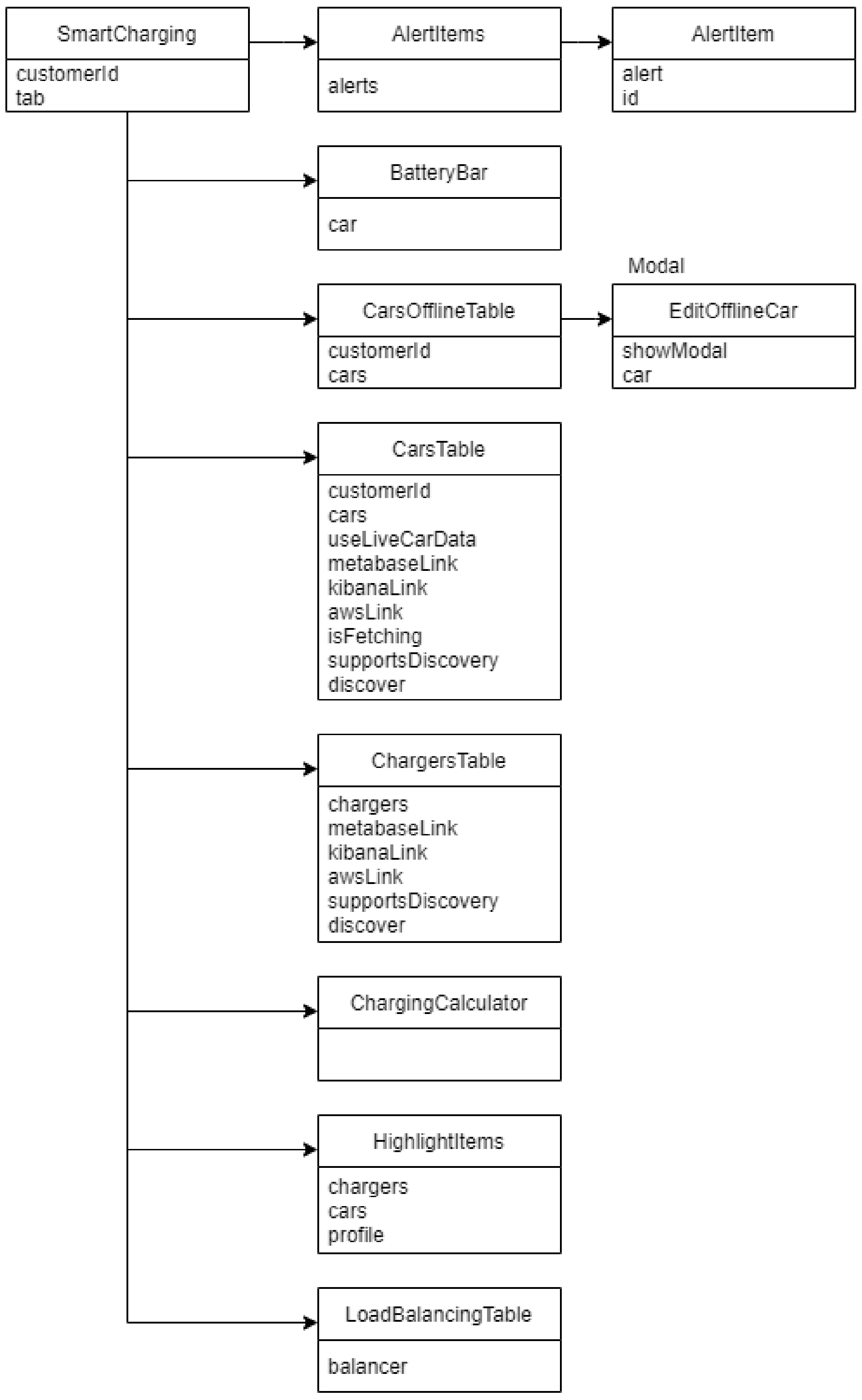
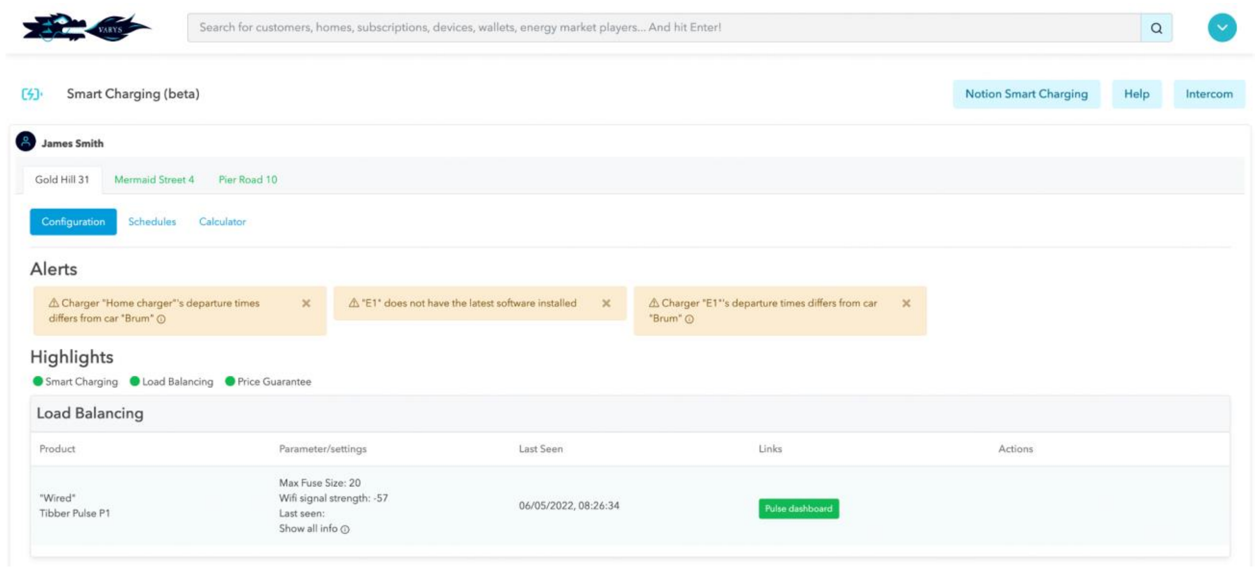


Figure 4.2 Smart Charging view components and their inherited properties.

Now the endpoints need to be implemented on the Varys backend server to be prepared for usage in Vue, the client side of the application. An endpoint on the backend is created such that it prepares an object that is sufficient and usable. A single backend endpoint often does multiple requests to a single or multiple services, sometimes in parallel when it’s possible to optimize the response time.



Alerts

⚠ Charger "Home charger"'s departure times differs from car "Brum" ⓘ

⚠ "E1" does not have the latest software installed

⚠ Charger "E1"'s departure times differs from car "Brum" ⓘ

Highlights

● Smart Charging ● Load Balancing ● Price Guarantee

Load Balancing

Product	Parameter/settings	Last Seen	Links	Actions
"Wired" Tibber Pulse P1	Max Fuse Size: 20 Wifi signal strength: -57 Last seen: Show all info ⓘ	06/05/2022, 08:26:34	Pulse dashboard	

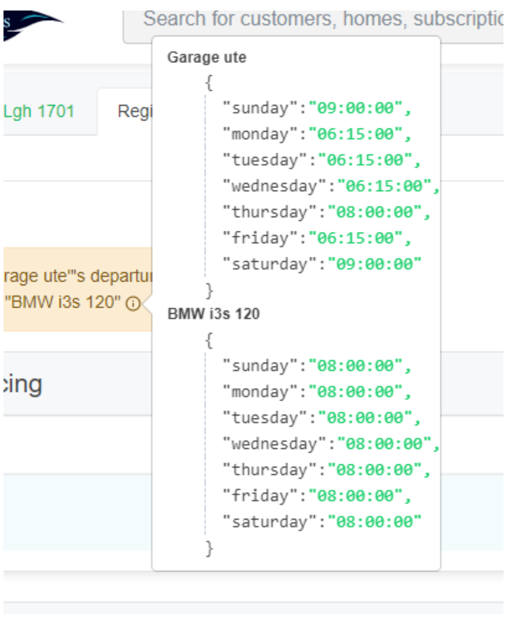
Figure 4.3 Navigation, alerts, highlights, and various configuration tables.

Navigation

Navigation is made intuitive through the use of tabs. To help the client and servers save some resources, each customer's home page includes its own lazily fetched page. This means that the data and UI for each home are only fetched and rendered when the client explicitly requests them, i.e., clicks a tab.

Alerts (Green: “Good”, Orange: “Warning”, Red: “Error”)

The alert messages serve as a detailed summary; they also assist the support user with quick troubleshooting by automatically generating messages based on available devices and their configuration state. Customer support agents can hover over the *‘i’-icon* to obtain additional information about the alert.

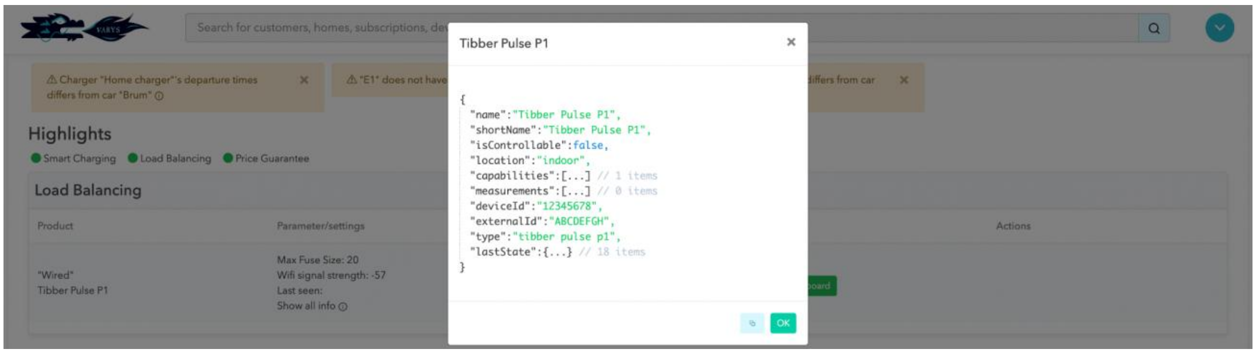


Garage ute
```
{
  "sunday":"09:00:00",
  "monday":"06:15:00",
  "tuesday":"06:15:00",
  "wednesday":"06:15:00",
  "thursday":"08:00:00",
  "friday":"06:15:00",
  "saturday":"09:00:00"
}
```
BMW i3s 120
```
{
  "sunday":"08:00:00",
  "monday":"08:00:00",
  "tuesday":"08:00:00",
  "wednesday":"08:00:00",
  "thursday":"08:00:00",
  "friday":"08:00:00",
  "saturday":"08:00:00"
}
```

Figure 4.4 Alert example where there is a misconfiguration.

Highlights
Similar to the preceding, but shorter. Green indicates that something is fully functional, for example, that Smart Charging is properly configured and expected to operate. Yellow alerts the support personnel to the possibility of an issue, typically due to ambiguous data. Red indicates that it is not properly configured or that it is missing required devices.

Configuration state tables
Feature a summary of devices with their type, name (if available) and most relevant configuration state variables. It also provides links to dashboards and available quick actions like start, stop or edit configuration. The rest of the configuration variables are available through a JSON-document with expandable properties by clicking *"show all info"* (see figure 4.5).

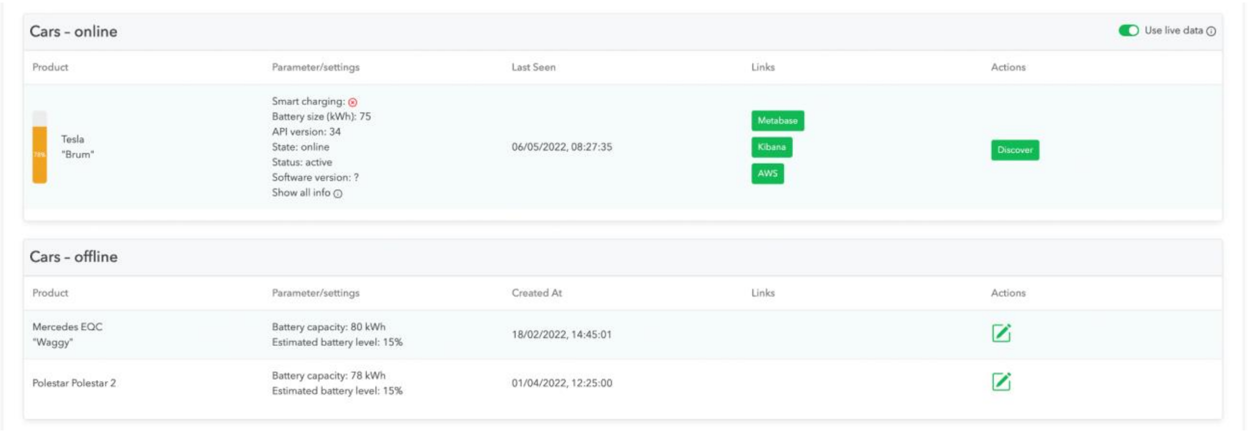


```
{
  "name":"Tibber Pulse P1",
  "shortName":"Tibber Pulse P1",
  "isControllable":false,
  "location":"indoor",
  "capabilities":[...] // 1 items
  "measurements":[...] // 0 items
  "deviceId":"12345678",
  "externalId":"ABCDEFGH",
  "type":"tibber pulse p1",
  "lastState":{...} // 18 items
}
```

Figure 4.5 JSON-modal that shows all info.

Online cars include battery and Smart Charging state. The orange bar on the left shows the charge state of the battery. It is orange to indicate that smart charging is disabled, otherwise it would be green. If it is currently charging, it will do an animation.

Retrieving live data from a vehicle could take up to 20 seconds. This was perhaps a result of having to wake up the car from sleep. To avoid a negative user experience, the interface includes a switch labelled "use live data" that enables the fetching of live data in the background while maintaining a responsive UI. Additionally, a pulsing circle indicator is added to indicate that data is being fetched (see figure 4.7).



Cars - online

Use live data

Product	Parameter/settings	Last Seen	Links	Actions
Tesla "Brum"	Smart charging: ⊗ Battery size (kWh): 75 API version: 34 State: online Status: active Software version: ? Show all info	06/05/2022, 08:27:35	Metabase Kibana AWS	Discover

Cars - offline

Product	Parameter/settings	Created At	Links	Actions
Mercedes EQC "Waggy"	Battery capacity: 80 kWh Estimated battery level: 15%	18/02/2022, 14:45:01		✎
Polestar Polestar 2	Battery capacity: 78 kWh Estimated battery level: 15%	01/04/2022, 12:25:00		✎

Figure 4.6 Table for the cars.

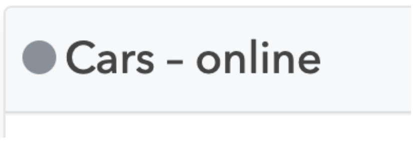
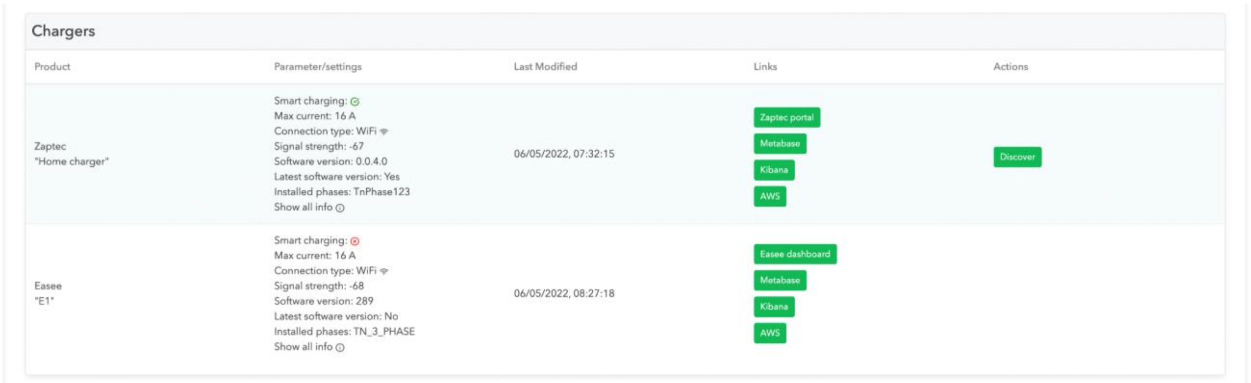


Figure 4.7 Loading Indicator.

Certain vehicles are not yet supported by the app due to the lack of APIs or the time required for in-house reverse engineering. Customers may still add their cars as “offline” where they manually set their charge status to still take advantage of a more primitive Smart Charging. These cars appear under the offline cars table.

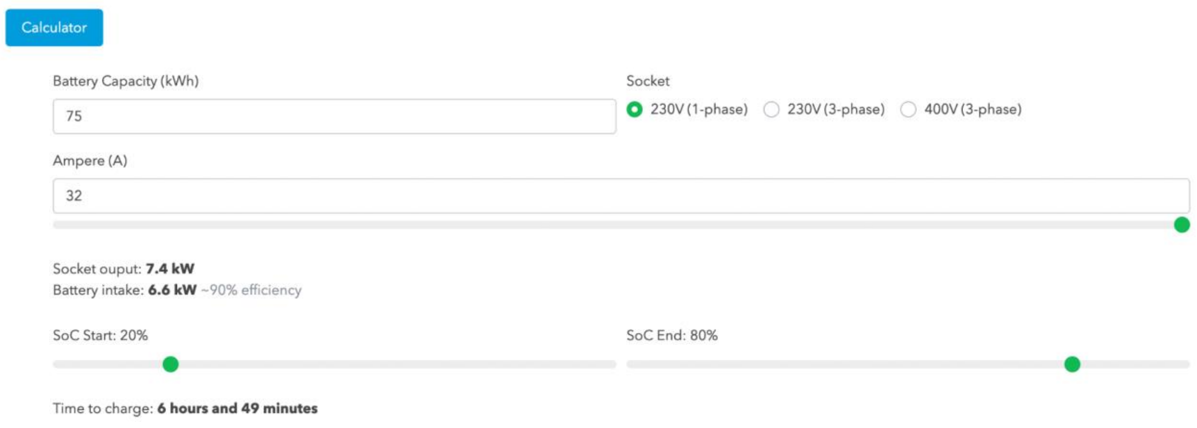
The chargers table (figure 4.8) shows information pertaining to the charger(s). If the customer has activated smart charging for the product a green icon checkmark is placed next to the “Smart charging” property in the “Parameter/settings” column, if it’s not activated a red x-icon is shown instead. This gives a quick glance value for the customer support person showing if the property is active or not. Other than that, it shows the max current for the charger, the connection type, which is either cellular or WiFi. If it has both types of connections, it will show WiFi, and its signal strength, measured in RSSI (Received Signal Strength Indicator). The software version property fetches the device’s installed software number and checks if it has the latest software installed. This is then displayed in the “Latest software installed” with a yes / no indicator. Should the customer support person not find the information they’re looking for in the column, they have the option of pressing the information-icon (i) which will display the entire JSON object literal.


Chargers

Product	Parameter/settings	Last Modified	Links	Actions
Zaptec "Home charger"	Smart charging: ✓ Max current: 16 A Connection type: WiFi Signal strength: -67 Software version: 0.0.4.0 Latest software version: Yes Installed phases: TnPhase123 Show all info ⓘ	06/05/2022, 07:32:15	Zaptec portal Metabase Kibana AWS	Discover
Easee "E1"	Smart charging: ⊗ Max current: 16 A Connection type: WiFi Signal strength: -68 Software version: 289 Latest software version: No Installed phases: TN_3_PHASE Show all info ⓘ	06/05/2022, 08:27:18	Easee dashboard Metabase Kibana AWS	

Figure 4.8 Table for the chargers.

Based on battery capacity, socket type, and current, the charging calculator (figure 4.9) provides an estimate of charging time. Additionally, it accounts for energy loss due to connection friction. The tool is currently quite basic, but it can be easily expanded to include features such as generated profiles and default settings based on the customer's available devices and configuration.


Calculator

Battery Capacity (kWh)

75

Socket

● 230V (1-phase) ○ 230V (3-phase) ○ 400V (3-phase)

Ampere (A)

32

Socket ouput: **7.4 kW**
Battery intake: **6.6 kW** ~90% efficiency

SoC Start: 20%

SoC End: 80%

Time to charge: **6 hours and 49 minutes**

Figure 4.9 Charging Calculator.

5 Results

5.1 Evaluation method

For the evaluation of the final result, we expanded upon an already existing metadata logger in Varys. This let us make metadata about the visit the person working in customer support had. It currently logs the following:

- **userId** - The ID of the customer support person, everyone in Tibber has their own unique Varys user ID
- **customerId** - The ID of the customer, every customer of Tibber has their own unique Varys customer ID
- **pageView_smartCharging** - A flag for the log entry that the entry has metadata regarding the smart charging view, so it can be filtered out from other log entries pertaining data about other views.
- **createdAt** - Timestamp of the visit
- **Metadata** - A JSON Object Literal containing the following:
    - homeId - the customer's home ID where all related devices are listed
    - Errors - A number of how many errors (from the alerts) where shown for this home
    - Warning - A number of how many warnings (from the alerts) where shown for this home
        - For example:
          {"home": "a3f446c6-787e-4ff2-b80d-8ffb6b794c8z", "errors": 1, "warnings": 1}

When the smart charging view is released for the customer support department one can track visits to the page, how many errors and warnings were present.

Another evaluation method is to send out a questionnaire to the customer support team. They are the ones who will use this new software the most, and therefore their feedback is valuable. They have the knowledge needed to verify if the finished product is satisfactory. Therefore, a

Likert scale survey was created. A Likert scale survey uses rating scale questions, which are one of the most common kinds for surveys that require respondents to rate the performance of a product or service, staff abilities, customer experience, and so on (QuestionPro, n.d.).

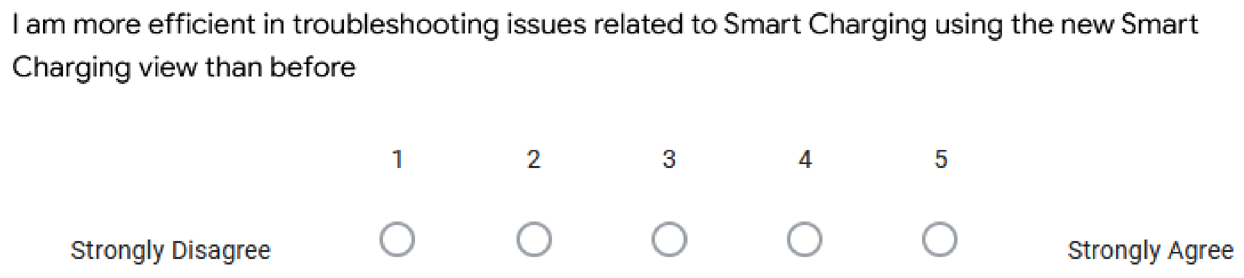

I am more efficient in troubleshooting issues related to Smart Charging using the new Smart Charging view than before

Figure 5.1 Example of a rating scale question.

The questionnaire included the following questions with a rating scale from "strongly disagree" to "strongly agree". All questions are obligatory to answer.

1	I am more efficient in troubleshooting issues related to Smart Charging using the new Smart Charging view than before.
2	It is easy to use the Smart Charging view.
3	In the Smart Charging view, it is easy to get the information I need for troubleshooting Smart Charging issues.
4	The Smart Charging view enables me to quickly troubleshoot issues regarding Smart Charging.
5	The information in the Smart Charging view is well structured.

Table 5.1 Survey questions.

In addition to the questions, an optional text box with the opportunity to write other types of feedback was also included.

5.2 Evaluation result

From the statistics tool that was developed we are able to monitor how much the support team is using the application. Figure 5.2 shows the initial days after release (including weekend days) and unique visits. The big spike on Tuesday April 19th was when the view was released to the support team. Please note that the number of unique visits mean that a user has used the view one or more times during a day.

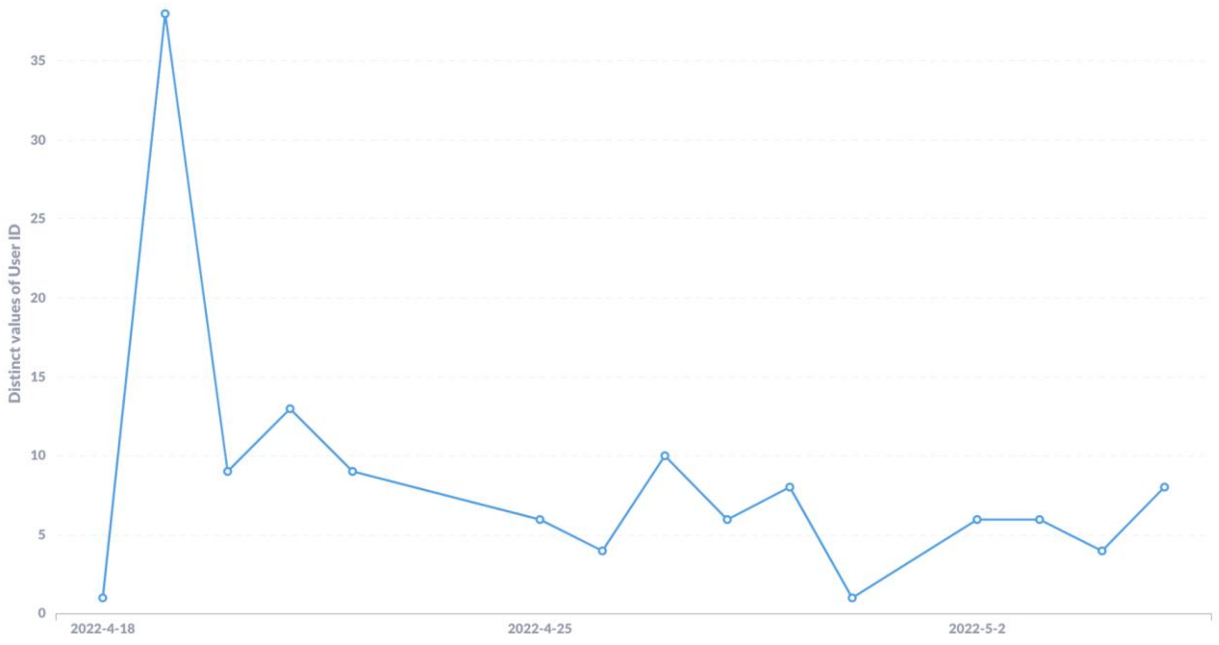


Figure 5.2 Number of unique users per day

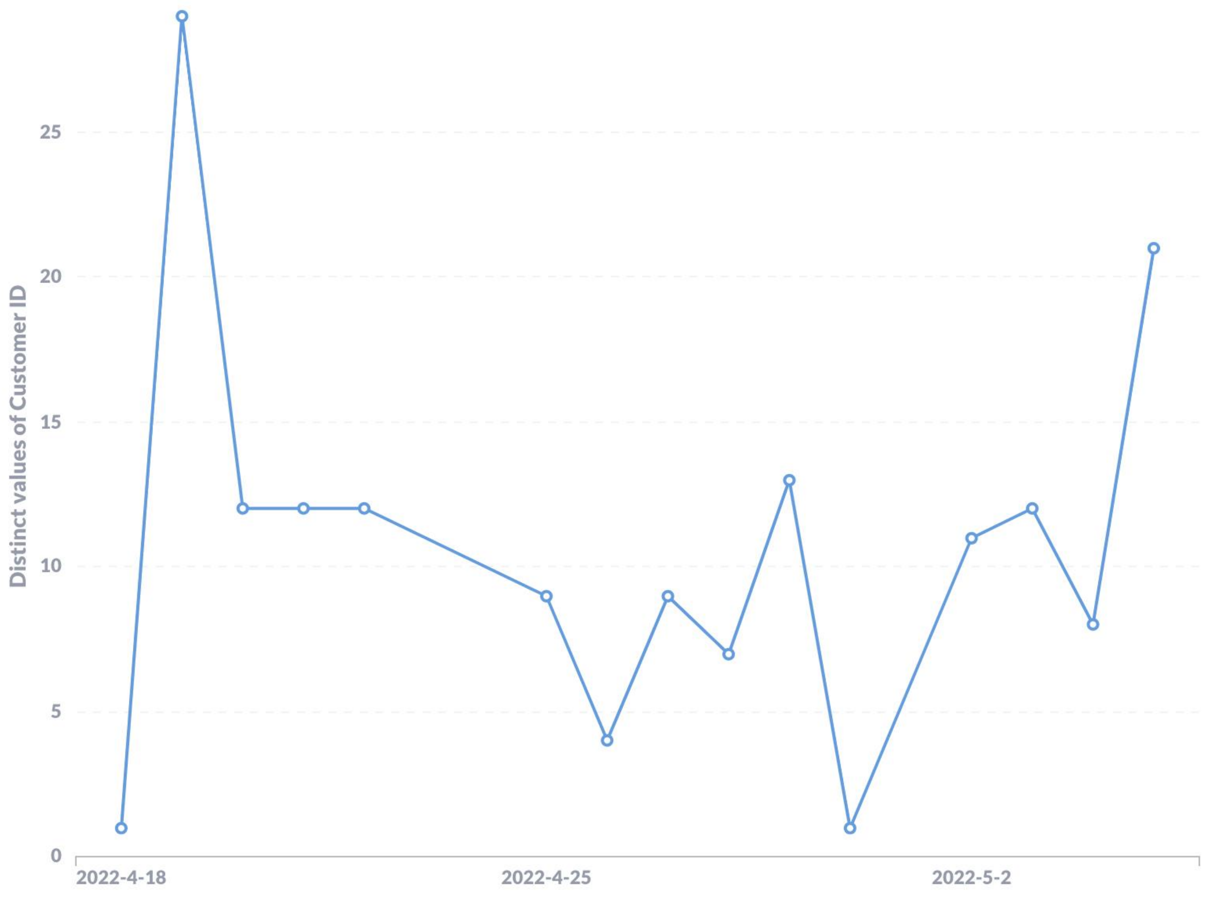


Figure 5.3 Number of unique customers that have been looked up.

In figure 5.3 you can see the number of unique customers per day the support personnel have visited in the smart charging view. Excluding weekends, this shows that there is a steady flow of customers that need assistance with smart charging.

The following graphs show the results for the customer support questionnaire.

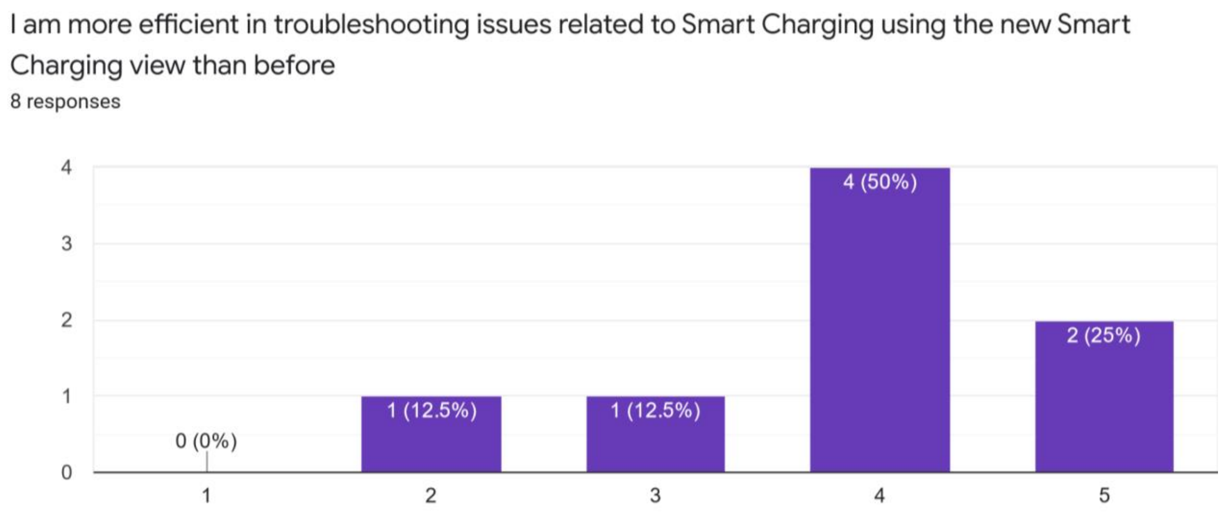


Figure 5.4 Results for question 1

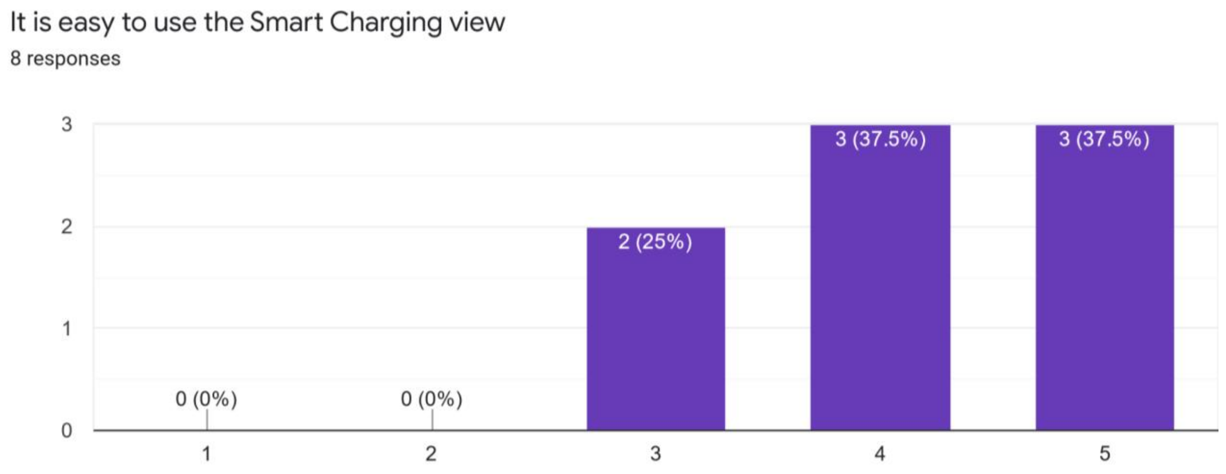


Figure 5.5 Results for question 2

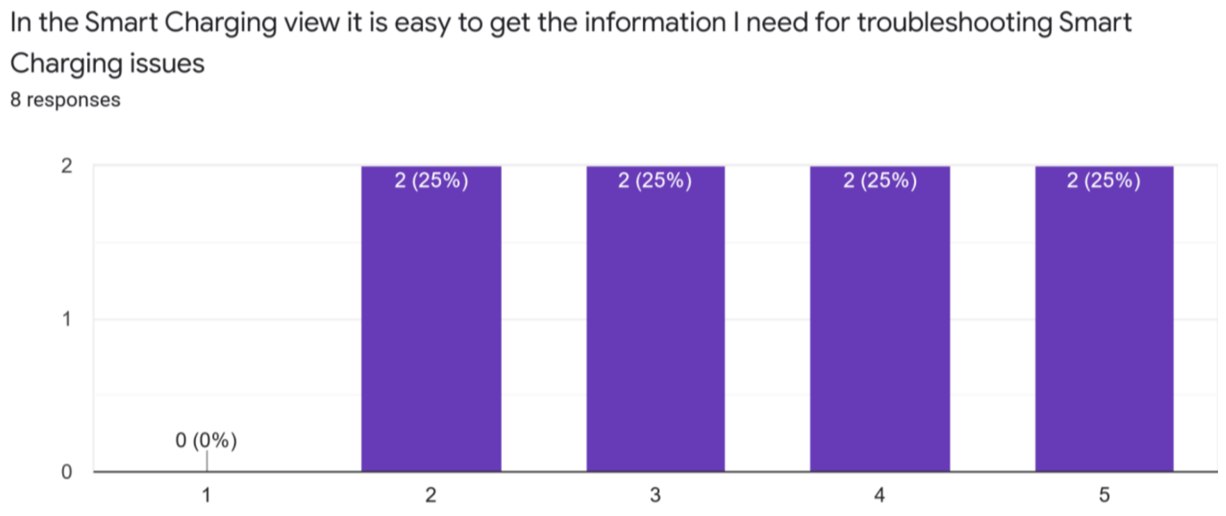


Figure 5.6 Results for question 3

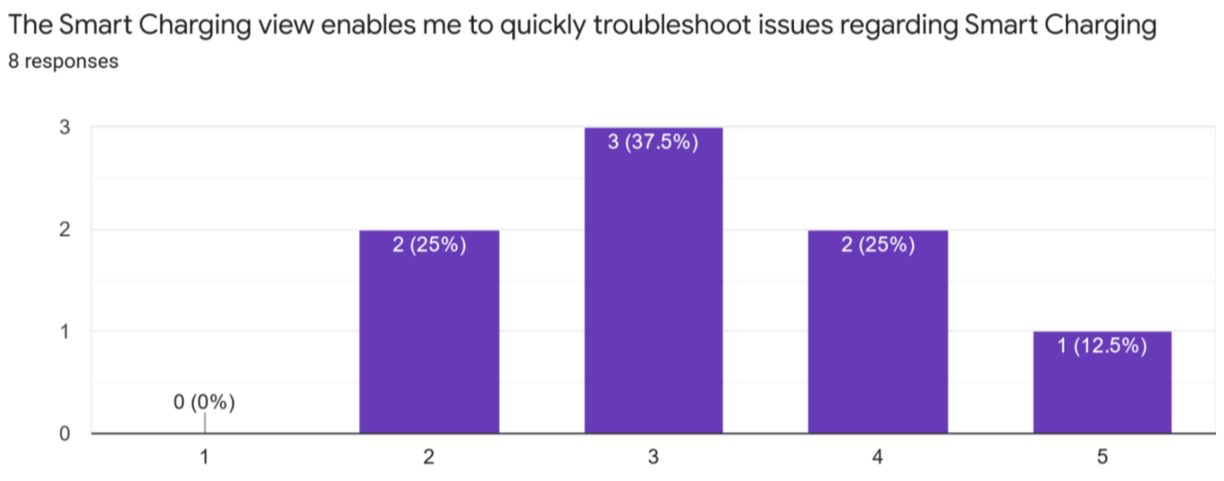


Figure 5.7 Results for question 4

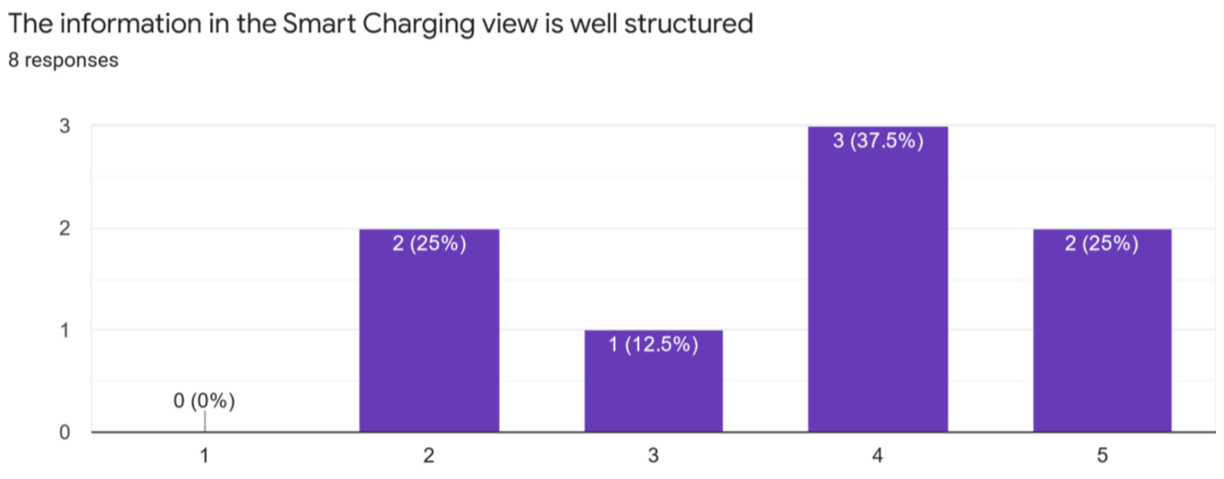


Figure 5.8 Results for question 5

5.3 Project result

After finishing the project Tibber is left with a view the support department can make use of. Support personnel are now able to find information about various devices related to smart charging in this new view. They get alerted if there are any configuration errors, whereas earlier they would have to find the information manually and conclude for themselves if there were any errors present.

Verifying if customers are eligible for price guarantees or smart charging no longer requires that support personnel manually locate the device and identify the charger type. This information is now instantly available in each customer’s smart charging view.

5.4 Project implementation

We started this project early in January 2022. In the beginning most of the time was spent learning about Vue and what micro services are. After we felt we had a good grasp we started planning for the view. From Tibber's requirements and mock-ups we roughly knew what we needed to do first, which was making a basic skeleton of the view with empty tables.

After getting the skeleton up and running in a way that conformed with the overall design of Varys we started on the next part of the project: fetching data to populate the tables. As the data was requested from the Device Orchestrator service, we spent some time learning and using Postman to figure out which endpoints we needed to use for our view.

Fetching and separating the data received into the different tables was what occupied us for a while. As the various device information gets returned as JSON Object Literals, we needed to figure out which properties Tibber wanted to display in the tables.

After we had fetched and prepared the data for the tables, we started working on the various configuration checks, one for the alerts and one for the highlights section.

During implementation of the vehicle table, it was discovered that retrieving live data could take up to 20 seconds as a result of having to wake the car from sleep. To avoid a negative user experience, a switch labelled "use live data" was added. This option enables the fetching of live data in the background while maintaining a responsive UI. Additionally, a pulsing circular indicator is added to illustrate that data is being fetched.

Before the minimum viable product was achieved a metadata logging system was put in place on the view, as per a request from Tibber. This enables them to monitor how the view is used and gives valuable feedback on how the workflow in Varys is as a whole.

After the view was released to the customer support department, time was spent fixing bugs and trying to implement other features.

6 Discussion

Working on the project felt efficient and frictionless thanks to a good framework, tools, and development methods. You can always deliver something with few obstacles if you make small code contributions and deploy frequently. Because the most recent code is always out, frequent deployment leads to more testing. If something breaks, it is usually related to the most recent code changes, which can be easily pinpointed, saving time and frustration. Also, seeing the newest changes and additions immediately after writing the code, makes the work more rewarding. It also makes it relatively easy to quickly see if it works or if it needs to be done in another way. That is why an agile methodology works really well in this scenario. There is always the possibility to go back and make changes to the code. If we had chosen to use the waterfall methodology, we would not be as flexible to adjustments.

Icons and other graphics are an important factor in a large number of user interfaces, visually representing objects, actions, and concepts. When used properly, they effectively communicate the fundamental idea and intent of a product or action while also providing numerous benefits to user interfaces, such as saving screen space and improving visual appeal.

The result from the customer support team questionnaire was as expected. Unfortunately, the questionnaire was sent out late and we did not receive as many responses as we wanted. However, the Norwegian customer support team at Tibber consists of 14 members, so the eight responses we received can be considered to sufficiently represent the whole team.

When discussing the evaluation results below, the questions will be referred to by numbers. The numbers will correspond with the table below.

1	I am more efficient in troubleshooting issues related to Smart Charging using the new Smart Charging view than before.
2	It is easy to use the Smart Charging view.
3	In the Smart Charging view, it is easy to get the information I need for troubleshooting Smart Charging issues.
4	The Smart Charging view enables me to quickly troubleshoot issues regarding Smart Charging.
5	The information in the Smart Charging view is well structured.

Table 6.1 The questions from the customer support questionnaire

For us, the first question is the most important one. The purpose of this project is to make the job easier for the customer support team and make them more efficient. The result for question 1 shows that a majority of the respondents, 6 out of 8, agree that they are more efficient when using the new Smart Charging view. We can assume that this is because the data is presented in a different and more optimal way than before.

Question 2 and 3 deal with ease of use. The view is simple and easy to understand, but still leaves something to be desired. While 6 out of 8 respondents think that the view is easy to use, the results for question 3 are more spread out. Even if the view is easy to use for most of the respondents, there are still improvements to be made when it comes to actually retrieving the information needed. 2 out of 8 slightly disagree that it is easy to obtain the information needed for troubleshooting smart charging issues, while 50% agree with the statement. Hopefully, this is something that can change for the better over time as the users get to know the view. Of course, the view was created to contain only the necessary information needed to answer questions about smart charging and to present this data in an intuitive way, but perhaps it is not yet perfected. The feedback from the customer support team will be taken into consideration to further develop the view.

However, the results for question 4 show that improvement is needed. For the view to be considered efficient, we want the users to be able to troubleshoot issues in a quick manner. Again, this might be something that can improve in the future as the team members get used to the view, but there are also possible changes that can be made to make the users more comfortable and able to quickly troubleshoot issues. In this case, their feedback and suggestions are welcome so that the view can be improved.

The last question (5) is about the structure of the information presented in the view. With this question, we wanted to know if the information presented in the view is the necessary information, and also if it is structured in a way that makes sense for the customer support members. This is one of the attributes we want every user to totally agree with, but there seems to be varied opinions. Some things are easier to realize after the product is developed and released and after using the product for a while, one notices things that would benefit from being done in a different way. The great thing about choosing an agile development method is that changes can easily be made late in the development cycle. Even if this project is complete for us, the developers at Tibber will continue to work on this product by improving it and implementing new features. They will hopefully take advantage of the feedback from the customer support team and use it to make the product even better than it is today.

As noted in section 3.4.2 Project plan, we received a Kanban board from Tibber with an overview over wanted features. Due to time constraints, we were unable to implement all of these features. However, the current product includes the most important features, and it will act as a starting point for the development team at Tibber as they plan on further developing the view.

If we could redo this project with the knowledge that we have now, we would try to complete and release a minimum viable product earlier, so that the customer support team could play a bigger role earlier in the development. As it was, the product was released too late to receive feedback and properly act on it. If we obtained the feedback earlier, perhaps we would have the time to make significant changes and the results from the evaluation survey would be more positive, i.e., the respondents would agree to the different statements to a higher degree. The idea of the survey as an evaluation method came late, so it was prepared in a hasty manner. With more time, the survey could perhaps include additional, more detailed questions. It could also have been nice to send out the survey early, act on the feedback, and then ask the customer support team to answer the survey again. It would be interesting to see the differences in the results after making changes based on their feedback.

7 Conclusion and further work

At the start of the project, we sought to answer how the various events and data streams involved in charging an electric vehicle can be visualized in such a way that non-technical support personnel can easily and efficiently assist customers. At the end of the project, we have developed a view within a web application that we think solves this problem. Because of the positive feedback from the project owner, the product manager, and the customer support team at Tibber, we are confident that the initial goal has been reached.

The customer support team gave mainly positive feedback, while there are things that can be improved. According to the statistics, the new view is being used on a daily basis by support personnel. As the project went on, ideas for new features surfaced. While we could not manage to implement all these features during the course of the project, the development team at Tibber will keep working on the Smart Charging view. They will tweak the already existing features, while implementing new ones. A new service that will be used to show “charging events” is already in the works.

All in all, based on words from the project owner, Marcus Almgren, and the product manager, Markus Persson, we believe that the product fulfills the requirements that were made at the beginning of the project and that we have given Tibber a satisfactory product that they can continue to develop.

8 Sources

Courtemanche, M., Mell, E. and Gillis, A.S. (n.d.) What is DevOps? The Ultimate Guide. Available at: https://www.techtarget.com/searchitoperations/definition/DevOps (Retrieved May 16, 2022).

Dippo, C.S. and Sundgren, B. (2000) *The Role of Metadata in Statistics*. Statistical Survey Paper. Washington, DC: Bureau of Labor Statistics.

Docker (n.d.) What is a Container?. Available at: https://www.docker.com/resources/what-container/ (Retrieved May 16, 2022).

Dragoni, N. et al. (2017). Microservices: Yesterday, Today, and Tomorrow. In: Mazzara, M., Meyer, B. (eds) *Present and Ulterior Software Engineering*. Springer, Cham. doi: 10.1007/978-3-319-67425-4_12

EnergyStart (n.d.) About Energy Efficiency. Available at: https://www.energystar.gov/about/about_energy_efficiency (Retrieved Apr 4, 2022).

ESLint (n.d.) Getting Started with ESLint. Available at: https://eslint.org/docs/user-guide/getting-started (Retrieved May 15, 2022).

Jaramilli, D, Nguyen, V, and Smart, R. (2016) *Leveraging microservices architecture by using Docker technology*. SoutheastCon 2016. doi: 10.1109/SECON.2016.7506647

OutSystems. (n.d.) What is Rapid Application Development? Available at: https://www.outsystems.com/glossary/what-is-rapid-application-development/ (Retrieved May 16, 2022).

QuestionPro (n.d.) Rating Scale: Definition, Survey Question Types and Examples. Available at: https://www.questionpro.com/blog/rating-scale/ (Retrieved May 7, 2022).

Tibber (n.d.) About Us. Available at: https://tibber.com/en/about-us (Retrieved Feb 15, 2022).

Wells, Don. (2013) Extreme Programming: A Gentle Introduction. Available at: http://www.extremeprogramming.org/ (Retrieved May 16, 2022).

Workfront. (n.d.) Waterfall Methodology. Available at: https://www.workfront.com/project-management/methodologies/waterfall (Retrieved May 16, 2022).

9 Attachments

1. Vision Document
2. Requirements Document
3. Project Handbook
4. System Documentation