UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Implementing a new DevOps service: configurable monitoring solution for data visualization among teams

Bachelor's Thesis Report

Software Engineering

Luz Emma Pereira

Facultat d'Informàtica de Barcelona

Director: Allan Pagaard Hansen

Ponent FIB: Ernest Teniente

Denmark, June 2022

Abstract

Whenever a company works on software development, especially when that work involves embedded software that will go into a physical product, a lot of data and information is generated. This may come from the day-to-day work, from testing done on running software, from results from pull requests when merging code...

Furthermore, *DevOps* practices are becoming increasingly important within organizations, to the point where many firms have their own dedicated teams for this. And, among their many tasks, a *DevOps* team strives to give support to developers and help achieve fast and continuous integration.

Developers at Danfoss spend a lot of time checking many information sources every day. Sometimes, even, data is missed or forgotten about because it is all separated and out of reach. The goal of this project is to design and implement a new service composed of an information screen incorporating different sources of data, which developers need to access every day, giving them the opportunity to customize and select the important information for their own team. This report aims to show that the easy access to this data through the info screen will streamline its analysis and increase overall efficiency of the developer teams.

Resum

Sempre que una empresa treballa en el desenvolupament de programari, especialment quan aquest treball implica programari incrustat que anirà a un producte físic, es generen moltes dades i informació. Això pot provenir del treball diari, de les proves fetes amb programari en execució, dels resultats dels *pull requests* en combinar el codi...

A més, les pràctiques de *DevOps* són cada vegada més importants dins de les organitzacions, fins al punt que moltes empreses tenen els seus propis equips dedicats a això. I, entre les seves moltes tasques, un equip de *DevOps* s'esforça per donar suport als desenvolupadors i ajudar a aconseguir una integració ràpida i contínua.

Els desenvolupadors de Danfoss passen molt de temps comprovant moltes fonts d'informació cada dia. De vegades, fins i tot, les dades es perden o s'obliden perquè estan totes separades i fora de l'abast. L'objectiu d'aquest projecte és dissenyar i implementar un nou servei compost per una pantalla d'informació que incorpori diferents fonts de dades, a les quals els desenvolupadors han d'accedir dia a dia, donant-los l'oportunitat de personalitzar i seleccionar la informació important per al seu propi equip. Aquest informe pretén mostrar que el fàcil accés a aquestes dades a través de la pantalla racionalitzarà el seu anàlisi i augmentarà l'eficiència global dels equips de desenvolupadors.

Resumen

Siempre que una empresa trabaja en el desarrollo de software, especialmente cuando este trabajo implica software incrustado que irá en un producto físico, se generan muchos datos e información. Esto puede provenir del trabajo diario, de las pruebas realizadas con software en ejecución, de los resultados de los *pull requests* al combinar el código...

Además, las prácticas de *DevOps* son cada vez más importantes dentro de las organizaciones, hasta el punto de que muchas empresas tienen sus propios equipos dedicados a ello. Y, entre sus muchas tareas, un equipo de *DevOps* se esfuerza por apoyar a los desarrolladores y ayudar a conseguir una integración rápida y continua.

Los desarrolladores de Danfoss pasan mucho tiempo comprobando muchas fuentes de información todos los días. A veces, incluso, los datos se pierden o se olvidan porque están todos separados y fuera del alcance. El objetivo de este proyecto es diseñar e implementar un nuevo servicio compuesto por una pantalla de información que incorpore diferentes fuentes de datos, a las que los desarrolladores deben acceder día a día, dándoles la oportunidad de personalizar y seleccionar la información importante para su propio equipo. Este informe pretende mostrar que el fácil acceso a estos datos a través de la pantalla racionalizará su análisis y aumentará la eficiencia global de los equipos de desarrolladores.

Contents

Chapter 1

Introduction

1.1 Contextualization

This project *Implementing a new DevOps service: configurable monitoring solution for data visualization among teams* is my Bachelor's Thesis in the field of Software Engineering for the Barcelona School of Informatics (FIB), conducted during my internship at Danfoss Drives.

Danfoss is a multinational danish company based in Nordborg, Denmark, founded in 1933, with more than 40.000 employees worldwide. They are an engineering company serving many markets, such as the automotive industry, refrigeration and air conditioning, energy and natural resources, among many others. Focusing on quality, reliability and innovation, they deliver an extensive range of products and solutions across their three business segments: Danfoss Climate Solutions, Danfoss Drives and Danfoss Power Solutions.[1]

My internship is at the Danfoss Drives segment, where they produce drives for different usages, like improving up-time in production facilities or minimizing how much energy is used. These drives are devices that control speed, direction, acceleration, deceleration, torque, etc. of the system they are connected to, which is usually an electric motor in a factory.[2] And within the organization, I am a part of the *DevOps* team (Development and Operations), that follow a set of practices focusing on giving support to other developers and ensuring fast and continuous integration, while ensuring software quality. The focus of my internship within the team is around a project that aims to create a new tool for developers: an information screen that incorporates different sources of data accessed daily by developers, with the opportunity for them to customize and select the most relevant information for their team.

1.2 Identifying the problem

The *DevOps* team, among many other areas they support, have a specific set of services that they have implemented in order to make their everyday tasks easier and also for other developers in the company. Some of these include: a Pipeline Analyzer, a test database that provides an infrastructure for developers to create reports and notifications around the success rate of any given test, a resource monitoring solution to visualize the usage of resources...

They also have a big amount of information and data, such as test results, pipeline runs, pull requests and dashboards, that is updated regularly, and each of these is relevant for different people. Everyday they work with different data visualization tools, such as *Azure DevOps*, which is a service used for version control, project management, etc., and *Power BI* or *Grafana*, which are tools used to visualize and analyze data.

All of this data can be very useful for different purposes, such as seeing the status of a pull requests, checking test results... but it all depends on the team and what their tasks are. So the relevance of this information varies for each group, but having access to it in an organized and easy way can be very beneficial in the everyday work of developers.

Furthermore, as you walk around the buildings where software developers work in Danfoss Drives, you find many big screens in the rooms, and most of them are not being used at all. These have been there for a while, in hope that they can show some info graphics at some point, but they never had a defined purpose.

This is where the idea to monitor all of this data through these screens came from: the *DevOps* team saw an opportunity for a new service to provide to both their own team, and also other developer teams in Danfoss Drives, that would allow them to, firstly, combine all of these different sources of information, and secondly, to display it in a customized info screen, according to each team's needs and preferences.

This project was planned to be developed based on an existing internal website they have, which is where they provide some of these previously mentioned services (such as the *Daily report* page or the *Pipeline Analyzer*). The website is implemented using the *ASP.NET* framework, which allows the development of web apps using as programming languages C sharp, HTML, CSS and JavaScript. And for the storage of data, it is connected to a data base also developed in the *ASP.NET* framework. So the development of my solution would be connected to their existing website, as an extension of this one.

1.3 Stakeholders

Since this project is destined to be used by a specific group of people, and is internal within the company, the stakeholders involved are few. So these are the following groups or individuals that are affected in some way by the making of the monitoring solution.

- **DevOps team members:** the members of my team are the ones that are the most involved in the project, since they will be users of the info screen and follow the development of the project closely, constantly sharing their insights and opinions to guide the project.
- **Other teams in the company:** These will also be users of the solution, configuring their own customized screen with their desired information displayed on it.
- **Individual developers in the teams:** the project is made so that, if possible in the end, the screen will also allow individual developers in the company to

use it for themselves, and not just as a team solution. So they have to be considered for future implementation so that the info screen will also be useful for them individually.

- **Allan Pagaard Hansen:** My supervisor and director of the project, he is the one who guides the direction of the project, and its scope and usage.
- **Philip Bang Hellesoe and Jacob Norrelykke Iversen :** They are two developers in the DevOps team who implemented their existing website and know the most about the infrastructure and framework around it, so they are my first point of contact when it comes to implementing the info screen, as they have knowledge to share and will usually be able to help me.
- **Developers:** in this case, I will be the only one implementing this project by myself, with the support of my supervisor and team.
- **Danfoss:** the company itself, since it provides the resources for me to develop my project.
- **Ernest Teniente, UPC supervisor:** he will be my supervisor from the university side, ensuring that the project is developed and documented in the right way to present my bachelor's thesis.

1.4 Definition of concepts

- **Drive:** a drive is a device that controls speed, direction, acceleration, deceleration, torque, etc. of the system it's connected to, usually an electric motor in a factory. The drive, by continually calculating and adjusting the frequency and voltage, makes sure that the motor receives only the power it needs, allowing lower energy costs, lower maintenance costs, and a smaller carbon footprint. There are many types of drives, used for different needs.
- **DevOps:** DevOps (from development and operations) is a set of practices based on the union of people, process, and technology to continually provide value to customers. DevOps enables formerly siloed roles —development, IT

operations, quality engineering, and security— to coordinate and collaborate to produce better, more reliable products. By adopting a DevOps culture along with its practices and tools, teams gain the ability to better respond to customer needs, increase confidence in the applications they build, and achieve business goals faster.[3]

- **Pipeline:** A *DevOps* pipeline is a set of automated processes and tools that allows both developers and operations professionals to work cohesively to build and deploy code to a production environment. A pipeline typically includes build automation/continuous integration, automation testing, validation, and reporting.[4]

- **Pull request:** A pull request, also referred to as a merge request, is an event that takes place in software development when a contributor/developer is ready to begin the process of merging new code changes with the main project repository.[5]

- **Azure DevOps:** *Azure DevOps* is a service offered by Microsoft that provides many tools, such as version control, reporting, requirements management, project management based on agile methodologies, automated builds, testing and release management capabilities.[6]

- **Power BI:** *Power BI* is a collection of software services, apps, and connectors that work together to turn unrelated sources of data into coherent, visually immersive, and interactive insights. Power BI lets you easily connect to your data sources, visualize and discover what's important, and share that with anyone you want.[7]

- **Grafana:** *Grafana* is an open source software service that allows analyzing and monitoring data, creating customized graphs with information from different sources, such as data bases, or other software services.[8]

- **ASP.NET:** An open source framework by Microsoft for building web apps and services using *.NET* and *C sharp* as the main programming language.[9]

- **User Story:** A user story is a general, informal explanation of a software feature written from the perspective of the end user. Its purpose is to articulate how a software feature will provide value to the customer.[10]
- **Primary Key** In a SQL data base, a primary key is the column or columns that contain values that uniquely identify each row in a table.[11]
- **Foreign key:** In an SQL data base, a foreign key is a field (or collection of fields) in one table, that refers to the primary key in another table.[12]
- **JSON file:** JavaScript Object Notation (JSON) is a standard text-based format for representing structured data based on JavaScript object syntax. It is commonly used for transmitting data in web applications.[13]

Chapter 2

Justification and Scope

2.1 Justification

For a while now, many rooms in the Danfoss Drives buildings have had laying around these big screens that are mostly unused, as it has been mentioned in the previous section. So the chance to give them a utility made the *DevOps* team come up with the idea of building this *Info Screen* solution. The need for this project was justified since, although they had already built some tools to visualize existing data, they needed something that would combine all different sources of information (*Power BI*, *Grafana*, *Azure DevOps*...) and allow users and teams to put these together in one big screen. So the initial idea was to implement this solution based on their existing internal website, called the *Viking DevOps* website, where they display daily reports with different data and metrics, among other things, so that it is possible for me to reuse some of this if I find it helpful, but it should be built as an independent project, with freedom to decide on the design and implementation of the monitoring solution.

But my project differentiates itself from their existing website in several ways. Firstly, the *Viking DevOps* website provides daily reports that are set and cannot be changed, whereas the info screen would allow each team to customize their own configuration. Secondly, my solution would allow users to combine different sources of information that they check regularly, with the possibility to change their configuration at any time, as well as adding new pages and components to their screen, changing which pages are displayed and for how long, deleting them or also modifying them.

This whole idea is represented in the following figure 1, which visualizes - without going into too much detail - what the monitoring solution is meant to achieve, and what this includes: a member of a team can configure their team's info screen, which is made of different pages, that they also create, and each of these is made of page components, which can be of different types, such as graphs from several sources or text areas. And the solution is displayed on the screen in the buildings, but it can also be opened in a browser on the user's laptops.

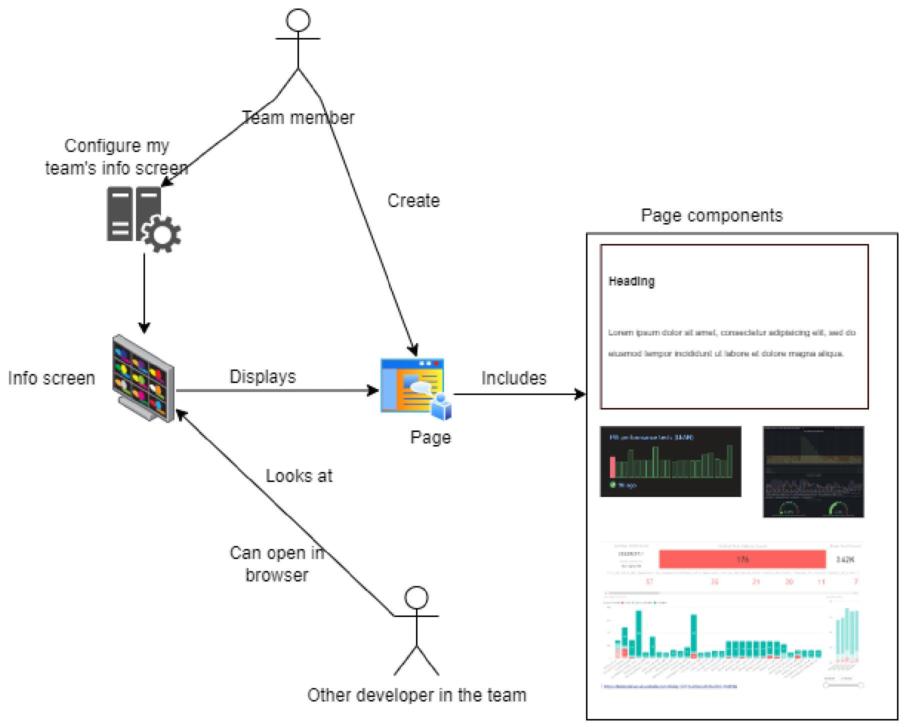


Figure 1: Monitoring solution. Own creation

2.2 Alternatives considered

When the *DevOps* team came up with the idea of this project, they considered different alternatives to solve the issue at hand, or existing tools that could help.

Danfoss already has in many of its building more big screens that do display some information. These screens are bought from a different company that provides the service of configuring them with the information Danfoss wants to display in them. So my team decided to try this and see if it was of any use to them, but after testing it, the main issue that was found, as it has been mentioned previously, is that this service did not allow them to combine data from many of the tools they use (such as *Power BI* graphs, among others), since these graphs could not be embedded into the page and mixed together on the screen. So they realized that it was necessary to develop their own service to be able to display all the information they wanted, so that their solution could be used by many interested teams, not just for themselves.

In terms of the architecture chosen to implement the info screen, the main reason why it was decided that I would use the *ASP.NET* framework, just like they do for the *Viking DevOps website*, is so that I could take advantage of the products they use in the company (they already use Microsoft products for most of their tools), and possibly re-use some parts or the structure of their website. It also made the most sense since it would avoid extra costs to the company to use a different product, and there could be compatibility issues since the information that is meant to be displayed on the info screens is mainly from other Microsoft products (such as *Power BI* graphs, *Azure DevOps* information...). Furthermore, *ASP.NET* is a great framework for building dynamic web pages. It allows the combination of several programming languages for the front-end code (HTML, CSS and JavaScript) and for the back-end (mainly C sharp).

2.3 Objectives and sub-objectives

The main goal of this project, as it has been described in the previous sections, is to create a monitoring solution that will allow teams and users to configure their own screens, combining data from different sources and tools, so that it can be displayed in each building accordingly. Each team should be able to customize their screen, with the possibility to create, modify and delete pages to display, select which ones will appear when the screen is opened, for how long, and which page components to include, among other functionalities. This solution should be implemented so that users will be able to use it for a long time, and it should also allow adding further functionalities in the future and expanding its use, since the project will continue even after I finish my internship, and there are already more ideas for its future scope, so it needs to be a scalable product.

If we go a bit further into the project's goals, we can identify several sub-objectives:

- Combining all existing data sources and visualization tools (*Azure DevOps*, *Power BI*, *Grafana*...) into one common place.
- Allowing each software team to choose which information they want to check daily, according to their needs and preferences, and which part is more important, or should be displayed for longer.
- Promoting the use of the many solutions and tools (such as the *Pipeline Analyzer*, a *Grafana* dashboard that displays the latest test results...) that the DevOps team offers to other developers: by giving teams this configurable screen, it is giving visibility to all the services that they already have, and can help other developers on their daily tasks, which they might not be aware of. For example, a developer from a software production team, might want to check the status of his recent pull requests, add a comment (a text note) next to another graph and check this every morning for several days in a row. Then, some days later, they might want to change the main

screen and display a different component, because the relevance for them has changed.

- Offering other teams (such as software production, UI/UX team...) and the *DevOps* team itself a new tool to make their work easier, by facilitating the visualization of relevant data.

2.4 Requirements

The following are the requirements that the project should comply with, differentiating between functional requirements, which represent the project's features that developers will have to implement, and non-functional requirements, which establish the criteria that dictates how the system should work when achieving the different functionalities. The functional requirements will be further specified in the next chapter.

2.4.1 Functional requirements

- Users with a Danfoss identification and within the teams that the info screen is directed for should be able to use the solution by being authenticated automatically.
- Each user should be able to access and visualize their team's info screen when entering the URL where this is displayed.
- Users in the different teams should be able to modify their team's info screen configuration on the settings page.
- Users should be able to create new pages for their info screen, as well as modify and delete these.
- Users should be able to create new page components for their info screen.

Based on these functional requirements, I will specify the main functionalities of the project. These are grouped into four blocks, depending on their value for each section. The functionalities are the following:

- **The main Info screen:** all of those related to the display of the teams' info screens.
    - Create different layouts for the info screen.
    - Display the user's info screen.
- **The Info screen's configuration:** all of those related to the settings page of the info screens.
    - Display the user's info screen's settings.
    - Saver a user's chosen configuration for their info screen.
- **Page Configuration:** all of those related to the creation, modification and elimination of a page configuration.
    - Create a new page configuration.
    - Modify an existing page configuration.
    - Delete a page configuration.
    - Preview a new or modified page configuration.
- **Page Component:** all of those related to the creation, modification and elimination of a page component.
    - Create a new page component.

2.4.2 Non-functional requirements

- Usability: the info screen's configuration should be easy to use by any user that has access to it, without prior experience necessary.
- Availability: the info screen should be able to be accessed and visualized at any point.
- Security: there needs to be a good security level for the project, since it treats and displays internal information of the company.

- Scalability: the system should be able to remain effective even if the number of resources, users or functionalities increases in the future.

2.5 Potential risks

In any software development project, there is a set of potential risks that could appear along the way that is good to take into account throughout the project planning. In this case, the following risks have been identified:

- **Bugs and errors:** this is one of the most common risks in any software development project. Bugs or errors can appear in different situations, such as throughout the implementation of the code, when merging different branches, etc., and they can cause a delay on the planning.
- **Deadline of the thesis:** Seeing as the delivery of this thesis has a set and specific date (as well as the different deliverables for it), time is crucial and, in case of delays or complications during the project's development, it is important to adapt the planning to these deadlines, since they cannot be extended.
- **Data privacy of Danfoss:** This project is based around internal data and information of the company, Danfoss, so this must be taken into account when sharing information to other institutions, such as my university, ensuring that all the necessary permission is granted.
- **Thesis abroad in Denmark:** When developing a project in a different country, I might bump into differences between my supervisor in Danfoss (in Denmark) and my professor and tutor from UPC (Spain).
- **Online meetings with UPC supervisor:** Since the thesis is being conducted in Denmark, as mentioned above, all my meetings with my supervisor from UPC will be done online, which can challenge communication sometimes.

- **Inexperience in some tools or programming language:** the info screen's website for displaying and visualizing it will be done using ASP.NET framework and the code will be written in C sharp mainly, and I will also be using *Azure DevOps* for the project management side, as well as obtaining data from other sources (*Power BI*, *Grafana*...). Since I am not fully familiar with most of these, there may be some difficulties using them, specially at the beginning of the project.

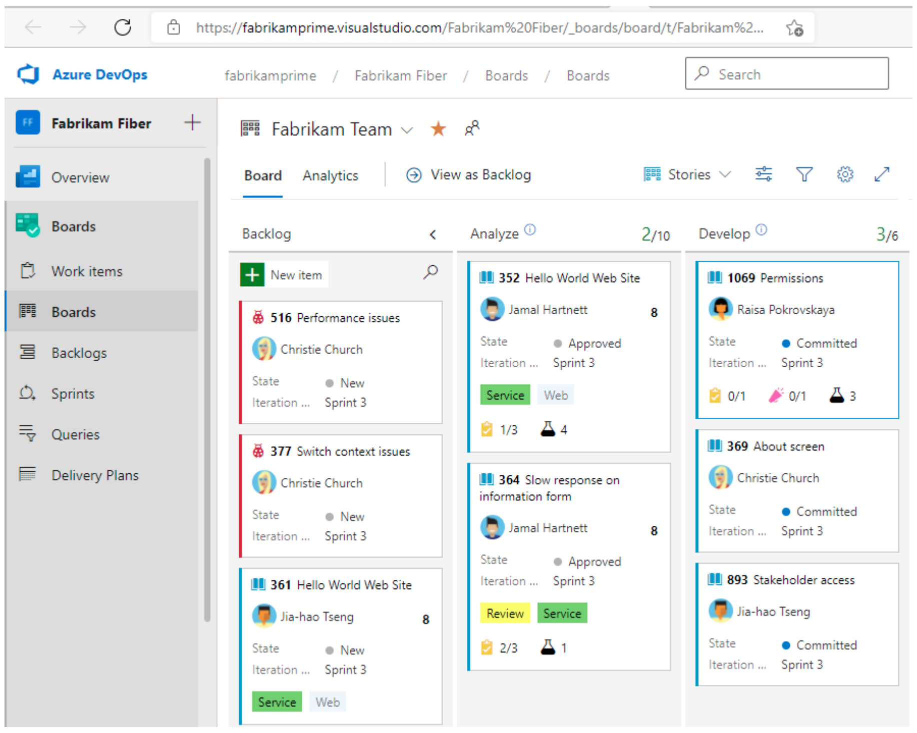
Chapter 3

Methodology

When I joined the *DevOps* team at Danfoss Drives, I was slowly introduced to their working methodology, which is based around *Scrum* and *Agile*, and also around *DevOps* practices. During the first weeks of my internship, I started with more freedom and flexibility in terms of my working methodology, but as the tasks got more specific and I was more familiar with the tools they were using, my project has been planned so that it will follow the *Scrum* methodology as well.

There are many existing tools to help developers use *Agile* methodologies, but the one that the *DevOps* team uses is *Azure DevOps*. *Azure DevOps* is a service offered by Microsoft that provides many tools, such as version control, reporting, requirements management, project management based on agile methodologies, automated builds, testing and release management capabilities. The project management tool is used daily to organize the team's tasks following the *Scrum* methodology.

The following figure 2 is an example of what the *Azure DevOps* Boards tool looks like, which allows team members to do many things in terms of the project management, such as create user stories, tasks, bugs, as well as add tags, indicate the state (to do, in progress, done...), estimate hours or effort for each of them, and assign them to the team members.



516 Performance issues
Christie Church
State New
Iteration ... Sprint 3

377 Switch context issues
Christie Church
State New
Iteration ... Sprint 3

361 Hello World Web Site
Jia-hao Tseng 8
State New
Iteration ... Sprint 3
Service Web

352 Hello World Web Site
Jamal Hartnett 8
State Approved
Iteration ... Sprint 3
Service Web
1/3 4

364 Slow response on information form
Jamal Hartnett 8
State Approved
Iteration ... Sprint 3
Review Service
2/3 1

1069 Permissions
Raisa Pokrovskaya
State Committed
Iteration ... Sprint 3
0/1 0/1 3

369 About screen
Christie Church
State Committed
Iteration ... Sprint 3

893 Stakeholder access
Jia-hao Tseng
State Committed
Iteration ... Sprint 3

Figure 2: Example of an Azure Board. Source: [14]

So as seen below in figure 3, *Scrum* is based on *sprints*. A *sprint*, in the *DevOps* team, lasts 14 working days, around which tasks are selected from the *product backlog*, so that these will be completed during this amount of time. Each big task is defined as a *User Story*, which usually has smaller sub-tasks and is assigned to one or several team members, to complete during the *sprint*.

The *Scrum* methodology takes into account different actors: the *Scrum Master* (SM), who guides the meetings, manages the processes and makes sure possible problems are solved; the *Product Owner* (PO), who communicates the team's product goals and the *product backlog* items, and the Development team, which is made up by 6 people in this case.

During a *sprint*, there are several ceremonies or meetings that take place with the team: the *Sprint Planning*, the *Daily Scrum*, *Refinement*, the *Sprint Review* and the *Sprint Retrospective*.

- **Sprint Planning:** This meeting takes place at the beginning of every sprint and it usually lasts between one and two hours, during which the team chooses

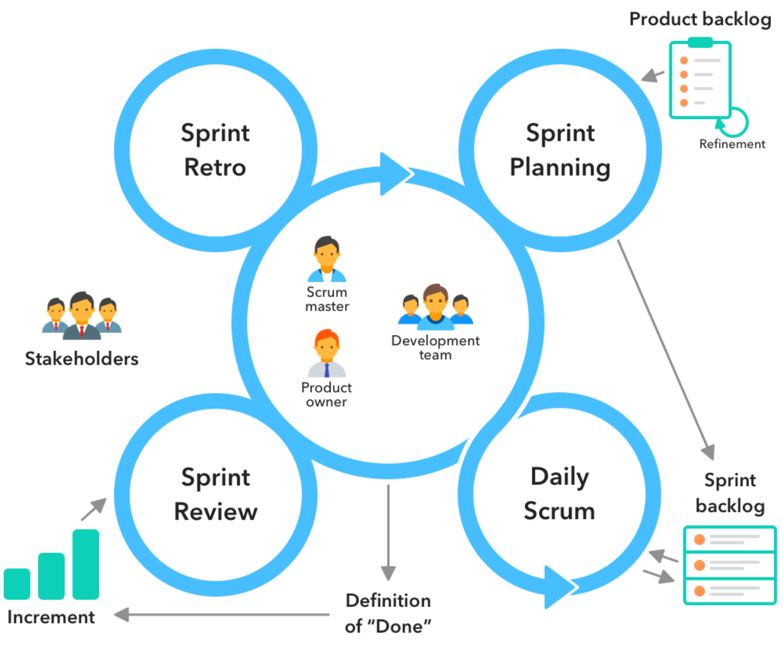


Figure 3: Scrum working methodology. Source: [15]

which user stories will be implemented during the sprint and assigns them to the members.

- **Daily Scrum:** This is a short, daily meeting where each team member shares with the rest of the team how their work is going, answering the questions "What did you do yesterday?", "What will you do today?" and "Are there any impediments in your way?". In the *DevOps* team, this meeting takes place every morning at the beginning of the day.

- **Refinement:** This meeting is not such an official ceremony necessarily, but in the *DevOps* team it takes place once a week on Wednesdays, and here is where the team gets together in two small groups and takes new tasks from the product backlog that need to be specified, so that the sprint planning is easier in the future.

- **Sprint Review:** During this meeting, which takes place at the end of every sprint, the team shares the work and tasks that have been achieved and how the goals have or have not been met.

- **Sprint Retrospective:** This is another meeting that is done at the end of a sprint, but in this case the team members get together to reflect and analyze what has been done correctly, what has gone wrong, etc., so that the work can improve in the future, and to also highlight the positive outcomes of a sprint.

Chapter 4

Project specification

4.1 Requirements specification

In this chapter I will go into more details on the design of the conceptual model of my system, including the changes that were made after the initial design and their reason. I will also specify the functional requirements mentioned in section 2.4.1 and describe them.

4.1.1 Specification of the conceptual scheme

Initial design

As it has been previously mentioned, my project is being developed based on an existing website, so the data base was not started from scratch. From the existing data base that I am using, I am only modifying one of the classes, which is the User class, by adding some attribute fields and relationships. Furthermore, I needed to create three new models: the *InfoscreenSettings*, the *PageConfiguration* and the *PageComponent*. These can be seen on the following diagram represented in figure 4.

In the diagram the relationships between the classes are also represented, as well as some restrictions regarding the minimum and maximum associated instances

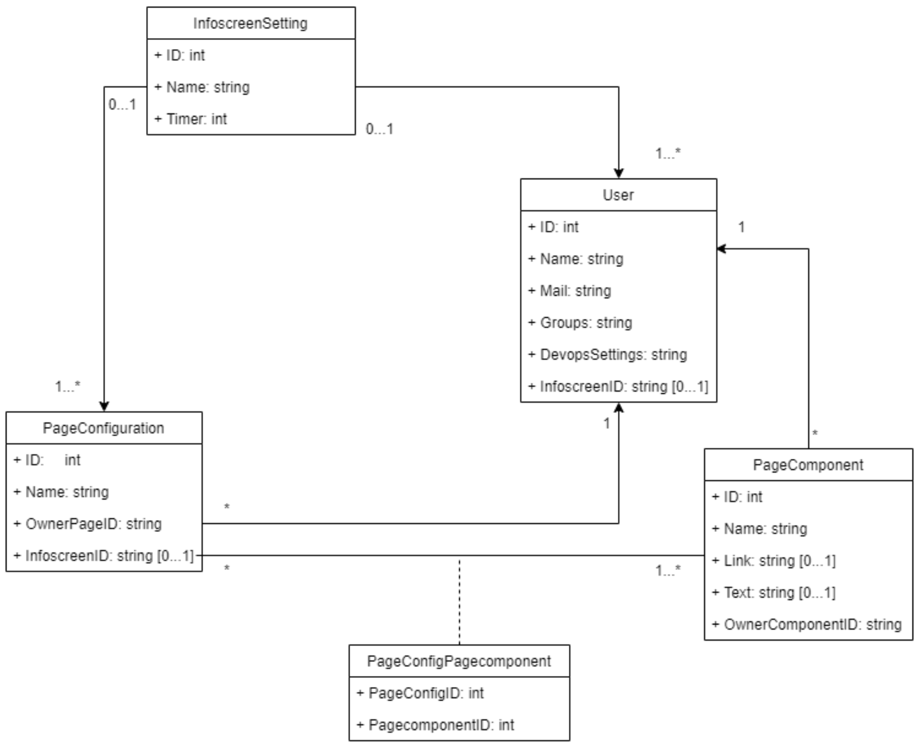


Figure 4: Initial specification of the conceptual scheme. Source: Own creation

between them. Other restrictions that could not be represented graphically are:

1. Primary keys [11] : (User, ID), (PageConfiguration, ID), (InfoscreenSetting, ID), (PageComponent, ID), (PageConfigPageComponent, PageConfigID + PageComponentID).

2. Foreign keys [12]: *InfoscreenID* in both PageConfiguration and User references InfoscreenSetting, *OwnerPageID* in PageConfiguration references User, *OwnerComponentID* in PageComponent references User, *PageConfigID* in PageConfigPageComponent references PageConfiguration, *PageComponentID* in PageConfigPageComponent references PageComponent.

3. There cannot be two instances of InfoscreenSetting with the same name.

4. There cannot be two instances of PageComponent with the same name.

5. Only a user who is has an administrator role in the system can change the configuration of their InfoscreenSetting.

6. A user can only configure the InfoscreenSetting of their own screen.

Change on the Page Component model

After some weeks during the implementation of the project, based on feedback received from other team members and several discussions, it was decided to make a small change on one of the models that I created for my project. Initially, the Page Component class was designed so that it would have all the attribute fields necessary to store information on all kinds of components (as it can be seen on figure 4). This meant that many values would be Null (for instance, a Text component does not need to store any information for the "Link" attribute, and so on). To avoid these unnecessary null values, and to make the design more scalable for future types of Page Components that might be added in the future, I redesigned this class so that it contains all the data related to the specific type of component in a *JSON* file [13]. This will allow to add different types in the future in an easy way, and will simplify the class by removing columns. The updated design can be seen in figure 5.

4.1.2 Functional requirements User Stories

In this section I will specify the User Stories [10] created for the different functionalities that were mentioned in section 2.4.1. The user stories have been put into groups known as epics, which represent the different areas in which the functionalities focus on.

The main Info screen

The user stories in this epic are all of those related to the display of the teams' info screens. These can be seen in the use case diagram in figure 6.

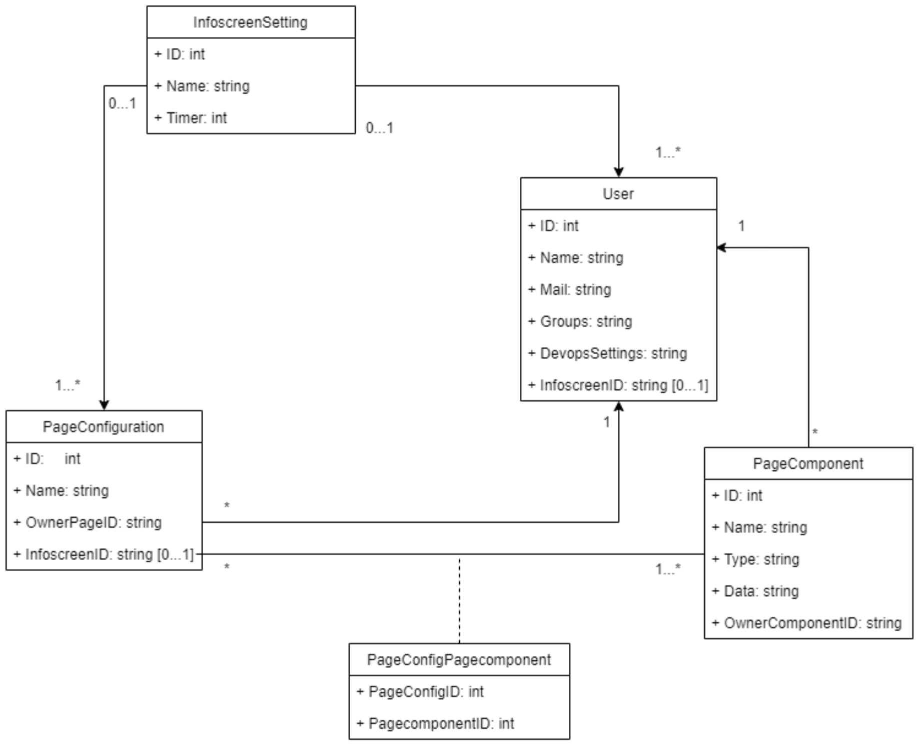


Figure 5: Updated specification of the conceptual scheme. Source: Own creation

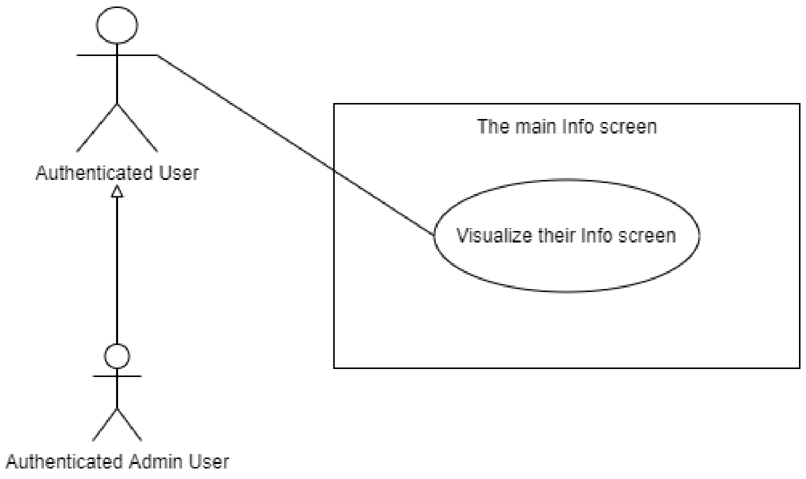


Figure 6: Use case diagram for the main Info screen epic. Source: Own creation

US1 - Display the user's info screen.

As a User, I want to be able to visualize my Info screen with its selected configuration (the pages I have chosen, with their components, and alternating between them after the specified amount of time).

Tasks:

- Load the user's Info screen settings from the data base.
- Display the first page selected on the screen.
- After the specified number of seconds (stored in the "Timer"), display the next selected page on the screen.

Acceptance criteria:

- An authenticated User can automatically visualize their own Info screen.
- The screen cycles through their selected pages after the specified number of seconds.

The Info screen's configuration

The user stories in this epic are all of those related to the settings page of the info screens. These can be seen in the use case diagram in figure 7

US2 - Display the user's info screen's settings.

As a User, I want to have access to a page where I can see my Info screen's configuration, and I want to be able to modify the selected pages and the timer. I also want to be able to click on buttons that will take me to add a new page, modify or delete an existing one, add a new page component, or save the configuration.

Tasks:

- Create a page layout for the screen's configuration.
- Display in the user's info screen configuration page its selected configuration.

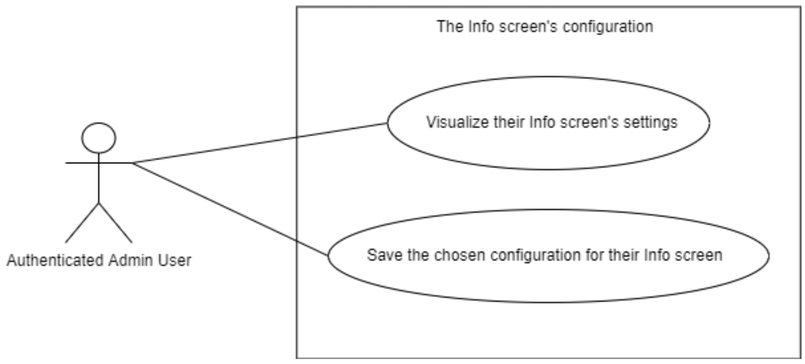

The Info screen's configuration
Visualize their Info screen's settings
Save the chosen configuration for their Info screen
Authenticated Admin User

Figure 7: Use case diagram for the Info screen's configuration epic. Source: Own creation

– Get from the data base the pages that the user has created, and display them as options.

– Add an input area for the timer (which displays the existing timer, if it has a value).

– Add buttons that will take the user to other functionalities: create a new page, create a new page component, modify a page configuration or delete a page configuration.

Acceptance criteria:

– A User can see their info screen's configuration page with all its elements.

– A User can select which pages to display on their info screen from a list with all the pages they have created.

– A User can choose a timer value to alternate between pages in their info screen.

– A User can click on the necessary buttons that will take them to add a new page, add a new page component, modify an existing page, delete a page or save the screen's configuration.

US3 - Save a user's chosen configuration for their info screen.

As a User, I want the specified configuration for my Info screen (which pages to display, and the timer) to be saved when I click the Save button on the settings page.

Tasks:

– Get the pages selected by the user and modify their InfoscreenSetting in the data base with these pages and the timer they have specified.

Acceptance criteria:

– The configuration the User selects for their Info screen is stored in the database, and therefore it is displayed when they open their Info screen on the main page.

– A User cannot save the screen's configuration if no pages have been selected for it.

– If more than one page has been selected for the screen's configuration, a User cannot save it if the timer value is null or equals 0.

– A User is notified if the saving of the screen's configuration has been successful or if there has been any error.

Page Configuration

The user stories in this epic are all of those related to the creation, modification and elimination of a page configuration. These can be seen in the use case diagram in figure 8

US4 - Create different layouts for the info screen.

As a User, I want to create pages with different layouts for my Info screen (so that each page may have a different number of components, from 1 to 6, for example).

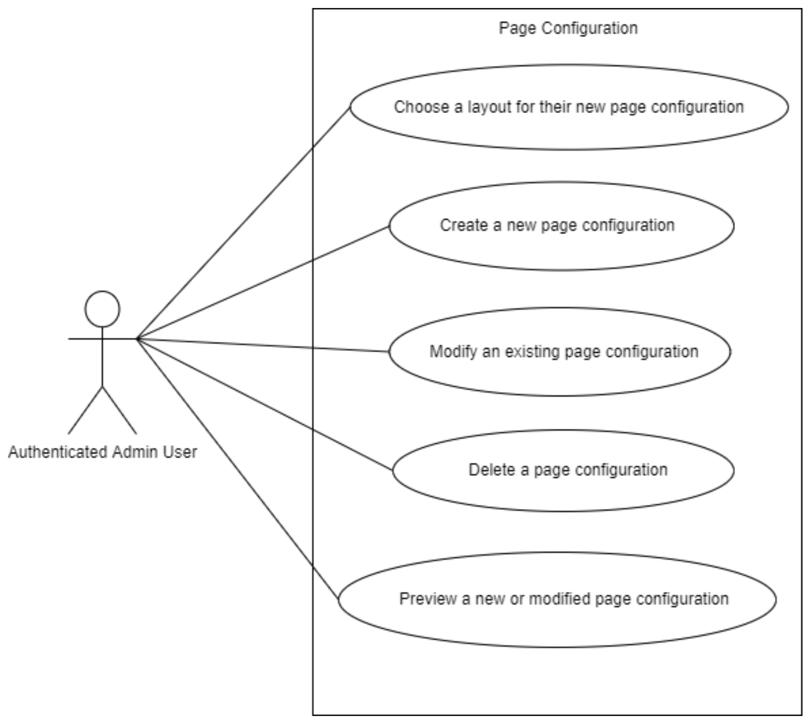

Page Configuration
Choose a layout for their new page configuration
Create a new page configuration
Modify an existing page configuration
Delete a page configuration
Preview a new or modified page configuration
Authenticated Admin User

Figure 8: Use case diagram for the Page Configuration epic. Source: Own creation

Tasks:

- Create 6 different layouts for the new page, each with a different number of components.
- Offer the user to choose which layout they want for their new page, and take them to configure it depending on which one they click on (on the "new page configuration" screen).

Acceptance criteria:

- A User can choose a specific layout for their new page.
- When the User chooses the layout for the new page, they are taken to that layout's configuration page.

US5 - Create a new page configuration.

As a User, I want to choose which components to display in each part of a new page for my Info screen. Then, I want this new page configuration to be saved when I click the Save button on the new page configuration.

Tasks:

- Load the user's page components as options to select from.
- Get the components selected by the user and create a new Page Configuration (owned by the user) in the data base with these components and the specified name for the page.

Acceptance criteria:

- A User can configure the specific layout they clicked on and choose which components should appear in each section.
- The User has given a name to the new page and all its components have a selected value.
- The configuration a User selects for their new page is stored in the database, and therefore it is displayed as an option when they try to change their Info screen's settings (as a page to select).
- A User is notified if the saving of their new page configuration has been successful or if there has been any error.

US6 - Modify an existing page configuration.

As a User, I want to be able to modify one of the pages I have previously created, so that I can change name of the page and which components are displayed in each part of the screen.

Tasks:

– Add a modify button on the pages showed in the InfoscreenSetting page that takes the user to the corresponding modify page.

– Create the modify page that loads the selected page's existing configuration (its name and the different components that were selected for it), but that allows the user to change these.

– Save the page's new changes in the data base

Acceptance criteria:

– A User can choose to modify one of the pages they have created, and they can change its name or components.

– The name page field is not empty and all its components have a selected value.

– This information will be updated on the database.

– A User is notified if the saving of the page configuration has been successful or if there has been any error.

US7 - Delete a page configuration.

As a User, I want to be able to delete one of the pages I have previously created.

Tasks:

– Add a delete button on the pages showed in the InfoscreenSetting page that opens a confirmation box.

– Create a pop-up confirmation box that asks for either "Accept" or "Cancel" when the user wants to delete a page.

– Delete the selected page from the data base after confirmation.

Acceptance criteria:

- A User can choose to delete one of the pages they have created.
- A User must confirm the delete action.
- After confirmation, the page is deleted from the data base, therefore it won't be displayed on the list of pages created by the User.

US8 - Preview a new or modified page configuration.

As a User, I want to be able to preview a page configuration after I select the components and name for it (either when I am creating a new page or when I am modifying an existing one).

Tasks:

- Add a "Preview" button on the new page configuration screen, and on the modify page.
- On the click of this button, display what the page configuration would look like with the selected page components.
- Add a close button that takes the user back to the previous screen and closes the preview.

Acceptance criteria:

- A User can choose to preview a page configuration they are creating or modifying, therefore getting a visualization of what the page will look like.
- A User can close this preview and go back to the previous screen.

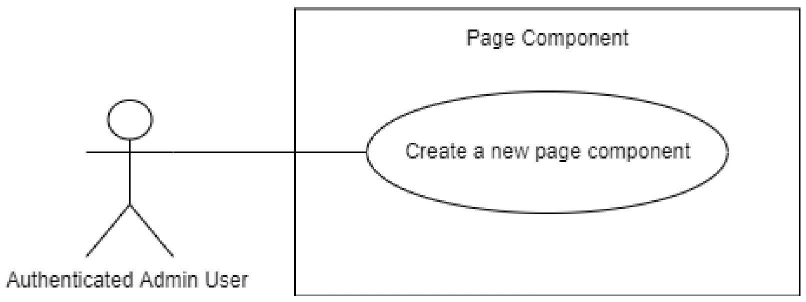


Figure 9: Use case diagram for the Page Component epic. Source: Own creation

Page Component

The user stories in this epic are all of those related to the creation, modification and elimination of a page component. These can be seen in the use case diagram in figure 9

US9 - Create a new page component.

As a User, I want to be able to add my own components (that have to be either Power BI, Grafana or a Text note). I want to do so by providing the source link for the graphs, or the text if it is a note.

Tasks:

- Create page for adding a new component, where the user must give a name for the component, choose which type of component they want to create (from the three different types) and enter the graph's source link or the text.
- Save this information as a new component on the data base (a Page Component), created by the authenticated user, when they click on the Save button.

Acceptance criteria:

- A User can add a new page component by providing its name, selecting the type and entering a source link or text.
- The type of component has been chosen, and neither the name or link/text fields are empty.
- The given input is stored in the data base, therefore, it will appear as an option when a User chooses the elements while configuring a new page/modifying an existing one.
- A User is notified if the saving of their new page component has been successful or if there has been any error.

4.2 Design and implementation

In this section of the project's specification I will go more in depth on the whole system's architecture and vision. I will first talk about the technical side related to the framework that is used, patterns and data base structure and testing, and then I will share how the users of the system are handled and how authentication is securely established. Finally, I will do an overview of what the whole project looks like with its functionalities, going through the screens and pages and explaining how it all comes together in the end.

4.2.1 System architecture

As it has been previously described, the framework that was used to develop this project is ASP.NET, which is the same they were already using on the *Viking DevOps* website, given that the info screen is implemented as a sub-project within that.

ASP.NET is a web application framework developed by Microsoft as an extension of their .NET platform, which was the successor of ASP (Active Server Pages). ASP.NET is designed to work with HTML, CSS and JavaScript, so that developers can create dynamic web pages or applications. It also allows programmers to use any .NET coding language, such as C sharp, VB or J sharp, which are usually used for the back-end code (business logic, data access...). In the case of the info screen project, C sharp is used for all the back-end part, such as for the access to the data, and for the data base itself.

More specifically within the framework, the Info screen is implemented as an ASP.NET Core project, which is a redesign of ASP.NET 4.x., using *Razor Pages*. Razor Pages is a server-side, page-focused framework that enables building dynamic, data-driven web sites with clean separation of concerns. It is a part of the ASP.NET Core framework, and it is based on the MVC pattern (Model-View-Controller), encouraging separation of logic.

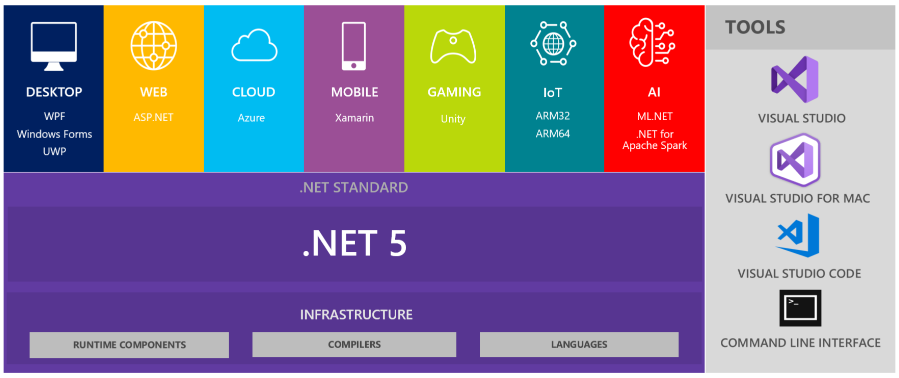


Figure 10: General view of the .NET platform . Source: [16]

In the implementation of my solution, Razor Pages are mainly used to separate sections on the User Interface logic, such as the layout, CSS and styling, and calling the JavaScript files that contain the code behind the page presentation.

Furthermore, the project follows the traditional *N-Layer* architecture, which consists of organizing the application logic into three layers: the User Interface (UI), the Business Logic (BLL) and the Data Access (DAL). This architecture is used to separate the user interface, the data and the business logic. All the user requests are handled in the UI layer, which interacts only with the BLL. This one, in turn, calls the DAL to access data.

The benefit of using layers is that it helps scale the application so that it is easier to code, debug and test some specific functions of the project.

The **UI** layer, meant to treat the user requests and then communicate with the BLL, is handled via HTML, CSS, and JavaScript pages in my project. Then, the **BLL**, represented by the Infoscreen Controller, receives this information and communicates with the **DAL** by making a request to the data base.

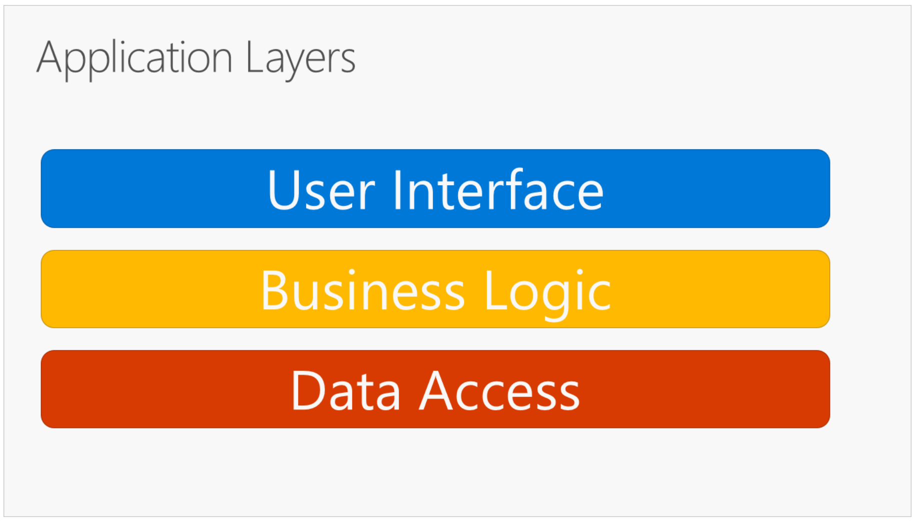


Figure 11: Typical application layers. Source: [17]

4.2.2 The data base

The database that I use to store the necessary information for the Info screen project is a relational database based in Microsoft SQL Server, which is a service used to manage databases, add tables, relationships, etc. This data base was already up and running when I joined the team, so it was not a part of my project to decide on any architecture issues for it, but rather to add my new classes and learn how to use it. I will give a general view of its structure next and explain how it is connected to the info screen's implementation.

In order to add my new models to the existing database, I had to modify the *Viking Devops DB* project, which is an ASP.NET project that uses *Entity Framework Core* to interact with the database.

Entity Framework is an open-source Object-Relational Mapping framework (converts data between type systems using object-oriented programming languages) for Microsoft .NET applications. EF allows developers to work at a higher level of abstraction when interacting with data and, as seen in figure 12, it fits between the database and the data layer that accesses it.

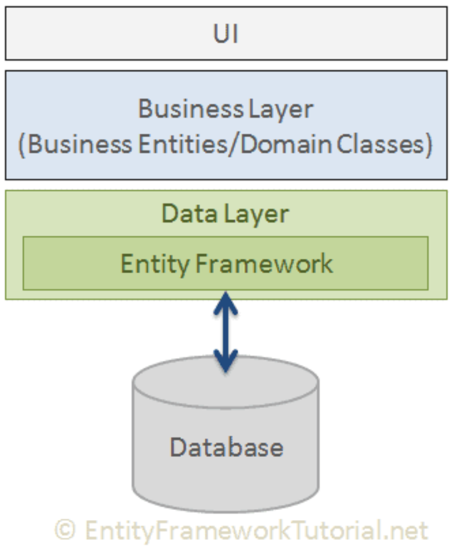
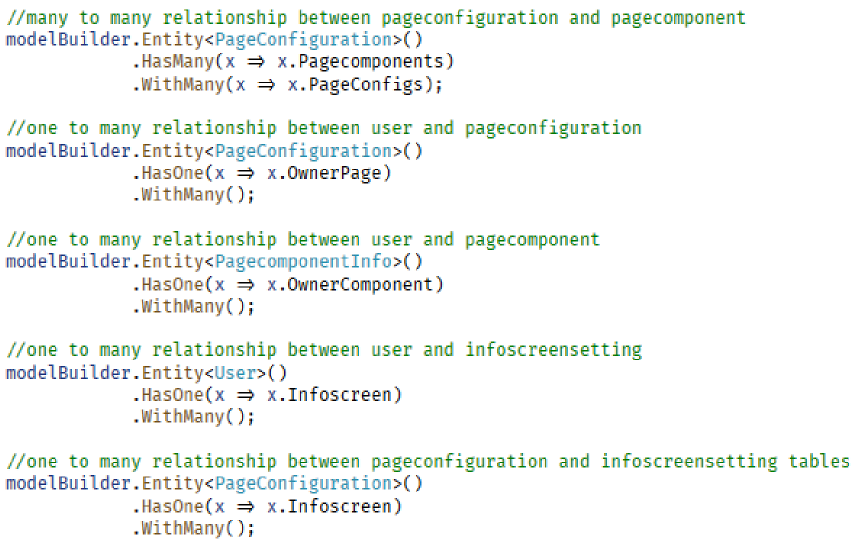

UI
Business Layer
(Business Entities/Domain Classes)
Data Layer
Entity Framework
Database
© EntityFrameworkTutorial.net

Figure 12: Entity Framework in a project's layers. Source: [18]

Entity Framework then allows me to connect with the Viking DevOps database through the *DevopsDBContext*. A context represents a session with the database which can be used to query and save instances of entities to a database, in this case to the DevopsDB database. This context provides models and helper methods for the database used to store the Info screen classes and their relationships.



```
//many to many relationship between pageconfiguration and pagecomponent
modelBuilder.Entity<PageConfiguration>()
            .HasMany(x ⇒ x.Pagecomponents)
            .WithMany(x ⇒ x.PageConfigs);

//one to many relationship between user and pageconfiguration
modelBuilder.Entity<PageConfiguration>()
            .HasOne(x ⇒ x.OwnerPage)
            .WithMany();

//one to many relationship between user and pagecomponent
modelBuilder.Entity<PagecomponentInfo>()
            .HasOne(x ⇒ x.OwnerComponent)
            .WithMany();

//one to many relationship between user and infoscreensetting
modelBuilder.Entity<User>()
            .HasOne(x ⇒ x.Infoscreen)
            .WithMany();

//one to many relationship between pageconfiguration and infoscreensetting tables
modelBuilder.Entity<PageConfiguration>()
            .HasOne(x ⇒ x.Infoscreen)
            .WithMany();
```

Figure 13: Example of code that uses Entity Framework to establish the relationships between classes. Source: Own creation

The database is then linked to the ASP.NET project where the info screen is implemented as a ***NuGet*** package. A NuGet package is a single ZIP file with the .nupkg extension that contains compiled code (DLLs), other files related to that code, and a descriptive manifest that includes information like the package's version number [19]. So the previously mentioned code that uses Entity Framework is the part I modify in order to update the data base with my new models. My code is merged into the master branch which is then built into a C sharp solution and the NuGet package. Once this is done, the package is published through a pipeline to the team's package manager solution, and this is updated on the main ASP.NET project's settings.

Figure 14 represents how this process is done in a more visual way: how the initial project (the code that uses Entity Framework) is built into a nuGet package, that is then published (in this case, privately), so that it can be installed in another .NET Project (where my Info screen is implemented).

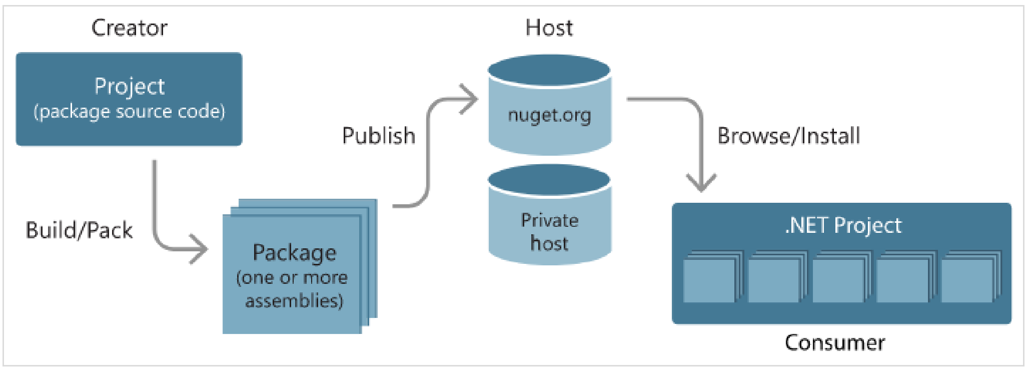
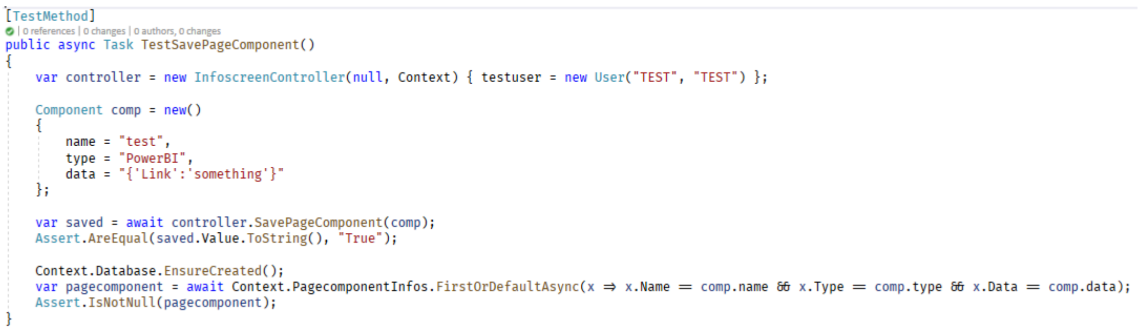


Figure 14: Building a NuGet package. Source: [19]

4.2.3 Testing

The main testing done for some parts of the info screen has been based on Unit Testing. This is a software development process that consists of testing the smallest testable parts of an application (the units), ensuring software quality at the most basic level. The testing has been done around the Business Logic Layer, that in my case means on the *Infoscreen Controller*, which accesses the data, and also on the Data Access Layer itself, so on the database project that uses Entity Framework. These tests are all implemented on new projects dedicated explicitly for Unit Tests, that are within the bigger ASP.NET projects where the web application and the data base are held.

In the database project, individual tests are made for each model that I added to store data about the Info screen (the Page Configuration model, the Page Component model, and the Infoscreen Setting). In the case of the Infoscreen Controller, which is a class in the main ASP.NET project where the solution is implemented, the unit tests are made for the methods that access the database layer, specially the more complex ones, which update or modify the database. Some examples of these cases are the methods that save a page configuration, update or delete it, the ones that save a page component, or those that update the info screen configuration.



```
[TestMethod]
| 0 references | 0 changes | 0 authors, 0 changes
public async Task TestSavePageComponent()
{
    var controller = new InfoscreenController(null, Context) { testuser = new User("TEST", "TEST") };

    Component comp = new()
    {
        name = "test",
        type = "PowerBI",
        data = "{'Link':'something'}"
    };

    var saved = await controller.SavePageComponent(comp);
    Assert.AreEqual(saved.Value.ToString(), "True");

    Context.Database.EnsureCreated();
    var pagecomponent = await Context.PagecomponentInfos.FirstOrDefaultAsync(x => x.Name == comp.name && x.Type == comp.type && x.Data == comp.data);
    Assert.IsNotNull(pagecomponent);
}
```

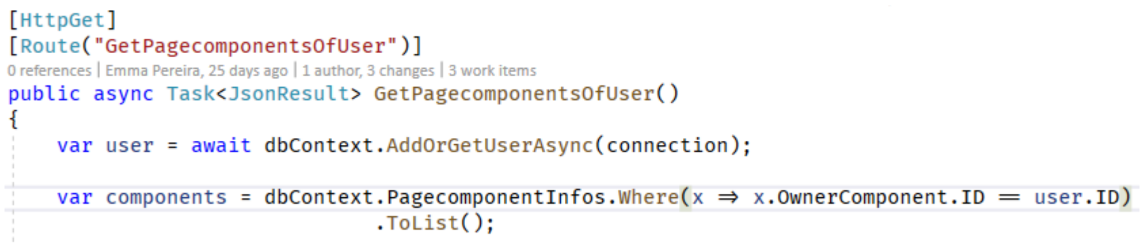
Figure 15: Example of a unit test for the method that saves a Page Component in the database. Source: Own code

4.2.4 Users and authentication

Users and their accounts are handled directly by the organization (by Danfoss) through Azure DevOps. All information is managed by Azure through the App Registration, so the project's database does not store any kind of sensitive or private information on them.

When I need to get a user's connection to access some of their information (such as the email, permissions...) I do this by using existing methods that get the user's connection. Because the team has certain permissions granted by the IT department for the app registration to access things, I am able to read user's non-sensitive profile data, which is then used to create relationships between users and the different info screen components.

An example of this can be seen below on figure 16, this is a method through which I establish a connection to the user's profile by using an existing method that was already implemented for other parts of the website. This way I get access to the user and their relationships to the info screen classes, such as their ID.



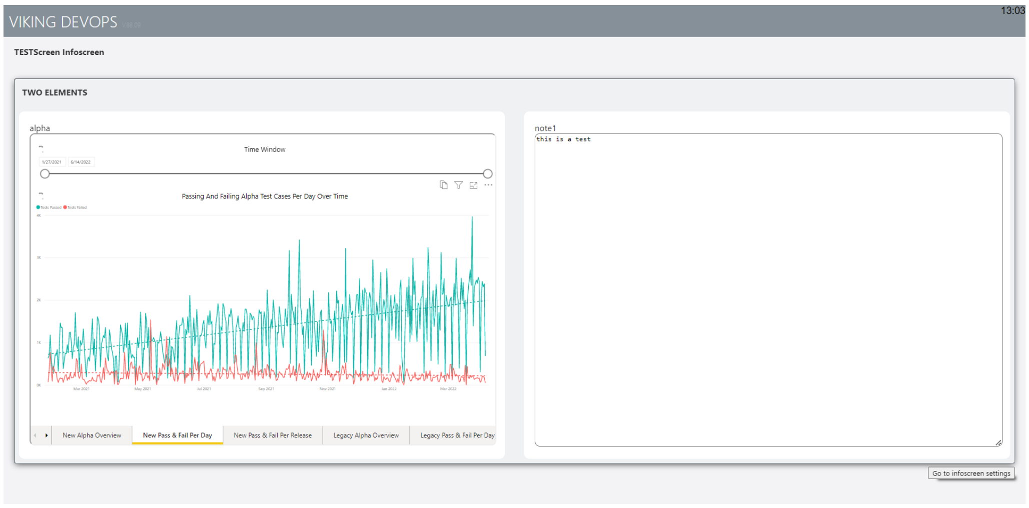
```
[HttpGet]
[Route("GetPagecomponentsOfUser")]
0 references | Emma Pereira, 25 days ago | 1 author, 3 changes | 3 work items
public async Task<JsonResult> GetPagecomponentsOfUser()
{
    var user = await dbContext.AddOrGetUserAsync(connection);

    var components = dbContext.PagecomponentInfos.Where(x => x.OwnerComponent.ID == user.ID)
                                    .ToList();
```

Figure 16: Method that accesses the user's information. Source: Own code

4.2.5 General vision of the whole system

In this following section I will share what the project looked like in the latest weeks, how the screens interact between each other and how everything came together, creating what will later be the first version of the Info screen. The following images are all from a testing stage, with non-relevant information and pages that I created just to give an example of what the Info screen could look like.

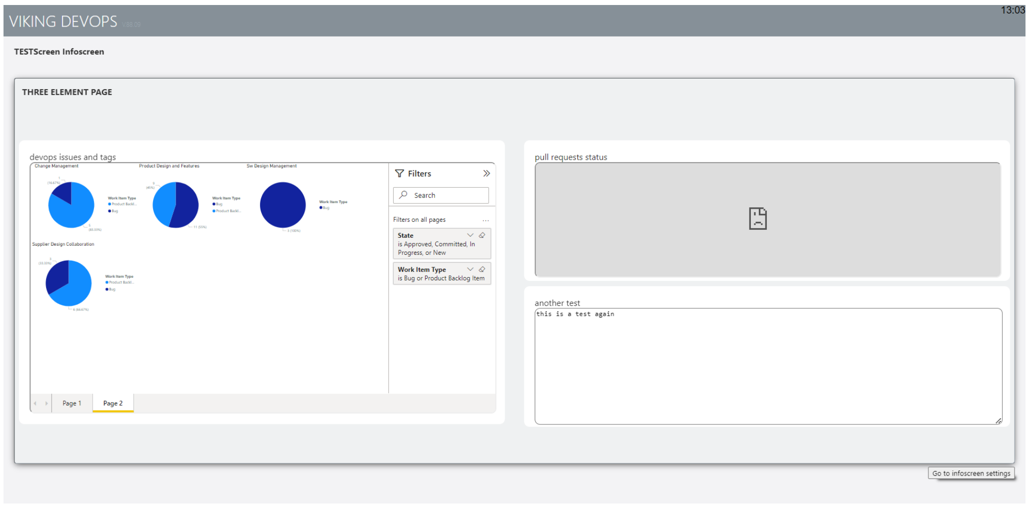
The first page that loads when the application is running is the main info screen. An example of this can be seen in figure 17, where the website gets the user's connection and loads their (in this case, my) Infoscreen. The screen automatically alternates between the pages I have selected for it, after the specified amount of seconds. In



```
this is a test
```

Figure 17: A test of the main info screen, first page. Source: Own creation

this case, the screen loads what is seen in figure 17 first, and after 10 seconds it changes into the page seen in figure 18.

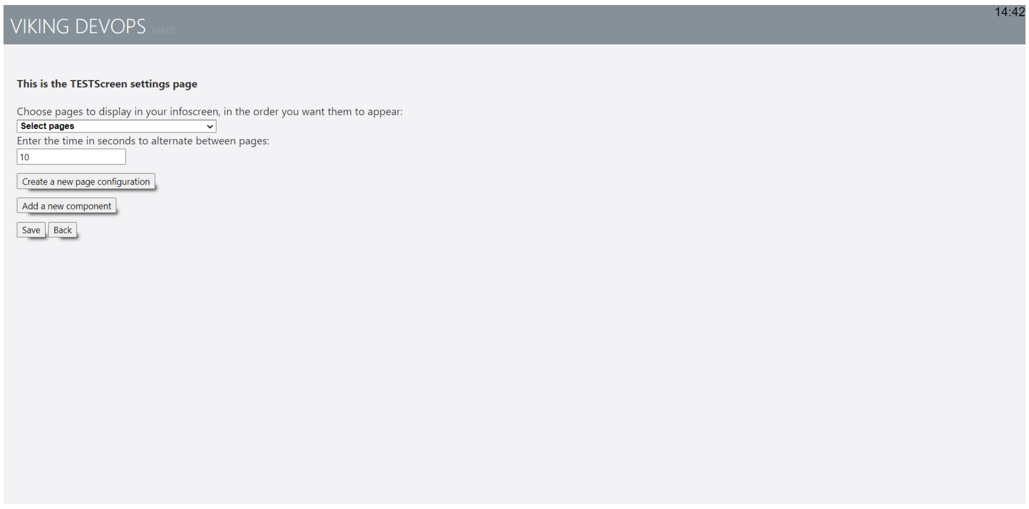


```
this is a test again
```

Figure 18: A test of the main info screen, second page. Source: Own creation

From here, we can access the Settings page of the Info screen.

This page automatically loads and displays several things, as it is seen in figure 19:


VIKING DEVOPS V.88.09

14:42

This is the TESTScreen settings page

Choose pages to display in your infoscreen, in the order you want them to appear:

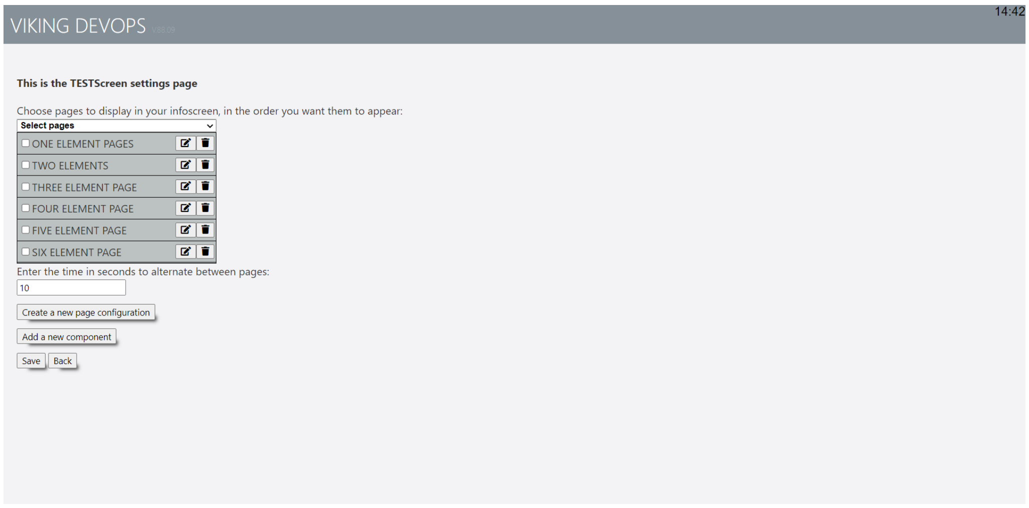
Select pages

Enter the time in seconds to alternate between pages:

10

Figure 19: The Infoscreen settings page. Source: Own creation

the name of the screen, a select object with the page configurations the authenticated user has already created, and the timer for the screen (if there is one already).


VIKING DEVOPS V.88.09

14:42

This is the TESTScreen settings page

Choose pages to display in your infoscreen, in the order you want them to appear:

Select pages

- ☐ ONE ELEMENT PAGES
- ☐ TWO ELEMENTS
- ☐ THREE ELEMENT PAGE
- ☐ FOUR ELEMENT PAGE
- ☐ FIVE ELEMENT PAGE
- ☐ SIX ELEMENT PAGE

Enter the time in seconds to alternate between pages:

10

Figure 20: The Infoscreen settings page with the user's pages. Source: Own creation

And it also allows the user to do several actions: they can change their info screen's configuration by selecting pages for it and a timer, and saving that, they can modify or delete any of their pages (through the icons seen next to the names of the pages in figure 20), or they can decide to go create a new page configuration or add a new page component.

Now we can go inside each of these other actions the user can proceed with.

Firstly, the modify page (figure 21). This page loads the selected page configuration's fields (its name and page components) and allows user input so that they can change any of these. There is also a "Preview" button that, when clicked, displays the selected page components in their corresponding space. The preview can be closed with the same button. And finally, the user can save this, modifying the page configuration on the data base.

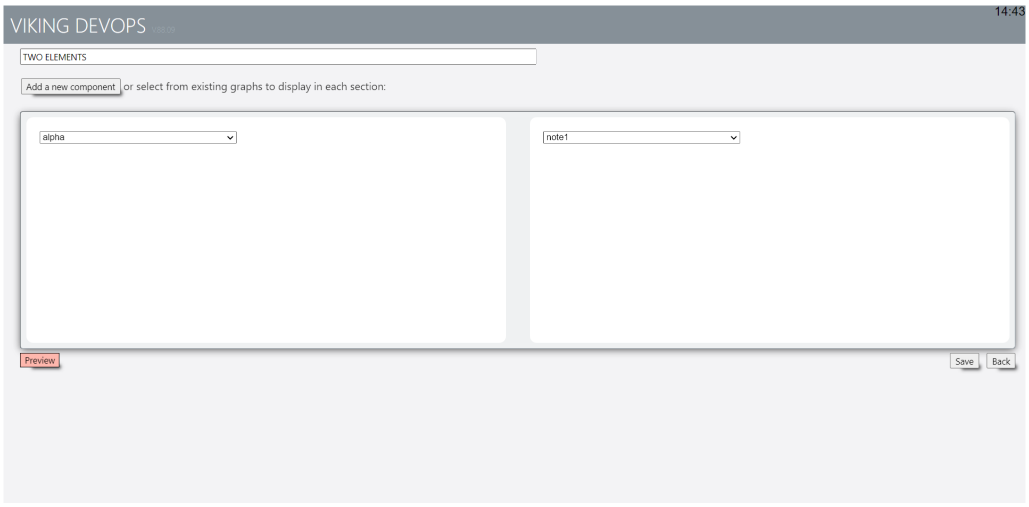
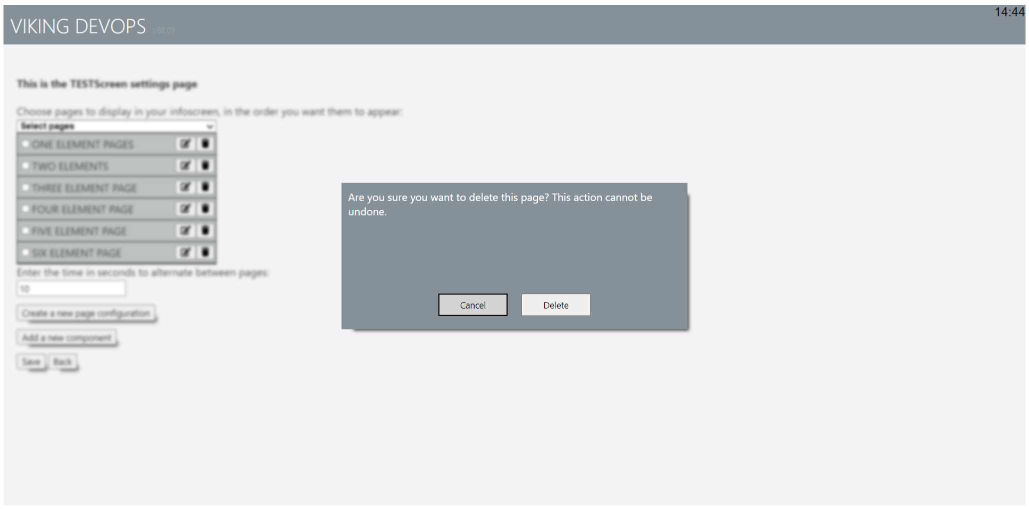


Figure 21: Example of a page configuration modify page. Source: Own creation

Then there is the "delete" functionality (figure 22), as seen below.

When the user clicks on the delete button next to a page's name a pop up box appears, asking for further confirmation to delete this page. If the user clicks on "Cancel", then the box disappears and nothing happens, but if they click on "Delete", then the page configuration is deleted from the data base (as well as its relationships) and the page is refreshed.

From the Infoscreen Settings page users can also access two other functionalities: creating a new page, or adding a new component.


VIKING DEVOPS

14:44

This is the TESTScreen settings page

Choose pages to display in your infoscreen, in the order you want them to appear:

Select pages

- ONE ELEMENT PAGES
- TWO ELEMENTS
- THREE ELEMENT PAGE
- FOUR ELEMENT PAGE
- FIVE ELEMENT PAGE
- SIX ELEMENT PAGE

Enter the time in seconds to alternate between pages:

10

Create a new page configuration

Add a new component

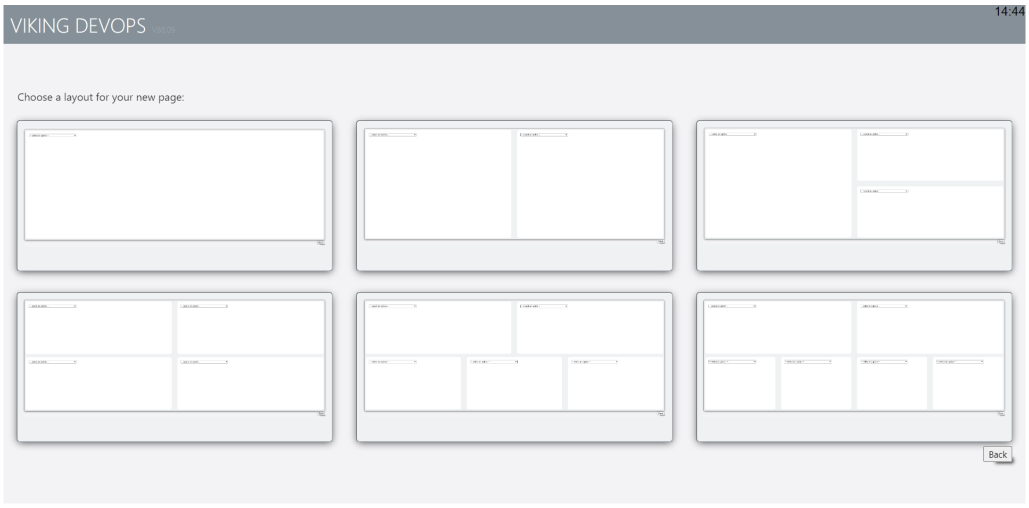
Save Back

Are you sure you want to delete this page? This action cannot be undone.

Cancel Delete

Figure 22: Example of the delete functionality. Source: Own creation

If the user chooses to create a new page, they are taken to a screen where they can first choose which layout they want to use for it, as seen above in figure 23. Based on their choice, they will be taken to the corresponding page.


VIKING DEVOPS

14:44

Choose a layout for your new page:

Back

Figure 23: Screen where the user can choose a layout for their new page. Source: Own creation

In this next screen, that looks like the one in figure 24, the user must give a name to the page and choose a page component for each part of the screen.

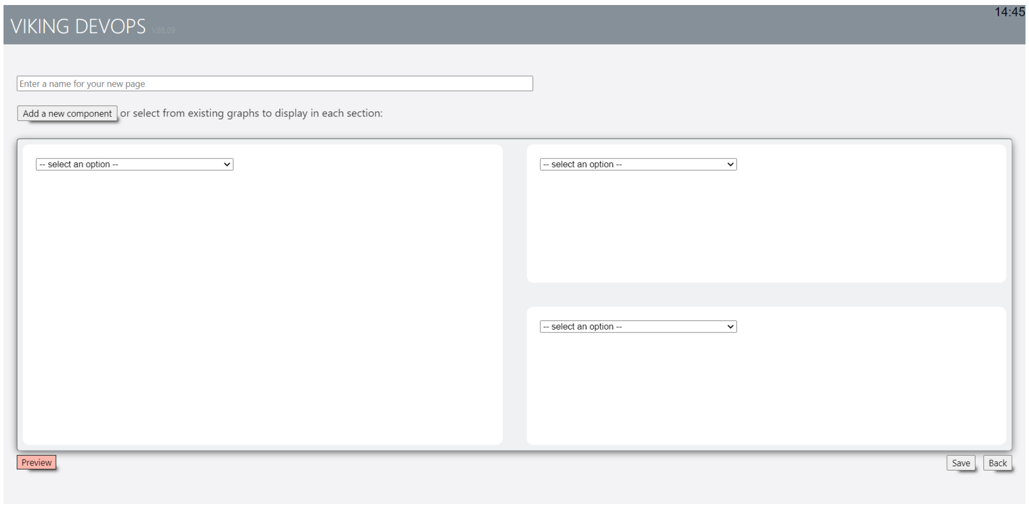


Figure 24: Screen where the user can create a new page. Source: Own creation

The page components that appear as options are those the user has already created, and they are grouped by type (Power BI graph, Grafana dashboard, or a text note), as displayed in figure 25. Before (or after) saving the new page, the user can also preview it, in the same way that it was possible in the Modify page, as it is seen in figure 26.

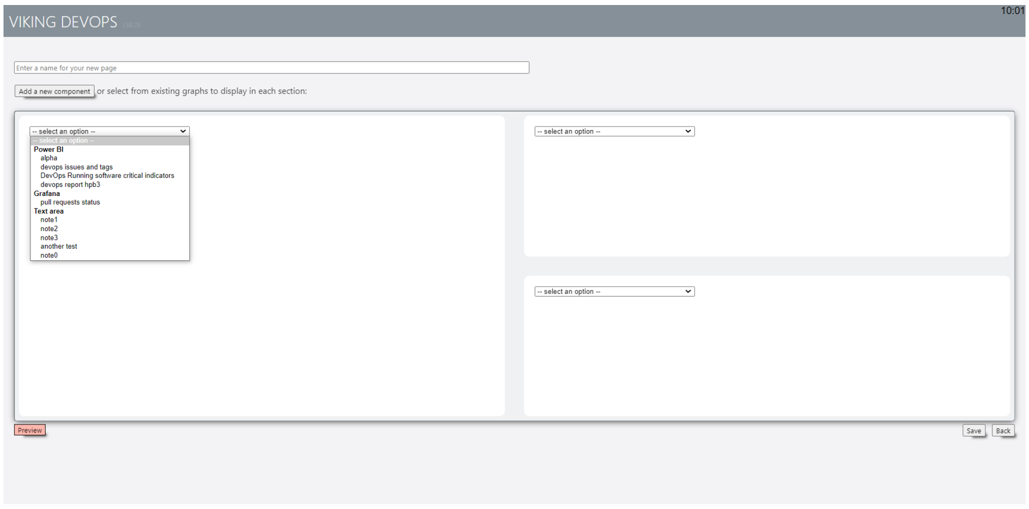
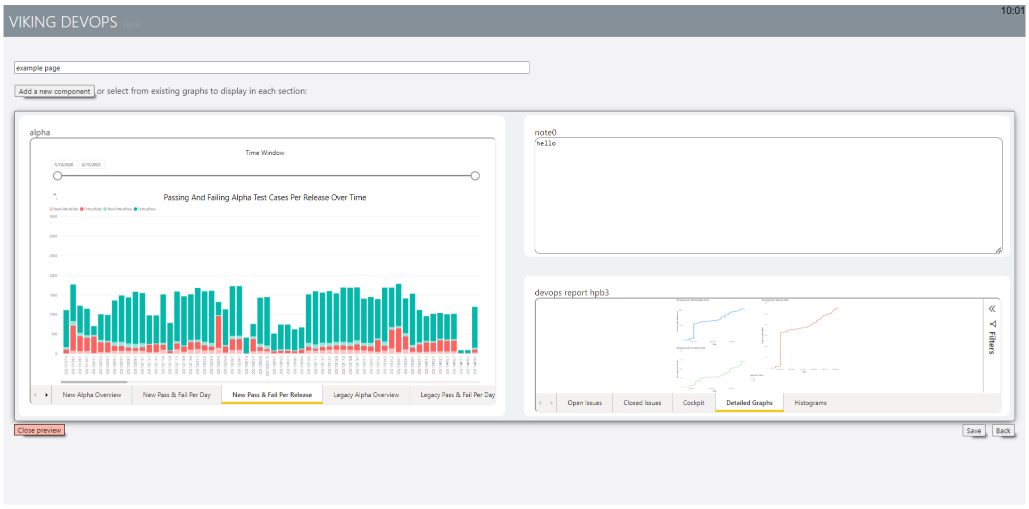


Figure 25: The page components offered are those the user has created. Source: Own creation



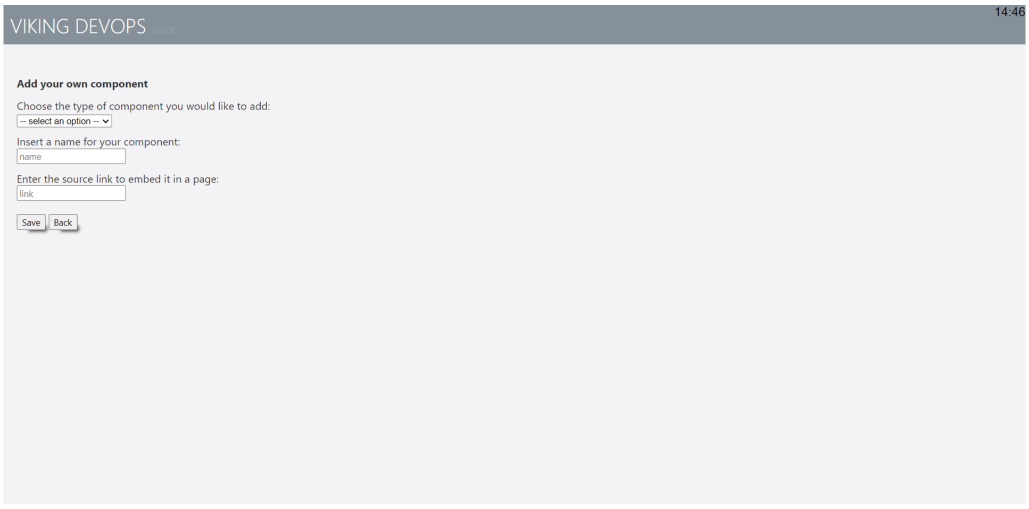
example page

or select from existing graphs to display in each section:

hello

Figure 26: Example of the preview of a new page configuration. Source: Own creation

Finally, if the user chose to add a new component, they will be taken to a screen like the one in figure 27, where there is a small form where they should enter the necessary input to create a new page component: select the type (like the ones offered in figure 28), give the component a name, and add a link or text depending on the type of component.



Add your own component

Choose the type of component you would like to add:

Insert a name for your component:

Enter the source link to embed it in a page:

Figure 27: Screen where the user can add a new component. Source: Own creation

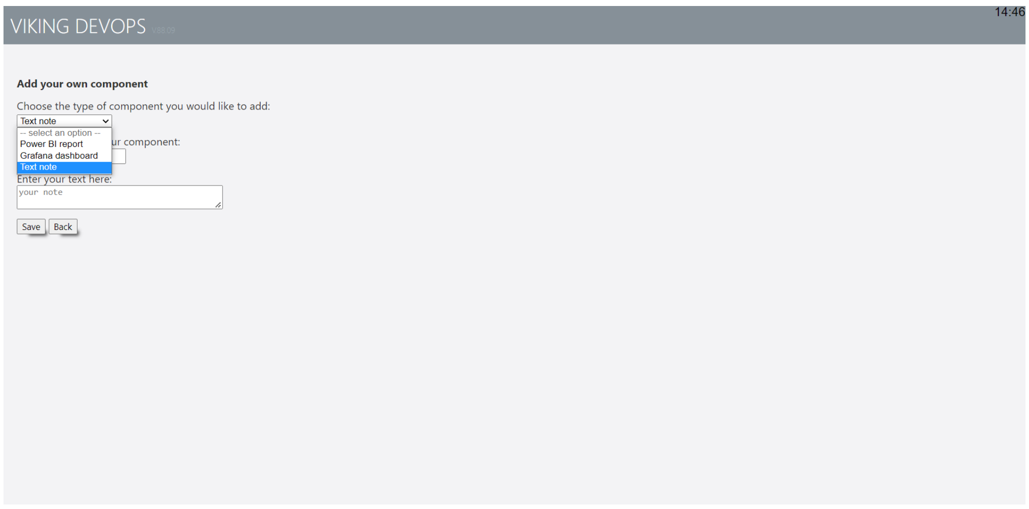


Figure 28: Screen where the user can add a new component, three types. Source: Own creation

Chapter 5

Time planning

5.1 Introduction

In this chapter of the document I will specify all the tasks that will be carried out throughout the project and the dedication on each of them. The implementation of this solution takes place between the middle of February and the end of June, even though my internship at Danfoss is longer than that. So I am planning the tasks in the project throughout the time span of 19 weeks. And with an average of 5 daily hours dedicated to it, the final amount of time for the thesis is of 475 hours, which is broken down into details in the different sections of this chapter.

5.2 Changes to the initial planning

Part of the planning of this project has been slightly affected, specially the last two weeks of it, due to the incorporation of a new member into the team. In the middle of the month of May, an intern (called a *lærling* in the danish system) joined the *DevOps* team to work with me on my project. She is an intern at Danfoss that rotates among teams every few months and, since she has a background on the same frameworks and technicalities the monitoring solution is developed with (C sharp programming, ASP.NET framework), she was assigned to join me on the implementation of the info screen. She will also be in the team for longer, and will be the person that takes over my project when I finish my internship.

Because of this, I have made some changes on the project implementation tasks, since some of the existing tasks will be shared with her, giving space to add new functionalities to the project. These will be added into the corresponding section below.

Furthermore, and unrelated to the new member incorporation, there have been changes on two more functionalities (IC11 and a new one IC14), which will also be explained and justified in section 5.3.3. Finally, the time estimated for tasks IC12 and IC13 has been reduced because, after finishing tasks IC6, IC7 and IC8, I realized that these two functionalities (saving a new page versus modifying an existing one) have a lot in common, therefore the complexity of the latter is reduced after achieving the first one.

5.3 Description of tasks

The tasks specified for this project have been divided into two groups: the ones corresponding to the project management and documentation, and those that are related to the actual implementation. These tasks are all mentioned in chapter 4.1, as part of the user stories described there.

5.3.1 Project management

- **PM1 - Documentation of the context and scope of the project (22h):** Documenting the contextualization, justification, scope and methodology of the project. No dependencies.
- **PM2 - Documentation of the planning (11h):** documenting the project's temporal planning and its tasks. This depends on PG1.
- **PM3 - Documentation of the budget and sustainability (12h):** Documenting and researching the project's needed budget and sustainability practices. This depends on PG2.

- **PM4 - Documentation of the final document (7,5h):** Putting together the three previous sections into one full document, with the necessary changes. This depends on PG3.
- **PM5 - Documentation of the official thesis (90h):** Preparation of the official document that I will have to hand-in and discuss with the director of my thesis at UPC for the follow-up meeting first, and in the end of June the full version. No dependencies.
- **PM6 - Preparation for the presentation (25h):** Preparation for the oral exposition of my thesis. This depends on PG5
- **PM7 - Sprint Planning (2 h - 5 times):** Meeting that takes place every 14 days, at the beginning of each sprint, to plan it and assign the user stories. No dependencies.
- **PM8 - Daily Scrum (30 min - 75 times):** Meeting that takes place every day in the morning, to share with the other members of the team what everyone has been working on. No dependencies.
- **PM9 - Refinement (1 h - 5 times):** Meeting that takes place once a week, on Wednesdays, to refine and specify new tasks in the product backlog. No dependencies.
- **PM10 - Sprint Review (1 h - 5 times):** Meeting that takes place every 14 days, at the end of each sprint, to discuss how the work has been and what has been achieved. No dependencies.
- **PM11 - Sprint Retrospective (1 h 30 min - 5 times):** Meeting that takes place every 14 days, at the end of each sprint, to reflect on the sprint and the work done. No dependencies.

5.3.2 Project implementation

As it has been mentioned in previous sections, this info screen that I am developing is meant to be a scalable product designed so that it is possible to expand and add functionalities to it in the future. This being said, with my supervisor's help, I have defined the "first version of the info screen" that will be developed for my thesis. Since my internship lasts longer than the actual thesis at UPC, it will be possible for me to further develop this project afterwards.

So this initial version that is conducted for this thesis consists of two big parts.

On the one hand, there is the implementation of the screen that displays the user's info screen by automatic authentication, which is also their team's screen. An info screen can be made of one or many pages, that alternate in a cycle based on the configuration that has been set for it. And each page allows from one to six components, so there are six different layouts possible. The components that form these pages can be of three different types for now: a *Power BI* embedded graph, a *Grafana* embedded dashboard or a text field with specific information that the user has specified.

On the other hand, we have the implementation of what would be the info screen's configuration. This will consist of several screens where the user can: choose which pages to display in their team's screen (and how many seconds they want for them to cycle through), create a new page with a selected layout and page components, add a new component to display in any of these pages, and modify an existing page.

But before jumping straight into the specific tasks to implement those two parts, I have defined **two initial tasks** that will be implemented in order to start the project:

- **IN1 - Plan the solution and design the new data base models necessary (24h):** General planning on the structure of the project, starting by the fact that the info screen needs a data base to store all its data. The data base used in this case is an existing one that is being used to store data for the Viking DevOps website. Based on this data base, I will make an initial design for the new models that I need in order to support my project, and all the relationships between these. No dependencies.

- **IN2 - Create the new models on the data base and their necessary dependencies and restrictions (17h):** Once the design of the new models needed for the info screen has been done, I need to implement this in the actual data base, which is also implemented in the AP.NET framework. This depends on IN1.

Once that has been achieved, and I have a basic structure which I can start working from, the following tasks will be implemented for each specific functionality of the info screen.

The main Info screen:

- **IF1 - Get from the data base the user's info screen configuration (19h):** Authenticate the user's connection, access the data base and look for the info screen configuration that belongs to the connected user. This will return its name, the selected pages to display, and the time in seconds to cycle through the pages. This depends on IN2.
- **IF2 - Get from the data base, for each page that needs to be displayed on the screen, its name and page components (7h):** Access the data base and look for each page's configuration. This will return their names, and the page components that form them each. This depends on IF1.
- **IF3 - Display the whole info screen configuration (38,5h):** From the data obtained, display the screen's name, and each page with its name and components accordingly, setting a timer of the specified number of seconds to alternate between each page. This task also includes creating the page layout and style of the info screen. This depends on IF2.

The Info screen's configuration:

- **IC1 - Create a page layout for the screen's configuration (8h):** Make a new layout where the user's info screen configuration will be. This must include a select option for the pages, an input space for the timer, a button that allows the user to create a new page and a button that allows them to modify an existing page. No dependencies.

- **IC2 - Display in the user's info screen configuration page its selected configuration (4,5h):** Display the screen's name on the top of the page, and the timer (if there is one already) in the input field for this. This depends on IC1.

- **IC3 - Get from the data base the pages that the user has created, and display them as options (9h):** Access the data base to obtain all pages that were created by the authenticated user, and display these as options in the select with check boxes. This depends on IC1.

- **IC4 - Save a user's input for their screen's configuration (14h):** Once the user has selected a configuration (selected one or more pages, and set a timer), the user's info screen configuration needs to be updated on the data base with the new input. This depends on IC3.

Page configuration:

- **IC5 - Create a page where the user can choose a layout for their new page (2,5h):** Make a new page where the user is offered to choose from six different layouts for the new page they want to create. Each option should take the user to the appropriate template for designing their page. No dependencies.

- **IC6 - For each layout option, create a page where the user can design their new page (10,5h):** Make the necessary pages where the user, depending on which layout they chose for their new page, can select which components to display in each field of the screen. There should also be a button that allows the user to create a new page component. No dependencies.

- **IC7 - Get from the data base the user's page components, and display them as options to select from for their new page (11h):** Access the data base to obtain all components that were created by the authenticated user, and display these as options in each select. This depends on IC6.

- **IC8 - Save a user's selection for their new page configuration (12,5h):** Once the user has selected the components for their page and given it a name, this new page configuration needs to be saved into the data base with all the selected components. This depends on IC7.

- **IC11 - Create a page where the user can choose an existing page to modify (3,75h):** Make a new page where the user is offered to choose from their existing pages so that they can modify them. No dependencies.

- **IC12 - Create a page where the user can modify an existing page (10h):** Make a new page where the page the user selected to modify is displayed, and each page component can be changed into a different one from a drop down list that shows the user's components as options. No dependencies.

- **IC13 - Save a user's changes on the page they were modifying (10h):** Once the user has done the necessary changes to a page, update this page's configuration in the data base with the new selection. This depends on IC12.

Page component:

- **IC9 - Create a page where the user can enter the necessary information to create a new page component (6h):** Make a new layout where the user's can enter the information for a new components, so there has to be an input for the type of component, the name, and the link (in the case of an embedded graph) or a text (in the case of a text component). No dependencies.

- **IC10 - Save a user's input for their new page component (7,75h):** Once the user has inserted the information for the new component (type, name and link or text), this new page components needs to be saved into the data base with all the input fields. This depends on IC9.

5.3.3 Changes and new tasks

- **IC11 (3,75h):** Instead of creating a page where the user can choose an existing page to modify from the existing ones, this task will be simplified by adding a small modify button on the pages that are displayed in the settings of the info screen. This button redirects the user to the modify page. This change has been made so that it is more intuitive for the user to modify a page they see, and to avoid unnecessary extra pages and make the interface more user friendly. This depends on IC3. This task will be **implemented by the new intern**, so it will not be taken into consideration on my task planning, Gantt chart, etc.

- **IC14 - Redesign of the Page Component model on the data base (8h):** As it was mentioned in section 4.1.1 of the Requirements specification chapter, the Page Component class is redesigned so that we can avoid null values and to make the project scalable for future types of components. This task includes the changes on the data base itself and also on updating some methods so that they support the new attributes of the Page Component. No dependencies.

- **IC15 - Add delete button that deletes a page after the confirmation from the user:** Add a button that allows the user to delete a Page Configuration. This button should open a pop-up box asking for confirmation of the user. This depends on IC3. This task will be **implemented by the new intern**, so it will not be taken into consideration on my task planning, Gantt chart, etc.

- **IC16 - Delete a Page Configuration from the data base (5h):** When the user confirms that they want to delete a page, this one is removed from the data base. This depends on IC15.
- **IC17 - Add a preview page when the user is creating a new page configuration (8,25h):** When the user has chosen the name and components for a new page, display a preview of this on the click of a button, so that the user has the option to visualize it before saving it. No dependencies.
- **IC18 - Connect the pages of the whole project by adding more buttons to move among the website:** Make the whole project more user friendly and easy to use by connecting the different pages with "Back" buttons, among others. This task will be **implemented by the new intern**, so it will not be taken into consideration on my task planning, Gantt chart, etc.

5.3.4 Resources

Human resources

The main human resource of this project will be myself (**EP**), since I will be implementing the project and writing the thesis. But there are also a few other important human resources from two different groups.

On the one hand, in the company, Danfoss, I have a supervisor (**APH**) who guides my project and its scope, and there is also a "second supervisor" (**SM**), who is the Scrum master of the *DevOps* team, that is also helping me organize and plan my tasks. Finally, two other developers in my team (Philipp Bang Hellesoe and Jacob Norrelykke Iversen) are the ones I will be in most contact with when it comes to the implementation of the monitoring solution, since they have knowledge on the tools I will be using for it, but in general I will count with the whole team's support (**DT**) throughout the project.

On the other hand, in UPC, my tutor Ernest Teniente (**ET**) will be supervising my thesis and making sure that I am developing it correctly. And there will also be GEP professors (**GEP**) who will give me feedback on the deliverables for this assignment.

Material resources

In terms of hardware, the essential resource will be a laptop. When I am implementing the project I will be using the one that Danfoss provides for me, which is a *Lenovo ThinkPad T15.* When I am working outside of the office, I will be using my personal laptop (*MacBook Pro* 2017).

In terms of software, the following resources will be needed:

- **Microsoft Visual Studio (VS):** integrated development environment (IDE) used to implement the monitoring solution.
- **Azure DevOps (AD):** Microsoft service used on a daily basis for the project's management, as well as for uploading my project's code and version control.
- **Microsoft Teams (MT):** communication platform used for meetings among the company and team.
- **DB Browser for SQLite (DB):** a visual tool I will use to create, design, and edit database files I will need to develop my info screen project.
- **Power BI (PBI):** software service from which I will obtain data that will be displayed in the info screen.
- **Grafana (GR):** another software service from which I will obtain data that will be displayed in the info screen.
- **Overleaf (O):** online editor that uses the Latex typesetting system language used for the writing of the thesis and of this document.
- **Drawio, for Gantt (GANTT):** online tool used to create the Gantt chart on the next section.

5.4 Estimates and Gantt

5.4.1 Summary of tasks

In table 1 we can see a summary of all the previously specified tasks (only the ones that I am implementing), with their corresponding hours, dependencies and resources needed for each of them.

Table 1: Table summarizing all tasks. Own creation.

Task ID	Task name	Hours	Dependencies	Resources
PM1	Documentation of context and scope	22	-	E, APH, SM, ET, GEP, O
PM2	Documentation of planning	11	PG1	E, APH, SM, GANTT, ET, GEP, O
PM3	Documentation of budget and sustainability	12	PG2	E, APH, SM, ET, GEP, O
PM4	Documentation of final document	7,5	PG3	E, APH, SM, ET, GEP, O
PM5	Documentation of official thesis	90	-	E, APH, SM, ET, GEP, O
PM6	Preparation for presentation	25	PG5	E, ET, O
PM7	Sprint Planning	10	-	E, APH, SM, DT, MT, AD
PM8	Daily Scrum	37,5	-	E, APH, SM, DT, MT, AD
PM9	Refinement	15	-	E, APH, SM, DT, MT, AD

Continued on next page

Table 1 – continued from previous page

Task ID	**Task name**	**Hours**	**Dependencies**	**Resources**
PM10	Sprint Review	5	-	E, APH, SM, DT, MT, AD
PM11	Sprint Retrospective	7,5	-	E, APH, SM, DT, MT, AD
IN1	Plan solution and design data base	24	-	E, APH, SM, DT, VS, DB
IN2	Create new models in db	17	IN1	E, DT, VS, DB
IF1	Get user's info screen from db	19	IN2	E, DT, VS, DB
IF2	Get page configurations from db	7	IF1	E, DT, VS, DB
IF3	Display info screen configuration	38,5	IF2	E, DT, VS, PBI, GR
IC1	Create page for screen's configuration	8	-	E, DT, VS
IC2	Display user's screen configuration	4,5	IC1	E, DT, DB, VS
IC3	Get user's pages from db and display them	9	IC1	E, DT, DB, VS
IC4	Save user's screen configuration	14	IC3	E, DT, DB, VS
IC5	Create page for layout choice	2,5	-	E, DT, VS
IC6	Create pages for new page design	10,5	-	E, DT, VS

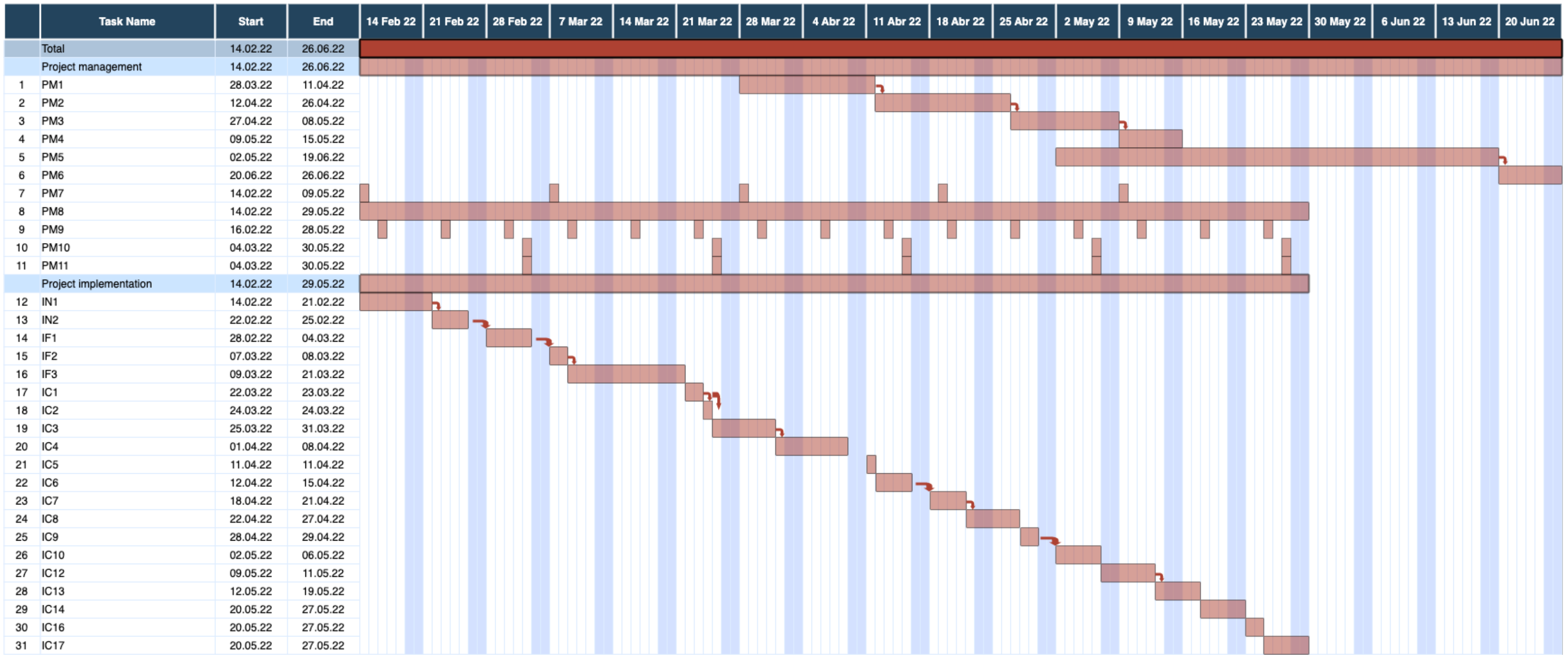
Continued on next page

Table 1 – continued from previous page

Task ID	**Task name**	**Hours**	**Dependencies**	**Resources**
IC7	Get user's page components from db and display them	11	IC6	E, DT, DB, VS
IC8	Save user's new page configuration	12,5	IC7	E, DT, DB, VS
IC9	Create page for new page component	6	-	E, DT, VS
IC10	Save user's new page component	7,75	IC9	E, DT, DB, VS
IC12	Create page to modify an existing page	10	-	E, DT, VS
IC13	Save user's changes on page	10	IC12	E, DT, DB, VS
IC14	Redesign Page Component model on db	8	-	E, DT, DB, VS
IC16	Delete Page Configuration from db	5	IC15	E, DT, DB, VS
IC17	Add preview for new page	8,25	IC7	E, DT, DB, VS
Total	-	**475**	-	-

5.4.2 Gantt chart

In the following chart, in figure 29, all the tasks and their duration are represented throughout the 19 weeks that the thesis project lasts.



	Task Name	Start	End
	Total	14.02.22	26.06.22
	Project management	14.02.22	26.06.22
1	PM1	28.03.22	11.04.22
2	PM2	12.04.22	26.04.22
3	PM3	27.04.22	08.05.22
4	PM4	09.05.22	15.05.22
5	PM5	02.05.22	19.06.22
6	PM6	20.06.22	26.06.22
7	PM7	14.02.22	09.05.22
8	PM8	14.02.22	29.05.22
9	PM9	16.02.22	28.05.22
10	PM10	04.03.22	30.05.22
11	PM11	04.03.22	30.05.22
	Project implementation	14.02.22	29.05.22
12	IN1	14.02.22	21.02.22
13	IN2	22.02.22	25.02.22
14	IF1	28.02.22	04.03.22
15	IF2	07.03.22	08.03.22
16	IF3	09.03.22	21.03.22
17	IC1	22.03.22	23.03.22
18	IC2	24.03.22	24.03.22
19	IC3	25.03.22	31.03.22
20	IC4	01.04.22	08.04.22
21	IC5	11.04.22	11.04.22
22	IC6	12.04.22	15.04.22
23	IC7	18.04.22	21.04.22
24	IC8	22.04.22	27.04.22
25	IC9	28.04.22	29.04.22
26	IC10	02.05.22	06.05.22
27	IC12	09.05.22	11.05.22
28	IC13	12.05.22	19.05.22
29	IC14	20.05.22	27.05.22
30	IC16	20.05.22	27.05.22
31	IC17	20.05.22	27.05.22

Figure 29: Updated Gantt chart with tasks. Own creation

5.5 Risk management

As it was previously mentioned in section 2.5 of this report, there are several potential risks that could appear throughout the development of this project, and it is important to have a solution for them. The following is the proposed plan to manage each of them.

- **Bugs and errors:** To try to avoid bugs and errors or simplify them, tests will be used for the implemented tasks, and specific task planning will be done in each user story. Furthermore, the planning hours for each task give a margin for these, so that the implementation won't be too delayed in case they appear, specially in the first implementation tasks (IF1, IF2 and IF3). And in case of a more complicated issue, two other developers in my team are always accessible for me if I need further help solving anything. This is considered a high impact risk and, in case I still need extra time for solving a bug, I will consider an extra 20 hour-task to solving some of these.
- **Deadline of the thesis:** It might be the case that I underestimate the difficulty of some tasks, and since the thesis delivery has set dates that cannot be moved, it is important to have a plan to meet this deadline. In case that, as I am developing the project, I realize that the project is not going according to plan, I should adapt my initial hour estimation and task organization, as well as the hours dedicated for each of them, to ensure that the project will be correctly finalized. This is considered a low impact risk, since the deadline is known from the beginning, so the planning of the project has been done taking this into account. Still, I will consider 5 extra hours that I may need to re organize the planning where I could, for instance, cut down some hours from IC9 and IC12, simplifying these functionalities.
- **Data privacy of Danfoss:** This is considered a low impact risk, since permission of the company will be asked at the very beginning of the thesis' drafting when it comes to disclosing any kind of private information, and

only superficial data will be included in the report. At the end of the project, before submitting the thesis, further approval will be asked from the company as well. In case that some private data cannot be included, this will be removed from the report or modified in such a way so that no crucial internal information is disclosed. This could cause an extra task of 5 hours in modifying or removing some information from the thesis documentation (which is done in task PM5).

- **Thesis abroad in Denmark:** In the case that I were to bump into any differences between my supervisor's and my UPC professors' opinions, communication with both parties will be key, ensuring that my university's expectations of my thesis and the company's requirements for the project are both met. This is a low impact risk, seeing as it is common for students to do thesis projects with companies in other countries, and Danfoss is used to supervising these kind of projects. I do not consider the need for extra hours to solve this risk.
- **Online meetings with UPC supervisor:** Due to being in different locations, all meetings with my supervisor will be conducted online, which can sometimes challenge communication. In order for them to be efficient and not cause any issues, meetings will be accorded with some time in advance, and an online communication team such as Google Meet will be used for these. I do not consider the need for extra hours to solve this risk.
- **Inexperience in some tools or programming language:** For my project's development I will be using tools and programming languages that I have not worked with in the past (the ASP.NET framework, C sharp programming language, *Azure DevOps*, *Power BI* and *Grafana*). In order to prevent big risks, the firsts tasks that I will implement have a bigger amount of hours estimated for them (IF1, IF2 and IF3), so that I have a learning margin. In case of further issues or a lack of knowledge throughout the development of the project, I will be in closer contact with other developers in my team who have expertise with these tools, and they would be able to help me so that this

does not delay my work. However, I will consider an extra 20 hours to manage this risk, as I may still need more time to gain knowledge on the technologies.

Chapter 6

Budget

6.1 Identification and estimate of costs

In this section I will consider all the elements that influence the budget for my project. This includes the personnel costs, the generic costs (such as amortization, software, hardware, etc.) and others (such as contingencies and incidentals).

6.1.1 Personnel costs per activity

In this section I will specify and calculate the costs for each task specified in section 5.4.1, and in the end the total cost of the staff (CPA). To do this I need to estimate the cost per hour of each role that is involved in my project. The roles that I identified for the project's development are the following.

The **project manager**, who takes care of the planning and structure of the project (in this case, the role will be played mostly by me, but also shared with my supervisor from Danfoss, my thesis supervisor from UPC, and the GEP professor). Then there is the **software developer**, who will carry out the implementation of the info screen solution. The **tester** will verify the correct functionality of some tasks of the project. And finally, the **technical writer** will be the person who documents the whole project development. All of these last roles will be played by me.

Role	Annual salary (€)	Price per hour (€)
Project manager (PM)	78.978	44,47
Software developer (SD)	52.734	29,69
Software tester (ST)	42.961	24,19
Technical writer (TW)	51.678	29,10

Table 2: Annual salary of each role in Denmark. Source [20]

In table 2 I have specified the average annual salaries for each role. In Denmark, there is no set extra amount that the employer must pay for social security, but it is rather handled through an 8% tax called *am-bidrag* [21] that is deducted from the monthly salary from the costs that appear on the table. Now, using this information seen on the table, I will calculate the specific cost per task, which can be seen in table 3.

6.1.2 Generic costs

Amortization

When making the budget for a software project, it is important to take into account the amortization of the material resources used for it. My work laptop has a life expectancy **(LF)** of 4 years, and my personal laptop of 7 years, and there are 253 working days this year in Denmark, with an average of 5 hours of work on the project per day. Then I will take into account the original cost of each resource **(OC)**, and the amount of time that I will use it for **(TU)** (my work laptop will be the main one, used for an 80% of the project). In the end I can calculate the amortization with the following formula:

$$\text{Amortization} = (\text{OC}/(\text{LF} * 253 * 5)) * (\text{TU}) \tag{6.1}$$

The resulting costs can be seen on table 4

Task	**Hours**	**Hours PM**	**Hours SD**	**Hours ST**	**Hours TW**	**Cost**
PM1	22	15	0	0	7	870,75
PM2	11	6	0	0	5	412,32
PM3	12	7	0	0	5	456,79
PM4	7,5	2	0	0	5,5	248,99
PM5	90	20	0	0	70	2926,4
PM6	25	25	0	0	0	1111,75
PM7	10	10	0	0	0	444,7
PM8	37,5	37,5	0	0	0	1667,63
PM9	15	15	0	0	0	667,05
PM10	5	5	0	0	0	222,35
PM11	7,5	7,5	0	0	0	333,53
IN1	24	0	24	0	0	712,56
IN2	17	0	13	4	0	482,73
IF1	19	0	16	3	0	547,61
IF2	7	0	5	2	0	196,83
IF3	38,5	0	31	7,5	0	1101,82
IC1	8	0	8	0	0	237,52
IC2	4,5	0	3	1,5	0	125,36
IC3	9	0	7	2	0	256,21
IC4	14	0	11	3	0	399,16
IC5	2,5	0	2,5	0	0	74,23
IC6	10,5	0	9	1,5	0	303,50
IC7	11	0	8	3	0	310,09
IC8	12,5	0	9	3,5	0	351,88
IC9	6	0	6	0	0	178,14
IC10	7,75	0	5,5	2,25	0	217,72
IC12	10	0	8	2	0	285,9
IC13	10	0	7	3	0	280,4
IC14	8	0	6	2	0	226,52
IC16	5	0	4	1	0	130,95
IC17	8,25	0	7	1,25	0	217,07
Total	**475**	**-**	**-**	**-**	**-**	**15.998,46**

Table 3: Estimated cost per task for each role. Own calculations.

Resource	**Price(€)**	**Years**	**Time used**	**Amortization(€)**
Lenovo ThinkPad T15	1.500	4	380	112,65
Macbook Pro 2017	1.200	7	95	12,87
Total	**-**	**-**	**-**	**125,52**

Table 4: Amortization costs for hardware resources. Own calculations.

Resource	**Price per month (€)**
Visual Studio IDE + Azure DevOps	43
Microsoft 365	34
Power BI	8,40
Total	**85,4**

Table 5: Costs for software resources. Own calculations.

Software

Some of the software resources used have a set monthly cost, these have been specified on table 5. The total cost for software resources is of 85,40 €.

Electricity

The current price in Denmark of electricity is of 0.12 € per kWh (source [22]). My work laptop has an average consumption of 66 W and it is used for a total of 380 hours (25,08 kWh then), so its total electricity cost is of 3,01 €. My personal laptop's consumption is of 61 W, and it is used for a total of 95 hours (5,795 kWh), then its total cost of electricity is 0,70 €. All in all, the electricity price is of 3,71 €.

Internet

The average price for Internet is of 70 € per month, and the project's hours are a total of 475. Then the total Internet cost is of 46,18 €.

Commuting and travel

Most days I will be working from my office, and in order to get there I use public transportation. The monthly bus ticket to travel between where I live and the office costs 92,72 €, and it gives me an unlimited number of travels. So the final cost for commuting during the months of the project is of 463,6 €.

Furthermore, I will need to travel to Spain and back to Denmark in the end of June, in order to defend my thesis in person at UPC. This cost is not one hundred percent accurate, since flight prices vary constantly, but based on current prices my trip from Denmark to Barcelona and back will cost around 300 €.

All in all, the total cost for commuting and travel is 763,6 €.

Summary of generic costs

The following table 6 summarizes all the previously calculated generic costs, and shows the sum of all of them (GC), which is a total of 1024,41 €.

Cost	**Total amount (€)**
Amortization	125,52
Software	85,4
Electricity	3,71
Internet	46,18
Commuting and travel	763,6
Total	**1.024,41**

Table 6: Total generic costs. Own calculations.

6.1.3 Other costs

Contingencies

In order to be able to face delays or other unexpected issues, a contingency fund will be calculated for the project. This given contingency will be of 15% of the total amount of the PCA and the GC costs, which is 17022,87 €. So the final contingency cost is of 2553,43 €.

Incidentals

As it was described in section 5.5, there are some risks that must be taken into account when planning this project's budget. Depending on the risk and its management plan, an extra cost has been calculated for some of these, which can be seen below in table 7.

The estimated cost is calculated based on the salary of the different roles involved in the task (in some risks, more than one role has to be involved).

Risk	**Probability**	**Hours**	**Estimated cost (€)**	**Cost (€)**
Bugs and errors	40%	20	566,3	226,52
Deadline of thesis	10%	5	183,93	18,40
Data privacy	10%	5	145,5	14,55
Inexperience	30%	20	593,8	178,14
Total	**-**	**-**	**-**	**437,61**

Table 7: Incidentals budget. Own calculations.

6.1.4 Final budget

Below is a summary of all the previously calculated costs for this project, which end up on a total of 20013,91 €, as it is seen in table 8.

Cost	**Amount (€)**
PCA	15.998,46
GC	1024,41
Contingency	2553,43
Incidentals	437,61
Total	**20013,91**

Table 8: Final budget. Own calculations.

6.2 Management of costs

In this section I will share the controlling mechanisms for potential budget deviations. In order to manage the costs and, in the end of the project be able to compare how accurate my budget was, I will then be calculating several deviation indicators.

The deviation refers to the difference between the estimated amount of a value I considered for the budget, and its real amount, that I will know in the end of the project.

I will be focusing on controlling the real amount of hours for each task and, based on that, I will also be able to calculate the deviations in the cost per task, in the total hours, and in the total costs. Note that if the deviation in the cost of a task is negative, I will need to allocate part of the contingency fund to cover this. If it was positive, which means that the time of the task has been overestimated, I will have the chance to use the extra money to cover other incidences.

- **Deviation in hours per task:**

$$\text{estimated hours per task} - \text{real hours per task} \tag{6.2}$$

- **Deviation in (personnel) cost per task:**

$$\text{estimated cost per task} - \text{real cost per task} \tag{6.3}$$

- **Deviation in total hours:**

$$\text{estimated total hours} - \text{real total hours} \tag{6.4}$$

- **Deviation in total costs:**

$$\text{estimated total costs} - \text{real total costs} \tag{6.5}$$

6.3 Laws and regulations

In order to correctly develop this project, I have considered the possible laws and regulations that could affect it. The most relevant ones that are related to it are those in the area of data privacy. As the info screen interacts with users of the company and accesses and stores some of their information, it is important to comply with the General Data Protection Regulation (GDPR), which is the European Union's most relevant law when it comes to data privacy of users.

Fortunately, the user accounts are managed by the company and are connected to a private directory, so no critical personal information is actually stored into the project's data base, but rather only fields that are necessary to connect them to an info screen (their name, id and info screen ID). The connection to the users accounts is done automatically through their *Azure DevOps* credentials, the same way my team already does this on many other tools they created. Also, there is no interaction between users, so the possibility to share private data among them is infeasible.

Chapter 7

Sustainability

7.1 Self-assessment

The term *sustainability* is seen everywhere nowadays. This is not a bad thing -even though sometimes it is used with the wrong intentions-, to the contrary, it means we are becoming more aware of it. But most of the time we are being called out on the things we buy, weather we recycle or not, how much plastic we consume... so it can be surprising, and refreshing, to see sustainability being considered in the area of software engineering.

I cannot say that throughout the span of my studies I have felt that any course gave much matter to this big issue. It might have been mentioned in some courses on specific occasions, but after answering the questions on the form I do not believe we have been educated enough on the sustainability of any software (or computer engineering) project.

From what the form asked, I felt like some areas I had more knowledge on (such as the understanding of the concepts of the three dimensions), but others, specially when it comes to measuring the environmental and social impact, and using indicators for it, I am lacking the same level of understanding.

But overall I am glad this section and form are included in the making of this document, since it makes sure that all of us students take sustainability into account, and are aware of the economic, environmental and social costs of our actions.

7.2 Economic dimension

Regarding PPP: Reflection on the cost you have estimated for the completion of the project

In any software development project, the cost and budget of it are very decisive factors to take into account, so the full cost of this project has been estimated in section 6.1, both in terms of material resources and also human resources, so that I can have an idea beforehand of what to expect from it.

I believe the budget of the project is realistic and reasonable, and I don't think there are any resources that could be changed in order to minimize the budget (without having a negative impact in a different dimension).

Regarding Useful Life: How are currently solved economic issues (costs...) related to the problem that you want to address (state of the art)? How will your solution improve economic issues (costs ...) with respect other existing solutions?

The creation of this info screen would be a solution that will allow developers at Danfoss to visualize data in a more clear way, and combine different inputs into one. Having access to all this data in one big screen can improve their working efficiency and make their daily work easier, specially when it comes to monitoring daily results from tests, pull requests, etc.

7.3 Environmental dimension

Regarding PPP: Have you estimated the environmental impact of the project?

The environmental impact of this project has not been calculated and no indicators have been used in order to quantify it.

Regarding PPP: Did you plan to minimize its impact, for example, by reusing resources?

The project itself does actually use an existing website and data base around which it is based, so the extra environmental impact that it creates is much smaller compared to a project where a new website (with server) and data base is needed.

But I do think that it would be possible to minimize the project's impact if my university allowed me to defend my thesis on-line, or at a different time. This would avoid the need for me to travel back to Barcelona and then again to Denmark, reducing my environmental impact since I would not need to take flights.

Regarding Useful Life: How is currently solved the problem that you want to address (state of the art)? How will your solution improve the environment with respect other existing solutions?

Even though I don't believe this solution has a great improvement on the environment (nor does it cause a big deterioration), I would assume that if developers had a configured screen with their different data sources all in one place, they would need to spend less time going on the different websites (the sources) and therefore, using less electricity to consult the same information.

7.4 Social dimension

Regarding PPP: What do you think you will achieve -in terms of personal growth- from doing this project?

I believe this project will allow me to learn a lot in different areas: the most obvious one, in software development, since I will be using a new technology and frameworks I have never worked with in the past. But it will also allow me to know what working in a real team, as part of a multinational company, feels like, and learn from Agile and Scrum practices daily. Finally, I will also be able to learn about the work behind Danfoss' products and how they impact the environment for the better.

Regarding Useful Life: How is currently solved the problem that you want

to address (state of the art)? How will your solution improve the quality of life (social dimension) with respect other existing solutions?

The project will help developers and teams in the Software department at Danfoss Drives on their daily monitoring tasks, therefore making a small part of their job easier, facilitating a way to visualize data from different sources.

Regarding Useful Life: Is there a real need for the project?

Yes, since the project comes from a need that the *DevOps* team itself felt internally, but they also realized this could be beneficial for many other teams and developers. This may not be an urgent problem to solve, but it is a solution that could improve the way teams work and benefit them by facilitating a way to visualize lots of data in an easy way.

Chapter 8

Knowledge integration

This project has been officially described in my faculty's system with a set of technical competencies, that go hand in hand to many Software Engineering projects. In this chapter I will go over the selected competencies and I will justify how they are in fact related to the project.

- **CES1.1:** *Develop, maintain and evaluate complex and / or critical software systems and services.* This project develops and maintains a part of a website, that interacts with users and a data base and combines many external sources of information.
- **CES1.3:** *Identify, evaluate and manage potential risks associated with the construction of software that may arise.* Risks that could appear throughout the project's development have been considered in the planning of this project, and management plans have been made as well.
- **CES1.5:** *Specify, design, implement and evaluate databases.* An extension of an existing data base has been designed and implemented to support the data that needs to be stored for the project.
- **CES1.6:** *Administrate databases.* The previously mentioned data base is regularly handled and accessed by methods in the website project.

- **CES1.7:** *Quality control and design testing in software production.* Unitary tests are carried out for the data base and the main functionalities of the project, assuring a level of quality.

- **CES2.1:** *Define and manage the requirements of a software system.* Both functional and non-functional requirements have been specified before the implementation of this project, as it has been seen in previous chapters of this document.

- **CES2.2:** *Design appropriate solutions in one or more application domains, using software engineering methods that integrate ethical, social, legal, and economic aspects.* The planning of this project has taken into account economic costs, sustainability practices and interaction with private user information.

Chapter 9

Conclusions

9.1 Achievement of objectives

At the beginning of the project, several objectives were defined, that were all related to one another, and after finalising it we can see how these have been achieved.

The monitoring solution offers a new tool that allows different users and teams to configure their screens, where they can visualize customized information, that comes from different sources: Power BI graphs, Grafana dashboards or their own annotations.

At the same time by providing this info screen to other teams, the use of other existing tools and information dashboards is being promoted and facilitated: users know what type of components they can check, and all they need to do is add a link and a name to them and put them into pages. Developers who are users of the info screen will no longer need to check data one by one, when this comes from different places, since now they can visualize relevant data all in one place.

9.2 Achievement of scope

All the functionalities that were specified in the scope of the project have been fully developed. Furthermore, due to an unexpected new member joining the team during the last two weeks of the project, there was even a chance to add a couple other tasks that fitted perfectly into the scope.

The general estimation in terms of hours for the tasks was pretty accurate since, although some tasks took a bit longer than it was planned, there were others that were also slightly shorter, so overall they balanced themselves out. I believe that it was a good decision to have estimated a bigger amount of hours to the first tasks of the project, given that I did feel like there was a longer learning process during the first weeks, during which I had to learn how to use the new framework and how the existing Viking DevOps project was structured.

9.3 Future plans for the project

Although the project specified for this thesis has a set scope, the Info screen solution will be further developed. Since my internship at Danfoss lasts longer than what this thesis takes, I will have the chance to implement further functionalities and add more features to the solution. Furthermore, once I finish the internship, the new intern that joined the team in the recent weeks will be taking over my tasks and maintain and expand the project.

Some functionalities that are already being considered for future development are: adding new types of page components, such as Azure DevOps pages or graphs, or pipeline graphs; allowing users to share pages between them, so that they can re-use other people's pages if that is of interest to them; adding filters to a user's page configurations in the settings page; deleting page components; adding other fields to page configurations, such as a description... These are all initial ideas, but it is expected to implement other functionalities that may also come up after putting the project in use with other teams and getting feedback from them.

9.4 Personal conclusions

To finalize this document, I would like to look back on the project's development and share some personal thoughts on it.

Firstly, partaking on this internship and implementing this solution has been a great learning experience. Although I had already taken part in courses around web development in the past year at university, developing a project on a bigger scale has allowed me to learn a lot, especially since I was also introduced to a new framework and programming languages, which has given me new expertise in something I had not used before.

Secondly, by working within the DevOps team during this period of time (attending daily meetings, hearing about what they do...) I have also gained an insight into a completely different area and working practices that I knew nothing about before. And I have also been able to use Agile and Scrum methodologies first hand, which I had already gotten a glimpse of in many courses, but I finally put them into practice in a real team.

Lastly, it has been a very satisfactory experience to develop this solution that has a purpose and, even if it's just a little bit, can help make other developers' daily work easier. It is also exciting to think that the resulting product has a lot of potential and will be used for a long period of time in the future, giving a motivation for the development of this project.

List of Figures

List of Tables

Bibliography

[1] About danfoss. URL https://www.danfoss.com/en/about-danfoss/company/danfoss-at-a-glance/.

[2] Drives. URL https://www.danfoss.com/en/about-danfoss/our-businesses/drives/.

[3] What is devops? devops explained: Microsoft azure. URL https://azure.microsoft.com/en-us/overview/what-is-devops/.

[4] Atlassian. Devops pipeline. URL https://www.atlassian.com/devops/devops-tools/devops-pipeline.

[5] What is a pull request? (2021). URL https://www.pagerduty.com/resources/learn/what-is-a-pull-request/.

[6] Azure devops server (2021). URL https://en.wikipedia.org/wiki/Azure_DevOps_Server.

[7] Mihart. What is power bi? - power bi. URL https://docs.microsoft.com/en-us/power-bi/fundamentals/power-bi-overview.

[8] Grafana® features. URL https://grafana.com/grafana/?plcmt=footer.

[9] Asp.net: Open-source web framework for .net. URL https://dotnet.microsoft.com/en-us/apps/aspnet.

[10] Atlassian. User stories: Examples and template. URL https://www.atlassian.com/es/agile/project-management/user-stories.

[11] Primary keys. URL https://www.ibm.com/docs/en/iodg/11.3?topic=reference-primary-keys.

[12] Foreign key sql. URL https://www.w3schools.com/sql/sql_foreignkey.asp.

[13] Working with json - learn web development: Mdn. URL https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON.

[14] KathrynEE. What is azure boards? tools to manage software development projects. - azure boards. URL https://docs.microsoft.com/en-us/azure/devops/boards/get-started/what-is-azure-boards?view=azure-devops.

[15] Job, J. Scrum diagram - the scrum framework captured in simple one picture (2020). URL https://jordanjob.me/blog/scrum-diagram/.

[16] .net framework (2021). URL https://r6solution.com/net-framework-c/.

[17] Ardalis. Common web application architectures. URL https://docs.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures.

[18] What is entity framework? URL https://www.entityframeworktutorial.net/what-is-entityframework.aspx.

[19] JonDouglas. What is nuget and what does it do? URL https://docs.microsoft.com/en-us/nuget/what-is-nuget.

[20] Salary data amp; career research center (denmark). URL https://www.payscale.com/research/DK/Country=Denmark/Salary.

[21] Danish taxes and the taxation system in denmark. URL https://www.oresunddirekt.se/en/working-in-denmark/taxes/danish-income-tax-a-short-introduction.

[22] Statbank denmark. URL https://www.statbank.dk/.