SE Project
Documentation

Consilium

Semester: Fall 2023

Project Team: Joel Sauvain
Noah Stalder

Project Advisor: Olaf Zimmermann

School of Computer Science
OST Eastern Switzerland University of Applied Sciences

Abstract

Project management and organizational software is a complicated domain and already a well-saturated and competitive market. Due to the various levels of technical affinity, user-friendliness is a key aspect for software, trying to gain a foothold in this sector. Especially for smaller ventures, and freelancers, project management and administrative tasks are often not their main activities, but something keeping them from spending more time doing what they really want. On top of that, project management, creating offers and invoices are often only seen as a means to an end and not something to be enjoyed. Therefore, project management software should make the lives of their users easier, by reducing the overhead to a minimum. Tools in this domain should offer broad functionality while not getting overly complicated to use.

This thesis identifies smaller ventures and freelancers as the perfect market segment to target with software tailor-made for their needs through a market analysis. *Consilium* addresses their needs by uniting different tasks, previously divided across multiple tools into one end-to-end software solution. While multiple freelancers and small ventures have previously been creating their offers with Word, tracking their efforts with Excel and making their notes in a text editor, they can now just use *Consilium*. *Consilium* identified the synergies that can be used between those various administrative tasks to make them simpler. By introducing innovative concepts, *Consilium* managed to become a well-rounded administrative tool, that can learn with the users, offering functionality from creating offers to creating invoices, while maintaining sufficient depth to model even complex workflows. *Consilium* is integrated into the AWS cloud and offers uncomplicated login functionality allowing potential users to sign up in seconds, using social logins. While primarily targeting freelancers, Workspaces are implemented, allowing users to collaborate. On top of all the functionality offered, *Consilium* maintains a high level of customizability, allowing for custom branding on exported documents and content to be added using a Rich text editor, which is then translated and exported into a PDF. *Consilium* manages to reduce the administrative overhead significantly by building smart workflows, through clever reuse of data. Early user tests show, Consilium is already in a state, which helps freelancers improve their productivity, by streamlining processes, showing significant improvements, compared to their previous processes.

Consilium has the potential to become even more. By introducing a customer-view *Consilium* could become a marketplace, where users could tender projects and freelancers could offer their work. Here, a match-making algorithm could achieve groundbreaking results for freelancers and make finding someone to implement your project extremely easy. Another interesting lane to go down would be to add the feature where customers could approach freelancers with a potential project description which through conversational artificial intelligence could be translated into an offer draft for the service provider.

Management Summary

This management summary provides an overview of the *Consilium* project, suitable for all interested parties. It focuses on the project's context, approach, strengths, and achievements, while also highlighting the potential future impact.

Context

The *Consilium* project, undertaken at the OST Eastern Switzerland University of Applied Sciences, marks a significant advancement of the prototype developed in the SE Project module. This prototype already offered a basic way to manage customers and projects and generate offers. Through a market analysis, we identified a target customer segment, which heavily influenced the direction of this project. This thesis takes the project management software prototype from the SE Project. It aims at implementing specific improvements, to the functionality, the architecture, and the user-friendliness, all to have the best possible chances for a successful market entry. Guided by project advisors and effectively executed by Joel Sauvain and Noah Stalder, this initiative aimed to address the growing need for efficient data management in small to medium enterprises or freelancers. The project set itself the ambitious goal of supporting complex workflows while at the same time making simpler workflows as easy as ever. On top of that, this thesis will move the previously local application into the cloud, integrate third-party components, and enable multi-user ability.

Approach

After identifying the key areas where improvements were needed through the market analysis, we took a multi-faceted approach, incorporating inputs from representatives of the identified customer segment, user-centric design principles, and agile project management techniques. This allowed us to develop a solution that not only meets the immediate needs but is also scalable and adaptable to future requirements. During the whole course of the conception and development phase, there was a big emphasis on software architecture, to keep the application and the functionality maintainable and extendable. The biggest impediment the project faced was the resources, limited by the scope of this thesis. This led to project management but also being creative being an integral part of this thesis, as the project team was faced with the challenge of how the available resources were best invested. Throughout the whole project, the team managed to find creative ways, of how sometimes multiple requirements at once can be met, with relatively little effort required.

Results

Key outcomes of the *Consilium* project include a user-friendly interface, improved data processing speeds, and robust security measures. These results were benchmarked against our initial goals and demonstrated a significant improvement in operational efficiency and user engagement. On the surface, the improvements in features and the addition of new functionality, allow for complex, end-to-end flows, starting with the generation of an offer and ending with the creation of invoices, based on the actual work invested and logged.

Consilium managed the leap into the Cloud effortlessly and had a simple, yet mighty multiuser concept introduced allowing for different levels of collaboration. Users can now create workspaces, and invite fellow collaborators with varying levels of access, with the invited users being notified by email.

A big improvement was achieved by professionalizing the experience for the users. The possibility of internationalization has been added and multiple languages are now supported. A big step has also been taken with minor tweaks to the User-Interface and more data being collected, processed and then graphically displayed for the user.

In addition to the improvements aimed at making the experience of the users more professional, a big step forward could also be taken by enhancing the look and feel of the documents generated with *Consilium* which our users send to their client. We view this improvement as integral, as we identified the importance of the quality of the documents the users provide their customers. Early rounds of requirements engineering showed, that our potential customers could not be satisfied with the basic implementation of the offer feature. Here, the team could display their innovative problem-solving skills, by finding a simple implementation which meets a whole array of requirements. Instead of implementing each requirement on their own, a solution could be found to address multiple requirements, while uniting the ease of use of the so-called *Offer Building Blocks* with the now growing possibilities and complexity.

Another cornerstone is marked by the implementation of invoices. This adds a new dimension to the workflows, possible with *Consilium*. It's now possible to precisely log the efforts invested, monitor them and then send bills to the customers using QR-code invoices.

When looking under the hood, the results are also impressive. The architecture of the application is well-thought-through and is clearly separated. In all areas, the application was already developed in a way which allows for further functionality and features. For example, at many points in the application, data is collected which is not yet processed. This was done, with future features in mind. The project's success is not just in its technical achievements but also in its potential for broader application. *Consilium* has the potential to revolutionize how small businesses approach data management, offering a model that can be replicated and customized across various industries.

Outlook

As we look to the future, the insights and technologies developed through the *Consilium* project could serve as a base for further implementation. *Consilium* is now at a stage where, it already is the best solution available for a lot of use cases. With a few minor tweaks, *Consilium* could take a leap and would have a chance to be competitive on the market. With more time at hand, there are many features we would like to implement.

When taking the project further, and trying a market entry, first a pricing model would need to be defined. In addition, we identified, that further work on user experience improvements would be needed, as our research showed.

Taking a step back, the most interesting direction we would like to take *Consilium* in, would be transforming it into a marketplace platform. We can envision *Consilium* being a place, where customers invite tenders for projects or directly approach service providers on the platform. *Consilium* would be suitable for this kind of application, as many data and views necessary for both of those views are already present. Through the use of artificial intelligence, we could develop a match-making algorithm, which would bring suitable customers and service providers together.

Acknowledgements

First of all, we would like to thank our advisor, Olaf Zimmermann for his support and passion. Not only did he provide invaluable feedback throughout this thesis, but he also acted as a member of our target audience and brought up valuable suggestions throughout. Not only did that help with the requirements engineering, but it was also a considerable boost in motivation.

Furthermore, we would like to express our gratitude to Joshua Beny Hürzeler, who started *Consilium* with us as part of the SE Project and allowed us to continue to work on the project, and provided valuable feedback throughout.

A special mention goes out to the founder of jOOQ, Lukas Eder who we had the pleasure to meet and once again made our lives easier through jOOQ.

Finally, we are very grateful to our supportive friend Kay Mogg for his incredible input and feedback giving us invaluable insights into the world of a freelancer.

Contents

List of Figures

Chapter 1

Introduction

This chapter serves as an introduction to this thesis. It gives an overview character and structure of this thesis and gives some insights into the solution.

The thesis started by working out an initial project definition with our advisor. The original project definition can be found in the Appendix A and will now be summarized. This thesis is a continuation of a project management application prototype started in the Software engineering project. Information to gain an overview of what was already in place before this thesis can be found in Section 2.1. In the project definition two main tasks and goals were defined: First, the architecture of the application should be designed in a way, which allows a continuous and efficient cloud operation with one of the leading, public cloud providers. The second main goal was, to analyze and implement functionality to gain an edge over competitors in this market.

With the project definition in hand, the first tasks of this thesis were analytical. The processes, the methodologies applied, and the results of the analysis can be found in Chapter 2. Required for a successful outcome of our requirements engineering was, getting to know the market and our competitors. We identified, that it is instrumental that we have a clear idea of our market segment, the needs of customers in this customer segment, and how we can provide a better solution for those customers than any other competitor in the market. This was done by a thorough market analysis, which can be found in Section 2.2. The results of the market analysis showed, that freelancers and smaller ventures are the perfect customer segment to be targeted. From there, we worked together with representatives of this customer segment to identify their needs. They described their previous workflows which helped us identify how we could improve their processes best. We then gave them a little play around with the prototype to get familiar with the basis of the application and asked them to provide a list of functionality and requirements they would benefit from. To get an even broader basis for our requirements, we defined personas (Section 2.3.1) so we do not run the risk of building the perfect tool for only a few people which is not beneficial for most others. With all this input gathered and refined, the requirements were refined: Section 2.3.

With the requirements defined, we began working on the concepts of how we want to achieve our goals and fulfill our requirements: Chapter 3. This evolves around designing the application in a way that is stable, extendable, and maintainable. That Chapter highlights the design and architecture principles we apply and how the application is made cloud-ready. We put a big emphasis on the design and conceptual phase of this thesis as it was one thing to just implement and thereby fulfill the requirements, but we demanded more from ourselves. Before starting to implement, we always wanted to create solid concepts, which make the workflow as easy and smooth as possible, while maintaining the possibility to support different workflows. The goal is to assist the user, while not restricting him to

one workflow, which may not fit. To achieve that, we identify synergies between different domains and processes modeled within *Consilium* to make the processes as straightforward as possible and provide as much assistance as we can while maintaining flexibility. The technical concepts defined to achieve the set-out goals and fulfill the requirements are listed in Section 3.2.

Building upon the technical concepts, we started with the implementation. The Chapter 4 highlights the most difficult and interesting parts of the implementation. Key challenges in the implementation included difficult refactorings and re-coupling and decoupling of previously tightly coupled components and entities. It shows integrations of external components and innovative approaches. Feedback, collected through user tests with representatives of our target segment, was also steadily incorporated into the implementation.

Finally, the results are presented in Chapter 5. The results are shown but also put into the context of which requirements they aim to fulfill. In addition, the representatives of our target segments were once again tasked, to evaluate and assess *Consilium*. This was done to determine the true value of our work, as it is one thing to match them against previously defined requirements, but it is possibly even more important to get real-world insights into how much *Consilium* can help freelancers and smaller ventures.

Chapter 2

Analysis and Requirements

This chapter explains the analytical process, the requirements, and how they were generated.

2.1 Foundation

To get a better overview of what was achieved and what could be achieved in this project, it is necessary to disclose what has already been in place before the start of this thesis. This is done in this section, which highlights and honours the work done in advance.

2.1.1 Initial Development in the Software Engineering Project

The inception of *Consilium* was marked during a Software Engineering (SE) Project, where the primary goal was to develop a functional prototype. This phase served as a foundational bedrock, demonstrating the potential of the concept. The intention was to create a prototype that was operational for practical scenarios, although within a confined scope. However, the limited timeframe of the SE Project produced several code flaws and unfinished features.

2.1.2 Transition to Study Assignment (SA): Preparing for Market Entry

In the current stage, as part of a Study Assignment (SA), the development of *Consilium* pivots towards a market-entry analysis, aiming to refine the software for an impending release.

The evolution of *Consilium*, from a SE Project prototype to a market-ready product in the SA, illustrates a significant developmental arc in software engineering. The strategic shift to a cloud-based infrastructure and the focus on collaborative features underscore a commitment to adapt to market demands and user requirements.

Existing Software

The prototype of the software supports the following user stories:

- Create a customer (either company or person)
- Create a project
- Create offers on a given project and update it
- Adjust offer template with icon and footer
- Simple time tracking on project
- Simple note-taking on project

Git Release

A release tag has been set on GitLab to provide a final codebase from SE-Project. The release tag is named ‘release-1.0‘ for the prototype. Everything named prototype will reference to release-1.0.

Database Model

To demonstrate effective changes made from the SE Project prototype, we will illustrate the database model in Figure 2.1 to the time of the ‘release-1.0‘ state.

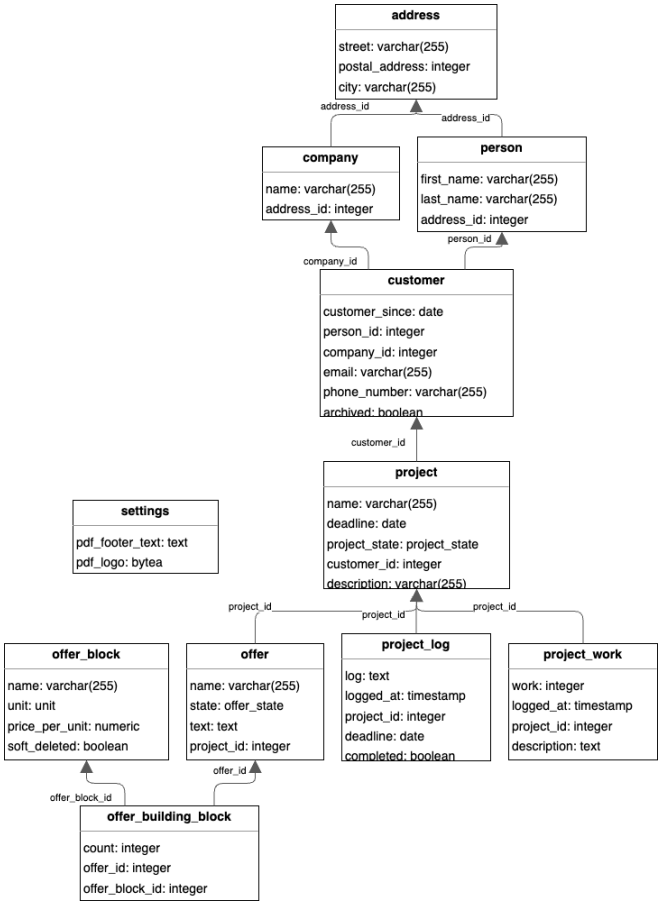

Figure 2.1: Database Model of Prototype

Problems of Current Solution

As the software has been built as an MVP proof of concept there are a few problems and flaws in the current software. This includes, but is not limited to, code flaws, bugs, unfinished features and missing data. These problems must be tackled first before we can start adding new features to the software.

2.1.3 Visual Comparison

To showcase the visual differences from the prototype, we created an extensive comparison, which can be found in Appendix D.

2.2 Market Analysis

As the project management tools market is a competitive sector, it is instrumental to the success of *Consilium* to identify a sub-sector and a specific target segment where we can gain a foothold by explicitly targeting their needs. Therefore, a thorough market analysis is necessary to identify sub-sectors of the market where there is demand, but it is not already saturated by competitors. First, a broad description of the target market is given, from where the target audience is derived from. From there, competitors are evaluated with the aforementioned market and target audience in mind. This all leads up to the conclusion where the areas are listed in which a competitive edge over our competitors could be gained, by meeting the specific needs of our target segment.

2.2.1 Market Description

As mentioned, the project management platform market is overall already saturated with a broad selection of tools already in place. Key to the success of *Consilium* is, identifying a sub-sector of this market, which is not already specifically targeted but is also due to grow in the coming years, as gaining a foothold in a dying sector is not sensible.

The market sub-sector identified is project management software for small businesses and independent individuals. This sub-sector was seen as the most promising for a number of reasons. In contrast to the needs of big corporations, smaller businesses and individuals tend to have more steps of the project management process done by one single person. In a big corporation, there would usually be a salesperson tasked with bringing in customers offering services and closing deals. There is often an account or client relation manager, managing the clients. A project or team leader is tasked with monitoring the progress across tasks or projects. There is usually a dedicated back office team, billing the work done by the team. For each of those steps and components of the project management process, there are already dedicated tools, exceptional at what they do. Contrary to big corporations, with smaller businesses or independent individuals there is no such separation of concerns across multiple individuals or even teams, or not to this extent.

When there is no such separation of concerns, using a separate tool for each project management discipline, is not a satisfying workflow for an individual. Therefore, this is where we see the potential for *Consilium* to come into play to provide a solution for smaller businesses and independent individuals, guiding and supporting them through each step of the project management process. As it is impossible to match each competitor in their best discipline, small businesses and independents are perfect, as their needs are more basic in each discipline and they do not have complicated processes as a bigger corporation would have. *Consilium* directly targets ventures looking for the first step to professionalize their operations. The ventures we are targeting could have been using Word to create their offers, logging their work by noting it down in notepads, keeping track of their deadlines via calendar entries

and invoicing their efforts themselves. *Consilium* enables you to easily create offers, send out bills and manage deadlines, without having to worry about legal or formal aspects. And everything is possible within one single tool. A key factor in satisfying this market requires little to no effort to get up and started.

Identifying the target market segment brought up questions which we need to be able to answer to stand a chance: How does *Consilium* differentiate itself from its competitors? Are there already any valuable features which give us an edge against the competition or must they still be identified?

Target Audience

This section wants to precise our target group and find out about other opportunities. It is an extensive breakdown into specific categories.

Age: Younger technically advanced people Geolocation: Switzerland Job: self-employed or small business Sector: Photography, Videography, Software Development, UX Design, Marketing, Digital Content Creation. Not limited to these Finances: Limited to no financial resources

We see younger and technically well-versed individuals as especially promising, as they are generally more open to jump onto new products and try them out to see if it is something that could benefit them. First up, location-wise the focus should be on Switzerland as we can easily spread the use of *Consilium* here. However, there is no limitation location-wise as the tool is available in English. The types of jobs which our target audience would mainly be working on were derived from the market description. The sectors which we see as most promising, are selected based on the potential for individuals and small, up-and-coming ventures to gain a foothold. We see those sectors as having the most potential for ventures to be in the position where we can help them best with taking the next step. Finances-wise, we see our target segment as having limited resources and would be looking for a tool they can use before they bring in enough money to invest in an expensive tool.

2.2.2 Market Size

To identify the potential of *Consilium*, it is essential to estimate the size of the market we are acting in.

If we are analyzing the entire market of project management software (PMS), the market size is estimated at 6 billion USD. Additionally, a growth of 10% is expected from 2023–2028. There are some big players such as Oracle, Microsoft and SAP. However, the rest of the market is finely divided between lots of different solutions.

As we are limiting ourselves to Switzerland for now, we have an estimate of half a million self-employed individuals. Additionally, there are also half a million micro-companies (less than 10 employees). With the estimated size of a million potential customers, we have to keep in mind, that lots of them are not working in a project-based environment.

2.2.3 Competition Analysis

We analyze two different solutions in the Swiss market, which operate in a similar section of the market as we intend to do.

Bexio

`bexio.com`

Bexio is an integrated business management software designed for small and medium-sized enterprises (SMEs) in Switzerland. It offers a comprehensive suite of tools covering accounting, invoicing, CRM, and more. Bexio is known for its user-friendly interface, Swiss localization, and integration with Swiss-specific services. It emphasizes data security and offers various pricing plans. Businesses can benefit from its tailored features, scalability, and local support.

They have a three-tier pricing model. Starting at 35.- CHF for a Starter version, and going up to 115.- CHF for the PRO+ version

Asana

`asana.com`

Asana is a project management and collaboration tool known for its user-friendly interface, task and project management capabilities, collaboration features, and integrations. It helps teams organize tasks, streamline communication, and automate workflows. While offering mobile accessibility and robust security, Asana's pricing is generally per user per month and can vary based on the feature set and user count. It's widely used for its customization, reporting, and ease of use, making it a valuable choice for improving team productivity and project management.

Asana is mostly focused on task-based project management. Features such as time tracking, and simple offer creation/billing, are missing.

They have a three-tier pricing model. Having a free version which supports simple use cases, it goes up to 25.- CHF for the Business version

MILKEE

`milkee.ch`

MILKEE is a young software made for independent or self-employed people. It supports you with billing, time tracking and accounting. It stands out for its extreme ease of use and fast setup. It lacks on collaboration with multiple people as it is only suited for self-employed people. MILKEE is great for accounting, offering and billing. However, it does not support you with organizing a given project and assigning tasks.

They have a two-tier pricing model. Starting off at 17.- CHF for a light version, it goes up to 25.- CHF for the plus version. These two tiers do not differ in functionality, instead they differ in gross income.

2.2.4 Market Potential Analysis

After already giving insights into the market potential, we want to get more detailed. First, we want to describe the potential of the market in our own words, then identify the key factors for a successful market entry and finally have an outlook on what the future could bring.

Market Potential

We see the market as one with considerable potential which is not fully saturated yet. The market already has a substantial size with loads of self-employed actors and startups. We do not see this trend reversing anytime soon but see it getting ever easier for startups and self-employed individuals to take a shot at trying to make it. We see this market as having potential to grow which means a growing demand for our solution for a number of reasons.

It has never been easier for individuals to land jobs than now. In the age of social media, for a skilled photographer, a well-managed Instagram profile can be all the marketing they need. For a medium-sized venture looking to shoot a marketing campaign about an event they're hosting, their best option may no longer be considering marketing agencies and going through their portfolio but checking the work of Photographers, Videographers and creative digital content creators via Instagram.

The sectors we identified worth targeting we also see as future-proof and bound to grow. We see those sectors as especially promising to pump out loads of new players whose needs we specifically can address to help them to a successful market entry. With the growing need for and impact of digital marketing, especially Photographers, Videographers and digital content creators find themselves in growing sectors. With individuals in positions of power in bigger corporations gaining awareness of that, we see the demand growing especially for individuals providing those services.

Market Entry Factors

With this analysis, we identified key factors for a successful market entry.

We need to get the attention of as many potential customers as possible. It is instrumental that the hurdle to using *Consilium* is as low as possible so as many potential customers as possible are willing to try out *Consilium*. An attractive look and feel will be instrumental for user retention as we need to subconsciously appeal need to potential new customers to give them a reason to want to use *Consilium*. This is why, a satisfying, minimalistic customer journey from first hearing about *Consilium* to signing up and quickly seeing first results is key. This will also considerably impact the potential pricing model, as this will always be a big hurdle, especially for ventures with limited resources to spare. The use of analytics tools to track where along the journey we lose potential customers will also be necessary, so we can directly target those areas.

We see the quality of our software as instrumental in gaining a foothold. Instrumental for the quality is a good look and feel and straightforward, uncomplicated processes without the need for repetitive steps for a user. The competition is too advanced to give immature software a chance. The quality of the functionality will be as or even more important than the scope. To effectively gain a foothold, the architecture is also detrimental, as we need to be extendable to grow with our clients.

A key element for success will additionally be the ease of use while also providing advanced features. The simplicity of the experience for customers with basic requirements must not be affected by more advanced features they do not need. This means a good structure will be instrumental.

Market Outlook

What will the future bring? Although artificial intelligence (AI) Tools are everywhere, they might not help you to plan your time and the time of projects. There is no doubt, that the market for project management software will eventually increase. Trends such as AI should be taken into account and should bring arguments on our side to improve *Consilium* continuously.

2.2.5 Conclusion

The market is already rich with a broad selection of different solutions, and it is not easy to find a place in this competitive sector. The competition offers very advanced accounting solutions and customer management and offers strong assistance for task management. Our strength should not be focused on these areas as we can not compete with each competitor in their best discipline.

All features that are being implemented in *Consilium* are in some way covered by at least one of the competitors. However, there is no such application as *Consilium* that combines all these features and additionally helps you get started with project and customer management. Most of the competing software is good at managing on a company or financial level, *Consilium*, however, should also support you on a project level. *Consilium's* opportunities lie in the easy way of use, fair pricing model, and compatibility for most use cases. In the future, we could additionally specialize *Consilium* for specific industries, such as Computer Science or Photography for example, where we see most opportunities, instead of trying to generalize it too much. Another possibility would be to offer multiple layouts with differing functionality within *Consilium* to achieve that level. Because not all functionality is required for each type of project the ability to hide those parts of the application, would prevent it from becoming overloaded.

We should focus on a broad range of features at the beginning, which can later be further developed to gain in depth. Our strategy should be to grow with our customers. With a growing customer base, we could start introducing community features, where our users could help and profit from each other.

2.3 Requirements

This chapter lists the functional, as well as the non-functional requirements. In addition, it details the process of how we generated those requirements. As we are limited to the scope of the Study Assignment-Thesis, there were plenty of useful requirements, which still had to be put out of scope.

2.3.1 Requirement Analysis Process

The goal of the requirement analysis is to identify the requirements needed for a successful market entry.

In a brainstorming session, the team already brought up some new requirements. In addition, the team collected further requirements from representatives of our target customer segment. The team handed the representatives of the customer segment a list of potential new features but also tasked them to bring up their own feature ideas.

Personas

To identify requirements but also later to help us with architectural, conceptual and design decisions we came up with detailed personas. We tried to think of an as heterogeneous user pool as possible. We used different areas of work, as well as different constellations to think of greatly varying personas and there for requirements.

Alexandra Alexandra is an artist who likes to express herself with a paintbrush in her hand and not with a mouse and keyboard. With her list of clients growing, she felt the need to professionalize the administrative part of her work and is looking for a tool to assist her. As she is not that well versed with computers she wants to have a single tool that can keep track of her clients and projects. Additionally, she needs to be able to create offers and track and bill her work. All she wants to put in her offers is the material needed for each job and the actual time she needs to finish the painting. She then wants to bill the material she needed for the job as well as the time she spent on the offer. She is looking for a tool which is easy to use and does not require a lot of overhead to set up.

Charles Charles is a landscape gardener and is proud of his work. He needs a tool to manage his clients and their projects. He does everything from identifying the preferences of his customers and working up potential garden plans to planting the actual plants and accessories. What Charles never got the hang of, is digital design. In this day and age, a lot of his customers ask for digital sketches of how the gardens would look once Charles finishes his work to help them decide. Luckily his niece Alexandra taught herself Photoshop and helps him out with digital sketches. Therefore, Charles wants to give Alexandra access to the management tool, so she can track her time, give him the sketches and create offers for her tasks. Additionally, he wants to give his customers access to those sketches directly in the tool but does not want them to be able to modify anything.

Daniela Daniela is a self-employed cyber-security consultant. Most of the time she offers out security reviews to software companies. She needs to be able to generate complex offers as they are usually not that straightforward. She needs to specify the deliverables, which she provides at the end of the review so the customers can not accuse her of not finishing her job. Additionally, she needs to be able to specify what she needs from her customers by which date, so she can meet her deadline. As most of her projects have a similar character she would like to be able to copy old offers or use templates, so she does not have to start from scratch each time.

Max Max is a gifted full-stack software developer. After jumping from software company to software company due to conflicts with his colleagues and bosses, Max realized, he works better alone and decided to start freelancing. Max wants a tool to keep track of his customers and his ongoing projects. In addition, he needs a tool to create offers for his customers as he grew tired of using Word for it.

Sergio Sergio primarily works as an accountant and has his own photography projects on the side. As he keeps getting more photography gigs, he needs to professionalize his management efforts and is looking for a suitable tool. For some jobs, he likes to involve his friend Lance who is a skilled drone pilot. As he does not understand much about Lance's craft and Lance always needs to rent props and equipment for their shoots he wants to give Lance access to the management tool so he can put the charges for all the equipment directly in the offers for the customers to see.

Lewis Lewis runs a small software company. He started on his own but as he kept getting more jobs he decided to expand and hire some developers. He now employs four developers. As he has not got enough funds to rent an office just yet, everybody strictly works from home with regular team dinners to keep in touch. Lewis needs his management tool to give him an overview of all ongoing projects to track the workload of his team. He is mainly focusing on customer acquisition and needs a tool to create offers for potential new customers.

From those personas, we identified different workflows we want to support with our tool.

2.3.2 Functional Requirements / User Stories

All the input we processed was then used to define the following functional requirements/user stories. For the user stories, we used different areas of work and different roles of customer to make them more tangible but the requirements are not exclusive to the described roles.

Offers – Customization

As a project leader with different types of customers and projects,

I want to be able to be flexible with my offers, being able to create offers with different parts, headings and lists

so they best fit the customer and project, as offers strongly depend on the customer and project in question.

Offers – Templates

As a freelancer with many similar projects,

I want to be able to create and use templates for offers

so I can save time.

Offers – Cloning

As a freelancer with many similar projects,

I want to be able to clone existing offers

so I do not have to build them from scratch each time.

Offers – Versioning

As a project leader in close contact with the customers,

I want to be able to view the previous versions of offers

so I can keep track of the history.

Offers – Binding times

As a project leader in close contact with the customers,

I want to be able to note binding times in my offers

so I do not run the risk of customers holding me to expired offers.

Offers – Obligations

As a project leader in close contact to the customers,

I want to be able to note obligations in my offers

so I can hold my customers accountable to those.

Collaboration – Access

As a backend developer working with a frontend developer on projects,

I would like to have access to the same customers and offers as my teammate

so that we can work on projects together.

Collaboration – Document storage

As a photographer, I want to share my work with customers,

I would like to give read-only access to the customers and have a place to put my work

so that the customers can see what I am working on.

Security – Login

As a freelancer with sensitive projects,

I would like to restrict my competitors from accessing my work

so my intellectual property is kept safe.

Billing – Time log

As a freelancing software developer working on client projects,

I would like to be able to log the hours I spent working on the projects on specific activities

so I can invoice my efforts later.

Billing – Invoice

As a *Consilium* user working on projects,

I would like to bill the work I logged

so I can directly send out an invoice.

2.3.3 Non-Functional Requirements

In this part, we talk about Non-Functional Requirements (NFRs). These are important because they tell us how our software needs to work, not just what it should do. We have made sure that these requirements match up with the different parts of our project. This helps us make a system that does its job well and is also strong, fast, and easy for people to use. We have thought about what our users and the people we work with need, and we've used that to make a set of rules that help our system be its best.

Functionality – Security – Confidentiality

A user is never able to access any data he is not authorized for. A user can only view his own data or data from workspaces he has been invited to.

Functionality – Security – Integrity

It is never possible for a user to put the data in an inconsistent or incorrect state.

Usability – Consistency

The error messages across the whole system are consistent.

Usability – Training time

A user needs only an introduction of 30 minutes to be able to fully use the tool. Without any training, the user should be able to create his first offer within 10 minutes of being logged in. With a further 10 minutes, a user is able to create activities, log work, and create an invoice, given he knows his bank details.

Usability – Usability standards

A password policy will be put in place with guidelines on length and complexity.

Reliability – Availability

The system should be available 99% of the time.

Simplicity

The tool should be easy to use and the workflows should not be harder to achieve than necessary. The following criteria were defined to verify this requirement:

- Given the user is on the dashboard and knows which project an offer belongs to, he can download the offer with less than five clicks.
- Given the user has set up his time tracking activities and categories and has logged work, he can determine easily, which categories have exceeded their estimates.
- When the user is in a project detail screen, and there is an offer with multiple versions, the user is able to access and view this version with less than 5 clicks.

- When the user has the necessary data available to him, he is able to fully set up the workspace settings in less than 3 minutes.
- Starting from the project detail view, given the user has logged work and not invoiced it yet and has set up the settings, the user is able to create and download an invoice with less than 7 clicks.

Performance – Response time

No single request should wait longer than two seconds for a response.

Supportability – Localization

The system should be available in English and German.

Compatibility

The software should be compatible with the latest versions of major web browsers, including Chrome, Firefox, Safari, and Edge, ensuring accessibility for a broad range of users.

Maintainability - Code Quality

The system's code should adhere to best practices for readability and maintainability, such as using clear naming conventions and having a modular structure. Even a Junior Software Engineer should be able to understand the code and continue implementing features.

Conclusion

To wrap up, the Non-Functional Requirements we've listed are key to making sure our software works really well. They are not just a list of things to check off. They guide us in making high-quality software. We've linked these requirements to our project goals and the stories of our users. This helps us make software that does what it needs to do and does it in a way that's smooth, safe, and efficient. We will keep looking at these requirements as we work on our project. We want to make sure our software stays up-to-date and keeps meeting the needs of our users and our team. Our goal is to make a product that not only meets expectations but goes beyond them.

Chapter 3

Solution Design

This chapter aims to highlight the concepts that we created to solve the problem we were confronted with. This chapter is on a different level compared to the chapter Chapter 4 which will highlight key implementation details. The goal of this chapter is also to provide insights into the architecture and how all software components interact with each other, as well as the design, but also showcase the most important concepts developed in this thesis.

3.1 Architecture and Design overview

This section shows the general architectural composition of our application and highlights the decision and thought process which led to this general design. More specific and in-depth decisions and designs can be found in Section 3.2.

3.1.1 Architecture Overview

To visualize our software architecture, we used the C4 model. However, due to the size of our software and the detail we want to provide in this thesis, we decided to only cover Context, Components and Containers.

Level 1: Context

The following Figure 3.1 gives an overview of our system landscape and showcases the interaction with *Consilium*, as well as its interaction with all external systems.

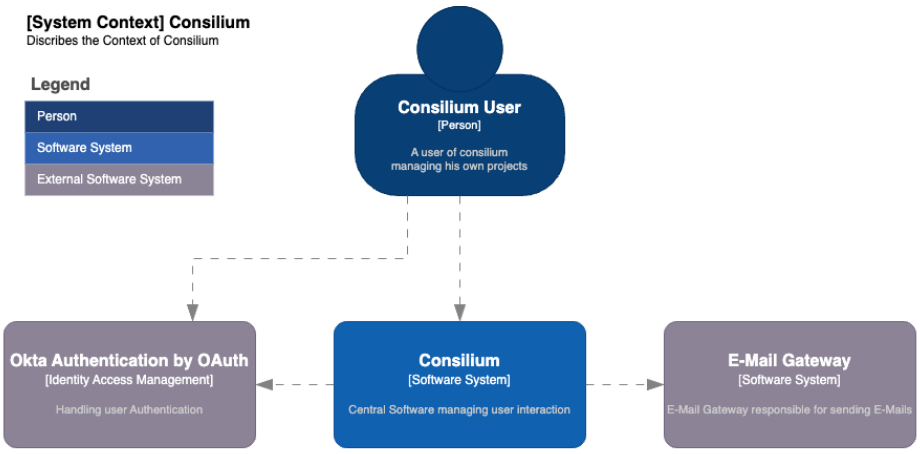

Figure 3.1: C4 Context Diagram

A *Consilium User* interacts with the *Consilium* software itself, as well as the Okta Authentication service, which is powered by OAuth. The *Consilium* software also interacts with a configured E-Mail gateway.

Level 2: Container

The container view in Figure 3.2 depicts a closer look into the system. Each container represents a separate application within the system.

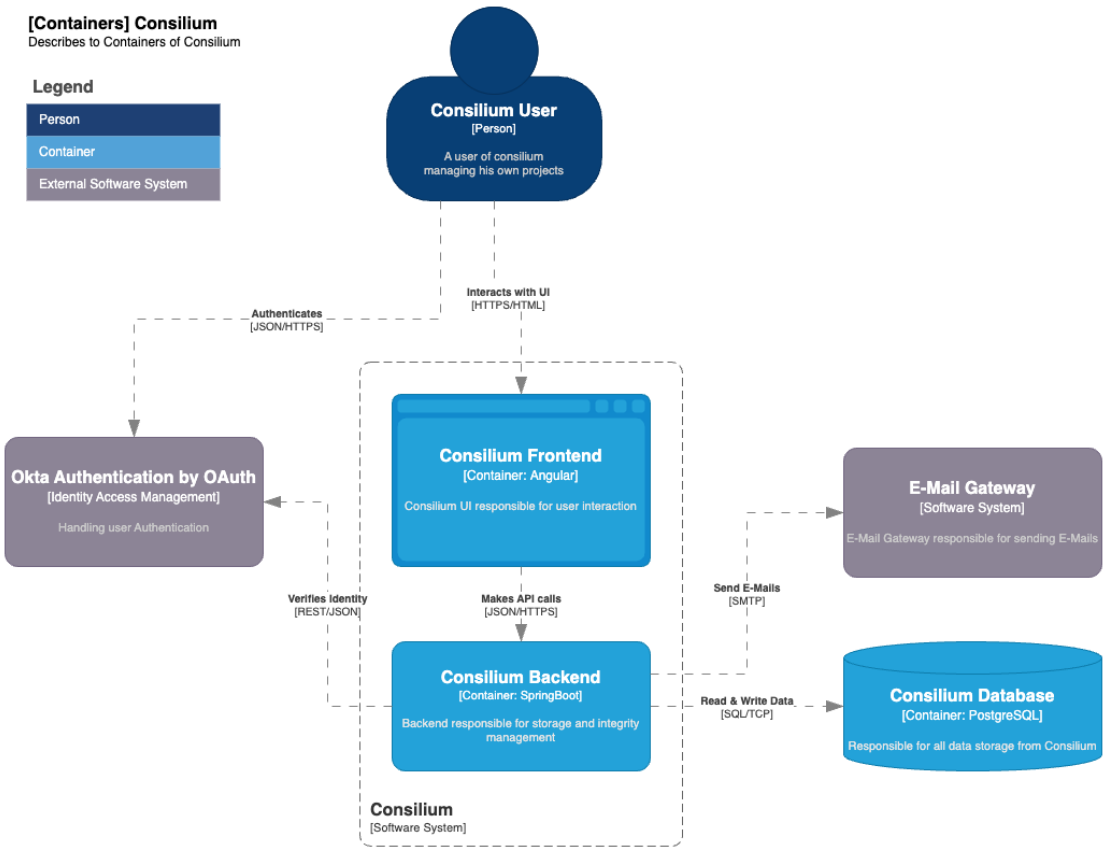

Figure 3.2: C4 Container Diagram

Consilium is split into *Consilium Frontend*, *Consilium Backend* and *Consilium Database*.

We wrote the *Consilium Frontend* in Angular/Typescript. We decided to use a Web Application instead of a desktop application for portability. With our Frontend also optimized for mobile clients, potential users can access the application from every platform. This decision was made as the team largely has experience with this technology. The Frontend communicates with the Backend via REST. We used the REST maturity level 2 to communicate with each other. Due to the fact, that we are in control of the Back- and Frontend, we found no need to satisfy all 3 levels of the *Richardson Maturity Model*, [1].

The Backend is a Spring Boot Java Application. We use this technology as it is optimal for the scope of this project and provides a lot of functionality. The architecture and technologies of the Backend are further highlighted in Figure 3.3. The Backend communicates with the database via SQL/TCP.

The database of choice is PostgreSQL, as the team has experience with it and PostgreSQL provides short start-up time which our testing benefits from. All of those decisions have been made in the SE Project module and are only explained in the document.

Level 3: Components

The following Figure 3.3 showcases a more detailed look into the architecture inside our *Consilium Frontend* and *Consilium Backend*.

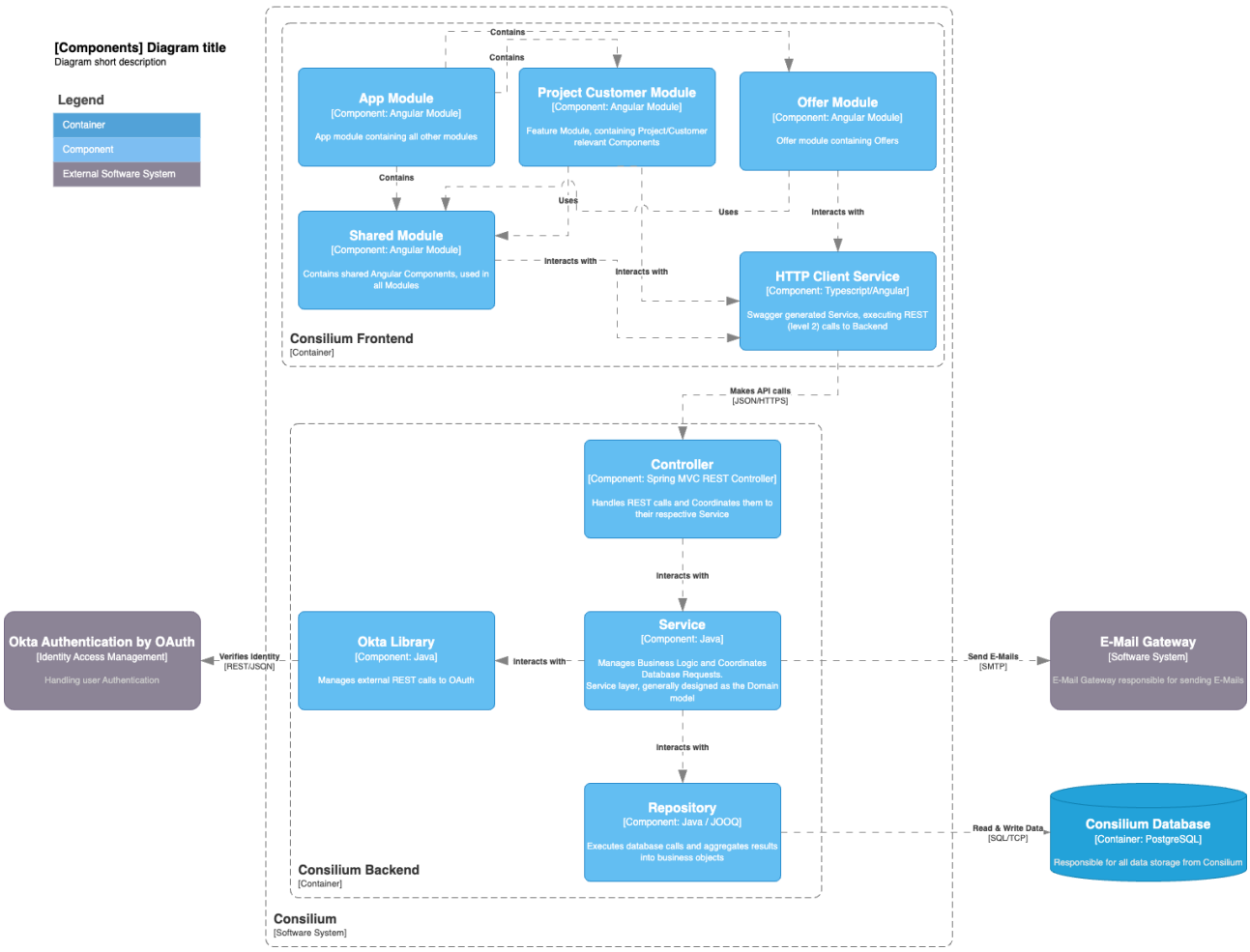

Figure 3.3: C4 Component Diagram

Backend Component The Backend is designed in a standard 3-level SpringBoot architecture. We focus on the insides of the backend and especially their respective data flow. The controllers provide the API. They take in the DTOs (Data Transfer Objects) from the calls and return DTOs to the Frontend. The controllers map the DTOs into business objects and pass them to the services, which handle all the business logic. The service layer is split up into several different domains. Due to the size, this could not be visualized in Figure 3.3. However, we provided a very simplified domain model in Figure 3.4 which describes, how our service layer is divided. The Repository layer interacts with our database and aggregates data which is then being provided to the service layer. JOOQ was chosen as it is easy to read and helps us write safe SQL. It is also very handy when it comes to adjustments or changes. JOOQ generates database classes during compile-time, which allows us to write typesafe SQL. To ensure integrity in our production environments we use Flyway, which is a database migration tool. This plays hand-in-hand with JOOQ to generate the Java database model.

Frontend Component The Frontend consists of several components, which interact with the *HTTP Client Service* which is being generated by Swagger. The Frontend is structured with angular best practices in mind taught at OST in the WE3 module. For styling Angular Material is used. We split the main features into separate modules to keep the code clean and maintainable. The following modules are used:

App Main module which contains all other modules. It defines the navigation. Also main routing module is located here.

Shared Contains all shared components, services, models and pipes, which are used in all other feature modules like the back button.

Project Customer *feature module* Contains all components used for the project and customer management. Has its own routing module.

Offer *feature module* Contains all components used for offer creation and building block management. Has its own routing module.

HTTP Client Service *auto-generated* by Swagger. It can be used to access the Backend.

3.1.2 Architectural Considerations

In this chapter, we go into further detail about the considerations which led to us choosing this architecture. The physical decomposition will not be addressed further in this section as it is already addressed extensively.

Logical decomposition

Backend With our backend architecture, we focused on clearly defined layers, to keep the code maintainable and extendable for potential new developers to get up to speed quickly.

This architecture is reflected in the package structure of our source code by being split into packages "controller", "service" and "repository". The controllers lay in the package "ch.ost.consilium.controller" with the DTOs laying in the sub package "dto" and exceptions in the sub package "exceptions". The services lay in the "ch.ost.consilium.service" package with the used Business Objects being put in the sub package "data". The repositories lay in the package "ch.ost.consilium.repository". All these three layers are then divided into different domains which can be seen in Figure 3.4. The previous idea of having separate packages for each domain has been discarded as the team decided together the split is already clear and clean enough. However, if the project would be growing even further, we would recommend to reconsider this decision.

In the service and repository layers, we decided to use interfaces for each service and interface to keep the components decoupled from each other. This aggressive decoupling, allows us to work efficiently with each other since an interface is enough to write code from both sides. The controller only depends on the interface of the service and the service depends on the interface of the repository.

Frontend In the Frontend, we aimed to achieve a similar logical decomposition that ensures maintainability, extendability, and quick onboarding for potential new developers. To achieve this, we divided our Frontend code into separate modules for each main feature. This allows us to extend the application with new functionality independent of others easily.

The modules are responsible for their own specific features, which makes the code more modular and easier to manage. Furthermore, this approach allows multiple developers to work on different modules concurrently, without intensive conflict, which speeds up the development process.

Furthermore, we implemented a modular architecture that allows us to reuse components across different modules with the shared module. This approach saves development time and ensures consistency across different features.

Cross-Cutting Concerns

Backend This concern has been at the heart of our architectural considerations and has led to us using this architecture. The logic is kept out of the controllers consistently and is kept in the services, which are cleanly split into the domains. The persistence layer with our repositories focuses only on providing the services with the needed data. This leads to each component having a clearly defined area of responsibility and small, clear functions. One of the main focuses was to decouple the logic and methods into small clearly defined methods and not huge, complicated ones.

Frontend In the Frontend, one of the cross-cutting concerns we addressed was separating the code, style, and theme for each component. Angular, the framework we used for our Frontend, does this very well with its component-based architecture. Each component in our application encapsulates its own HTML template, CSS styling, and TypeScript code, making it easier to manage and modify each component without affecting the others. Additionally, Angular's use of dependency injection allows us to further decouple our components and make them more modular and reusable. This has helped us to keep our codebase organized and maintainable and has made it easier for us to add new features or modify existing ones without introducing unexpected bugs.

Extendability

Backend The decision to use interfaces and decouple the components from each other has also partly been to allow further extensions in the future. The implementation of those interfaces is interchangeable and new functionality can always be added.

Frontend In the Frontend, our decision to use separate modules for each main feature allows us to easily extend the application with new functionality independent of others. This approach ensures that adding new functionality to one feature does not impact the functionality of other features.

Scalability

As our system is intended for multiple users, see Section 3.2.2, it is a central aspect of our application to be scalable. With this in mind, we developed our Backend highly performing and asynchronous. By enabling the possibility to go into the cloud, our application can run more than once, which makes our application extremely scalable.

Hosting Distribution

As we are carrying three decoupled components, it is technically possible to host the database, the Backend and the Frontend on three completely different hosts. However, on the current production setup, the Backend is serving the Frontend files and the database is on the same host inside a docker container.

Consistency

We want to explain our consistency concerns using the ACID (Atomicity, Consistency, Isolation, Durability) theorem. With our SpringBoot and JOOQ setup, each request is by default managed inside a transaction, managed by SpringBoot. This feature guarantees the **A**tomicity property. Having as

restricted Database Field rules as possible, we want to ensure **C**onsistency inside our Database. However, there are some situations inside the application, where inconsistent states could be achieved. This would be something which must be addressed before going into production. **I**solation and **D**urability have not explicitly been addressed in our work.

Observability

If going into production it is indispensable to have a well-thought-through logging and monitoring concept. However, in the scope of our thesis, we have unfortunately not been able to tackle this concern. We identified two key concerns, which are currently missing.

1. `Logging and Storing Logs`: Although logging is set up in our application, we must store the logs for later investigation. Additionally, we should integrate default loggings for each request.
2. `Monitoring and Health Checks`: Spring has a health check feature (covered in the actuator dependency), which enables checking for the system state. Although this feature is currently in use to verify the database connection, these health checks should be extended to check for all our relevant interfaces. As a next step, automated reporting and monitoring would be required to ensure a low incident response time.

3.1.3 Domain Overview

To provide an overview of the problem domain we are situated in, we created a highly simplified domain model. This model is supposed to give a broad overview of all our problems and addresses their relation to each other. More detailed models will be explained and shown in Section 3.2.

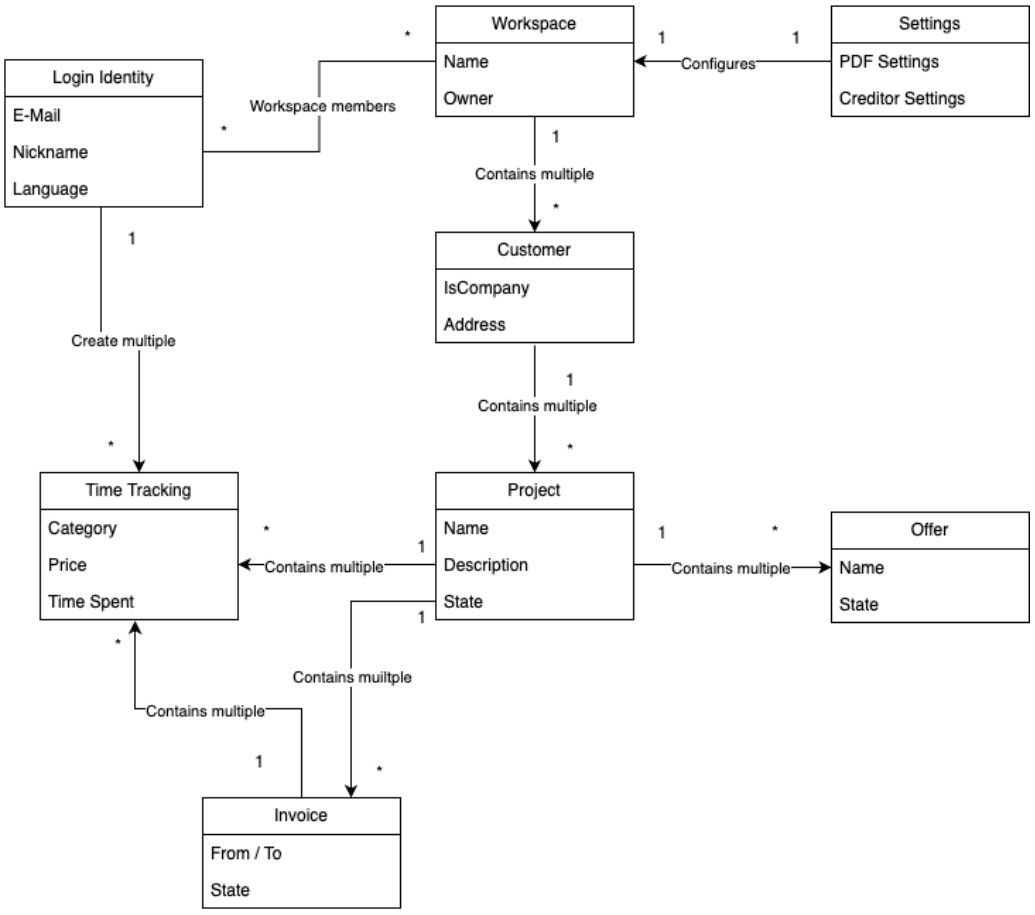

Figure 3.4: Simplified Domain Model

1. `Workspace`: A *Login Identity* can have multiple *Workspaces* and *Workspaces* can have multiple *Login Identities*. A *Workspace* contains multiple *Customers*.
2. `Settings`: A *Workspace* has *Settings* which configures *Workspace* specific properties.
3. `Customer`: A *Customer* lives in a *Workspace* and contains multiple *Projects*.
4. `Project`: A *Project* lives in a *Customer* and contains *Offers* and *Time Trackings*.
5. `Offer`: An *Offer* lives in a Project and manages multiple versions of *Offers*.
6. `Time Tracking`: *Time Trackings* live in *Projects* and manage categories, activities and entries.
7. `Invoice`: An *Invoice* lives in a *Project* and contains multiple *Time Tracking* entries.
8. `Login Identity`: A *Login Identity* is the Identity of a User and his login information.

3.1.4 Database Model

As we demonstrated the prototype database model in Section 2.1.2, we want to compare it to the final database model.

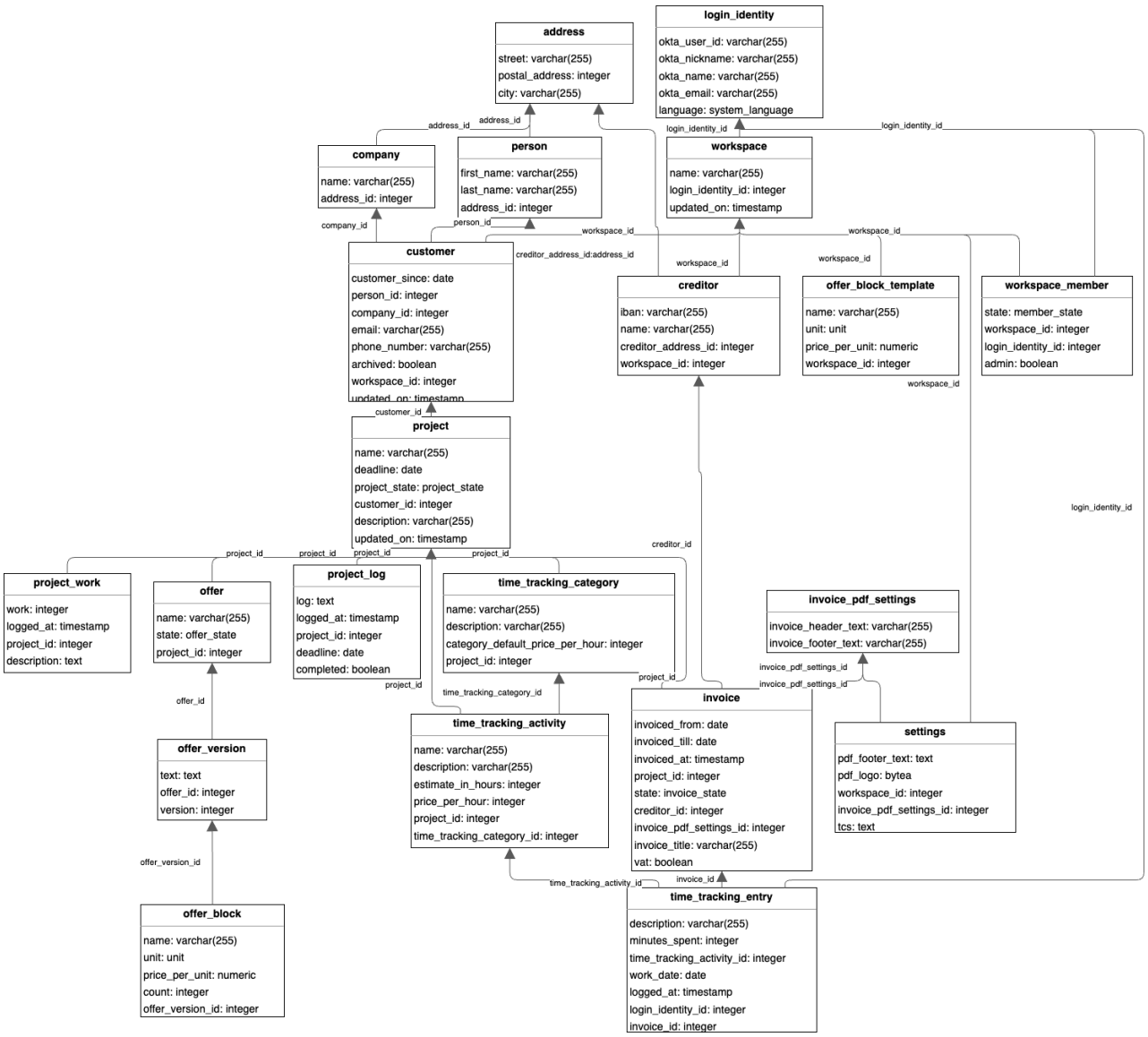

Figure 3.5: Database Model

As we can see, besides the tables *project*, *customer*, *company* and *person* everything has changed. To understand how these changes arose, we must take a look into Section 3.2 where we explain the several concepts, which we worked out.

3.2 Technical Concepts

This chapter showcases, how conceptual and technical challenges were tackled.

3.2.1 Consilium SaaS

Consilium has by now just been running on virtual servers. As a next step, *Consilium* should be able to run on one of the most used cloud solutions nowadays. As we decided to go the SaaS (Software as a Service) path with *Consilium*, we want to give insights into what we understand with SaaS and what Consilium does to be a SaaS application. We had the option to go either with an on-premise solution, where every company can install *Consilium* on their server and also maintain it by themselves. And the other option however was, that we offer *Consilium* as a Service, or simply said, we offer a Web-Application, where all users can register themselves. Due to our target group, it was an obvious step to go the SaaS way. Freelancers or micro-companies do not want to deploy and manage software by themselves. To realize this intention, there was an analysis of the three most common cloud providers.

- Amazon Web Services (AWS) - Microsoft Azure - Google Cloud

All of these cloud providers are compatible with our software since we compile our complete software into a single JAR file. Based on the complexity of deployment and pricing, we decided to go with AWS. Although AWS is our favourite to go with, we have decided against using it during development, as this would have produced a significant bill. However, we proved our intention by deploying *Consilium* in a proof of concept (POC) manner onto AWS and have been running successfully for one week. To deploy the JAR application, we use *Beanstalk by AWS*. Besides the running JAR application, we do additionally need a database, which can be used, using *RDS by AWS*

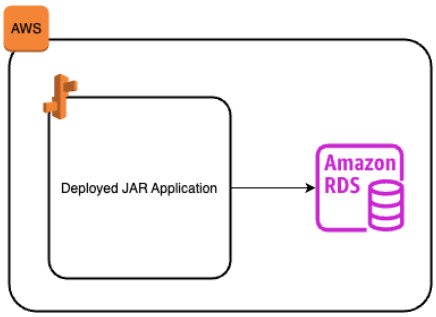

Figure 3.6: AWS System Diagram

First of all, everything is deployed inside of AWS. AWS in that case acts for *Consilium* as an IaaS (Infrastructure as a Service).

Next, the SprintBoot Application is deployed inside of a *Beanstalk* Container. This Container could be duplicated and load-balanced if needed. This is where our SaaS (Software as a Service) is running.

Lastly, we use Amazon RDS (Relational Database Service) to run our *PostgreSQL* application.

For an extensive guide on how to configure AWS, we have attached our README documentation in Appendix E.

3.2.2 Multi-user Concept

A key challenge for this project is to allow multiple users to work with the tool and work in collaboration. To enable this functionality, authentication, authorization as well as a thought-through user concept need to be established. This concept aims at fulfilling the functional requirement Collaboration - Access.

Collaboration Concept

First, a concept was needed on how we would support collaboration and allow a team of users to work together. As our customer segment can be diverse, our goal was a solution supporting different workflows. The personas identified for the requirements (Personas) helped identify the different workflows we want to support with our tool. We are going over our chosen solution and also brush over the alternate, discarded variants.

Chosen Concept: Workspaces After looking at different ideas, we chose the one that supports the right workflows for our customers, is easiest to use, and fits best with our current system. The last point was critical as implementing a complex solution with a big impact on the current architecture would have taken up a lot of resources which we would rather invest in more features.

In the end, the concept, which best met all the requirements, was workspaces. Workspaces is a concept newly introduced to Consilium, wrapping everything. Everything from customers to their projects and their offers is contained in a workspace. Per default, a new user has a personal workspace which he can use right away without having to set up everything. Workspaces can also be collaborative, meaning multiple users can access the same workspace. The owner of a workspace can invite collaborators but also viewers who will be granted limited access.

Figure 3.7 shows the domain model which we came up with to realize the workspace concept.

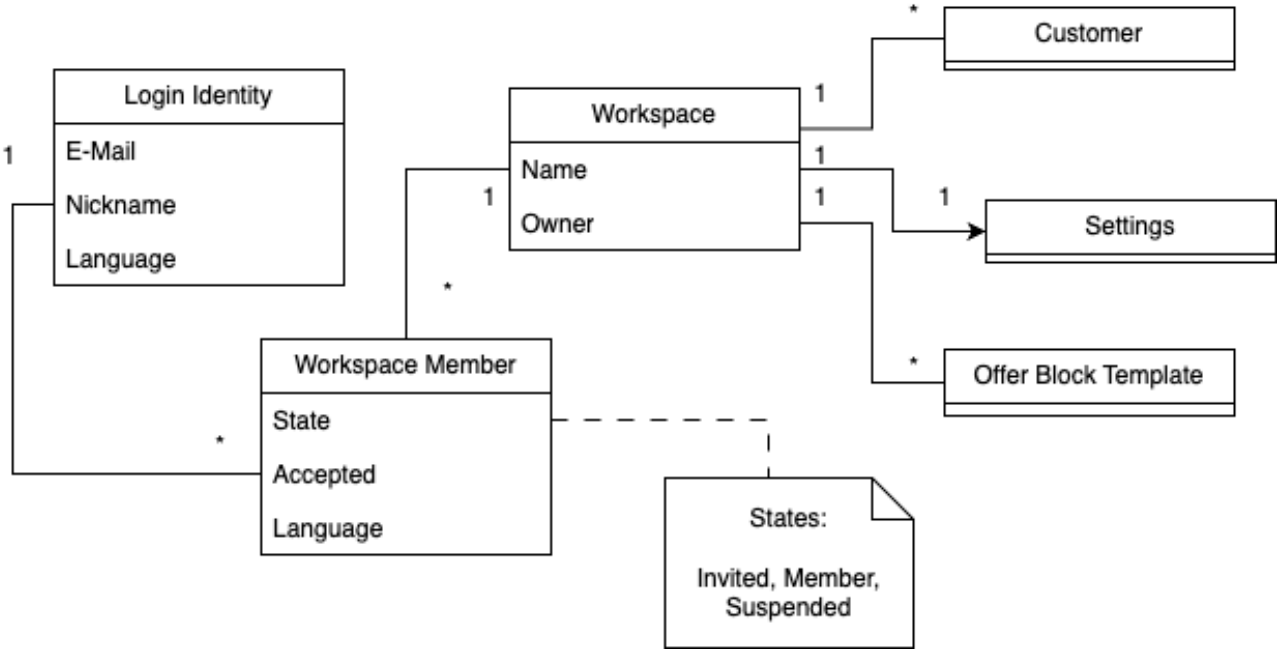

Figure 3.7: Domain Model for Workspaces

This solution was chosen as it allows for collaboration without needing a lot of management overhead to get it up and running. Another key factor is, that users who will never use the tool to collaborate are not faced with more effort for something they do not need. With the given structure of the application, this concept was also not overly complex to introduce compared to others. Another big benefit of this concept is its potential for expansion. In the first instance, a simple version with a minimal role and authorization concept can be implemented, which can then be extended with growing complexity.

Alternate variants Other variants have been considered but were not chosen in the end.

Different instances One solution would have been to change little to nothing on the application but require customers to host their own instances with their own login process.

This variant was discarded mainly as it would be a considerable hurdle for potential new customers to get hands-on with the tool. Additionally, there would be no possibility to give limited access to end customers. We also considered it to be a step in the completely wrong direction as we rather go down the SaaS route with our product. The only benefit of this solution was, that nothing would have needed to be changed in the architecture.

Shareable Another variant was to introduce the possibility of sharing something on different levels. For example, the owner of a project could share access with another user via a link.

This variant was discarded due to its inferior simplicity. With this variant, we saw the risk of it becoming harder for customers to keep track of which data is shared with which users. Another key reason was its implementation complexity.

3.2.3 Authentication

With the move into the cloud and the introduction of a multi-user concept, a solution for the access-management is required. This concept aims at fulfilling the functional requirement Security - Login. To aid us in finding a solution, we defined the key forces we look for in a potential solution:

Simple login and sign-up process To keep the retention rate of customers entering *Consilium* as high as possible, it is decisive to steer clear of complicated sign-up and login processes.

Decoupled A key force for us was to keep the login process as decoupled from the application logic as possible, to allow us to keep the possibility to change this component in the future.

Low-to-medium effort Even though we are interested and well-versed in writing access-management software, we decided to limit ourselves to resource-friendly solutions, as we wanted to keep the focus of this thesis on other aspects.

Security Although we opted for a simple login process, we did not want to completely disregard security.

Based on those forces, we identified and evaluated the following potential solutions:

Do it yourself

One potential solution was, to implement the whole login process by ourselves. By building the complete functionality ourselves, we would perfectly have the simplicity of the solution under our control. By implementing a decoupled, standardized process, for example, by propagating a signed JWT (JSON Web Token) we could keep the solution decoupled from the business logic. With our expertise in this area, we would have been able to implement a secure solution. What drove us away from this option in the end was however, that it would have required considerable effort and would have changed the character of this thesis. Although the implementation of a custom authentication solution is a very interesting challenge, we decided against it in favour of other interesting challenges. Not only the time itself to realize the authentication would have taken far more time but we would have been required to consider an external review of our authentication, to be sure that the implemented solution would be fully secure.

All of those factors have made us decide against this solution.

Airlock Identity Access Management

Another option we evaluated was to use Airlock IAM (https://www.airlock.com/en/secure-access-hub/components/iam). While Airlock IAM offers way more than simple registration and login flows, they would still be very much possible. A complete decoupling would still be possible, as after the login process an Identity could be propagated to our backend through a JWT. With their support of various security factors and measures Airlock IAM would have gotten the best marks for security, for the considered possibilities. The setup of the Airlock IAM would have required more hosting efforts as their SaaS Solution is not ready yet and it would have needed to be on-premise. This in the end led us to not pursue this option, as an on-premise solution would have gone against the spirit of this thesis.

Auth0 by Okta (chosen variant)

In the end, we decided to use Auth0 by Okta (https://auth0.com/). Auth0 supports features which are perfect for our use case, out of the box. Especially the ability to use social logins, is a perfect fit for our requirements and philosophy, as this makes it extremely simple to log into *Consilium*. As we did not want to make the login a key part of this thesis it was a big plus that we found reports and many tutorials, which showed us how easy it was to integrate Auth0 into setups like ours. With Auth0 it is also possible to provide a JWT after login and with great integration into spring, we do not even need to verify the signature, the issuer and the audience by ourselves. This makes it both, decoupled and easy to integrate while maintaining a sufficient level of security.

3.2.4 Authorization

This section explains the authorization concept we established for *Consilium*. This section is tightly tied to sections Section 3.2.2 and Section 3.2.3 as the concept developed for authorization needs to be compatible with those two concepts. The authorization concept aims at fulfilling the non-functional requirement Functionality - Security - Confidentiality, by preventing unauthorized access.

We identified the following criteria for an authorization concept:

Simple setup and assignment It should be easy for an administrator to set up the authorization for all his members.

Clear overview The administrator should easily see which member has which access rights, so there is not any accidental leak.

Fine-grained The authorization concept should also allow for fine-grained customization, in case an administrator wants to give his members very specific access rights.

Extendable The authorization concept should not be implemented in a way which breaks with future development.

As already mentioned in the section Section 3.2.3, the emphasis of this thesis is not on security, so we decided to go for a rather simple concept. However, in this thesis, there is an emphasis on well-designed, extendable software, so we designed a proper concept and designed the software in a way, that the proper concept could be put into place at a later stage.

Implemented authorization concept

We have adopted a straightforward authorization approach. The topmost access level in a workspace is designated as the owner. This designation is a critical component designed to ensure that only the owner has the authority to elevate or reduce the roles of team members to administrators. This precaution is necessary to prevent scenarios where a newly promoted user could potentially downgrade the workspace owner, a situation we deemed undesirable.

Following this, the second-highest level of access is the administrator. An administrator has comprehensive control over the workspace. This role is distinct from a basic user primarily in its ability to alter workspace settings. These settings adjustments can range from modifying PDF template features like the logo, inviting new members, changing terms and conditions, to tweaking creditor information.

Our objective was to create a user role management system that is both elementary and efficient. Consequently, outside the settings of a workspace, the application does not differentiate among users. Our design also emphasizes ease of adaptability. With minimal modifications – just a few lines of code – we can alter access rights to all features of *Consilium*. This flexibility even extends to a potential feature that would allow an owner to customize permissions within their own workspace. In Section 4.1.2 we have a detailed insight, into how we realized the authorization concept.

Advanced concept

The solution which we came up with, which would be the easiest for an administrator to manage, while also allowing the access rights to be fine-grained, would be based on roles. For each domain object in our software, there would be the possibility to configure read, write and delete access.

From there, the administrator could define roles, which translate into access definitions to those domain objects.

The administrator would then have the possibility to either assign the user roles or give him each access right separately.

To give the administrator a clear overview, for each member the detailed access table could be displayed.

Roles could then potentially be grouped into groups which members could be assigned to, inheriting those roles. However, this feature would be more suitable for a tool targeting big corporations.

3.2.5 Time-Tracking and Invoicing

To make *Consilium* become an end-to-end tool, being able to create invoices was a crucial requirement. The basis for the invoicing is time-tracking, which means while designing the time-tracking concept, we needed to keep the impact on the invoicing in mind. With this concept, we aim to cover the functional requirements Billing - Time log, as well as Billing - Invoice. The goal was to create a concept which allows the time-tracking and invoicing process as seamless as possible while also maintaining much-needed flexibility.

Time-Tracking Concept

While the invoicing was always considered while designing the time-tracking feature, there are other key forces which played into this concept:

Ease of Use The creation of time-tracking activities is often tiring mundane work. We aim to reduce the overhead here as much as possible.

Flexibility The time-tracking should be flexible. A lot of projects contain different tasks, done by different members at varying rates.

Monitorability Time-tracking should be monitorable and transparent. It is instrumental, that a project manager can keep track of the progress and the time spent.

Those forces were addressed through a combination of multiple components.

Time-Tracking Activites Activities are elements which can be created and then used to log work on. Activities are made up of a name, description, an hourly rate and an estimate. The estimate is ground for monitoring features, as it allows for easy tracking of the work spent on activities compared to the estimate.

This allows for complete flexibility as unlimited activities can be created, each with a different hourly rate.

Time-Tracking Categories We defined categories as a container for activities to improve some administrative processes. Categories are entirely optional, but they allow multiple activities to be grouped into categories and allow for a default hourly rate to be defined.

This further improves the monitorability of the work invested, as it makes it easier to identify in which are the estimates have been exceeded.

Time-Tracking Import To make time-tracking as easy as possible, we came up with the concept, of enabling users to import the time-tracking activities from the offers. We identified, that there is usually quite a big overlap between the positions offered and the activities. By allowing users to import the time-tracking activities from offers while also allowing the users to create their own activities, we managed to shave off a lot of overhead while not limiting the user by still enabling custom activities.

However, due to our limited time and some technical concerns, we were required to postpone the automatic integration from the offers.

Time-Tracking Entries Time-tracking activities can then be used to log work on. Those entities are called time-tracking entries. Those entries are instrumental for monitoring features, as the effective effort is calculated through these entries. It also shows which member logged how much work, as each entry is tied to the user who logged it.

To put all those entities into relation, we want to demonstrate their relation to each other using the following figure.

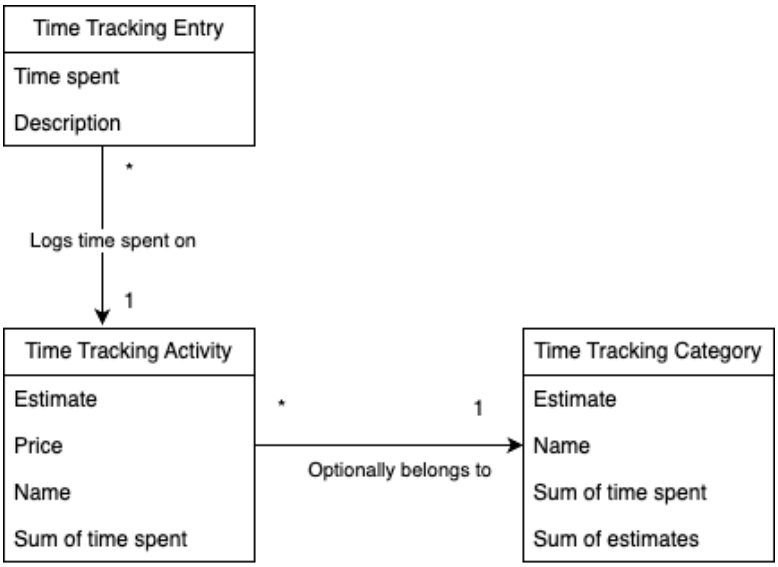

Figure 3.8: Time Tracking Domain Model

Invoicing Concept

The invoices are based on the time-tracking entries. We identified the following driving forces we want to address before coming up with the concept for invoices:

Customizability The invoices should be customizable as it may be necessary to put additional information on the invoices.

We tackled this issue, by allowing the user to set a custom header and footer text appearing before and after the invoice positions on the document.

Ease of Use We want to make the invoice process as easy as possible. The invoice process should be transparent yet as uncomplicated as possible.

This is addressed by basing the invoice around an invoice period. The user can select the invoice period, and the time invested is calculated and summarized. For the process to be transparent, the user is given a preview when creating an invoice.

Furthermore, users can define default header and footer texts, which will be suggested for each invoice and can be adapted.

Professionality Invoices need to come across as professional. A user should be comfortable sending out the invoices to their customers.

To improve the look and feel of invoices, we want to support QR-Bills which allow end customers to just scan the invoice with their e-banking app and pay **it**. Section 3.2.6 shows, how the integration of the QR-Bills was designed.

Additionally, we want to allow for custom branding, by reusing the branding ability from the offer creation.

Clarity Invoices need to be as clear as possible. The user creating the invoice as well as the customer paying the bill need to see what is on the invoice, quickly.

Here, the previously defined elements from the time-tracking concept come into play. Categories are used to summarize multiple activities into an area of work. It makes it easy to identify, which sector took how much work and cost how much. To keep transparency, the activities are also put on the invoice as sub-items of the categories. This allows us to model the scenario when activities have different hourly rates.

3.2.6 Invoice Swiss QR Code

We had many possibilities to create the QR Code for the invoices.

Firstly, we could have created the QR Codes ourselves, as the specification of the QR Code is public. Secondly, there were various APIs we could have integrated. After contacting various providers we were granted API keys. The third possibility was to integrate libraries, which can create those QR Codes for us.

The only restriction we had was, that it had to be free. With each option, we found a provider that would have allowed us to use their services or code. In the end, we decided to go for the library option. This decision was taken, as writing the code ourselves would have been more effort without a significant improvement and we would rather spend those resources elsewhere. On top of that, using the library provides more stability than using an API. Without having the dependency on an external

API, potential downtimes of the API do not affect the availability of functionality within *Consilium*. We chose the solution from https://www.qr-invoice.ch/ in the end. While the free tier only offers restricted commercial use, a premium tier could always be bought at a later stage for a one-time fee of 900.- or the functionality could be built within *Consilium*.

3.2.7 Offer Versioning

In the following concept, we will introduce you to our offer mutability problem and show our way from conception to implementation. This concept addresses the functional requirement Offers - Versioning.

Introduction

In our current system, users are allowed to modify offers at any time. While this provides flexibility, it lacks accountability and clarity in tracking changes. Additionally, it is challenging to ascertain which version of the offer was accepted by the customer. To address these issues, we propose implementing offer versioning.

Problem Statement

The existing system's primary drawback is the absence of a change history, making it difficult to track modifications. Furthermore, when rates or other details are altered in an offer, all historical offers are affected, leading to discrepancies in accounting and contract management.

Proposed Solutions

We considered two main solutions for offer versioning:

Solution 1: Saving a PDF for Each Version
Advantages:

- Simplicity in implementation.
- Requires less storage space.

Disadvantages:

- Limited functionality as old versions cannot be reused or modified.
- Lack of dynamic interaction with the offer data.

Solution 2: Copying Building Blocks for Each Version
Methodology:

- Each version is immutable.
- Building blocks are duplicated for each offer version.
- Blocks are editable within an offer.
- If offer building blocks are being adjusted, this does not affect existing offers.

Advantages:

- Consistency in rates and information across different offer versions.
- Enhanced flexibility and adaptability in offer management.

Disadvantages:

- Increased storage space requirements due to duplication of data.

Chosen Solution

After careful consideration, we decided to proceed with the second solution, copying building blocks for each version. This approach ensures that each time an offer is modified, a new, immutable version is created. This method enables accurate tracking of changes and preserves the integrity of historical data.

Implementation The implementation involves:

- Copying all relevant information at the database level into a new offer whenever modifications are made.
- Ensuring that changes in rates or other details in the system do not affect old offers.
- Maintaining the original offer's terms, even if future changes are made to the building blocks.

To visualize the chosen concept we designed a model, which explains the concept in a diagram.

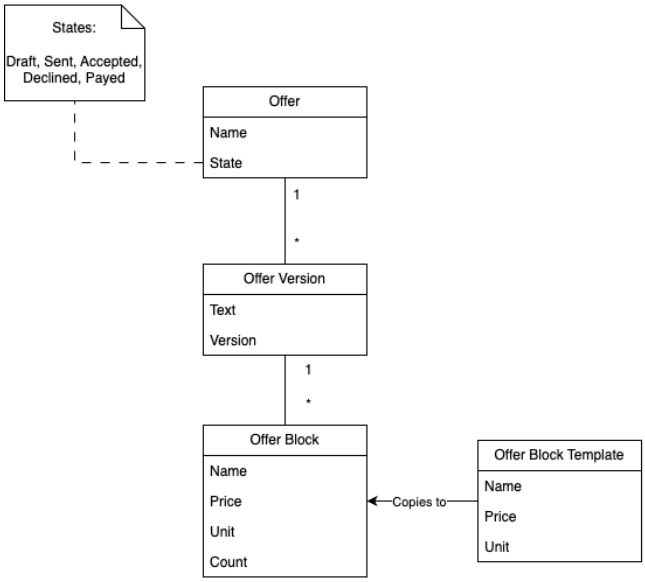

Figure 3.9: Domain Model Offer Versioning

Conclusion

By adopting the second solution, we can effectively address the issues of accountability and historical accuracy in offer management. This solution not only provides a robust framework for tracking changes but also safeguards against unintended alterations in historical offers, thereby enhancing the reliability and integrity of our accounting and contractual processes.

Chapter 4

Implementation

4.1 Implementation

This Section highlights the most important and interesting implementation tasks.

4.1.1 Integration of Rich Text in Project Offers

This section provides a detailed explanation of the technical implementation of adding rich-text capabilities to project offers within our building block system. Our objective was to enhance the flexibility and aesthetic appeal of project offers by enabling the inclusion of formatted text.

Implementation Details

Initial Setup The project offers are generated using a building block mechanism, where blocks of tasks, such as *"Photographer, 1h, $300"*, can be added. This structure forms the backbone of our offer generation system.

Incorporating Rich Text To introduce rich text functionality, we utilized **Quill**, a powerful library known for its rich text editing capabilities. Quill allows users to add formatted text, making offers more informative and visually appealing.

Challenge with PDF Conversion Quill generates rich text in HTML format. However, our existing PDF solution did not support the direct incorporation of HTML content. This posed a challenge in integrating Quill's output into our PDF-based offer documents.

Solution: HTML to PDF Conversion To overcome this challenge, we implemented an *HTML to PDF conversion* process. This conversion functions as follows:

- The offer, including all building blocks and rich text, is first composed in HTML format.
- This HTML document is then passed through an HTML to PDF converter.
- The converter effectively translates all formatting and content into a PDF document, retaining the rich text features.

Enhanced Flexibility with the ##offer## Placeholder To further enhance the flexibility in designing offers, we introduced a special placeholder, '##offer##'. This placeholder can be inserted anywhere in the rich text. The location of the '##offer##' placeholder determines where the building blocks (task lists) will be displayed in the final offer document. This feature allows users to choose whether the custom rich text appears above or below the offer table, providing greater control over the layout and design of the offer.

Conclusion

The integration of rich text into our project offers, along with the added flexibility of the '##offer##' placeholder, marks a significant improvement in the way we present information to clients. This implementation not only adds aesthetic value but also enhances the overall usability and effectiveness of our offer documents.

4.1.2 Workspace Authorization and Permission Management Implementation

In our application, we have implemented a robust system for managing permissions and authorizations within different workspaces. This system is valuable for maintaining data security and ensuring that only authorized users can perform certain actions within a workspace. The implementation primarily relies on Spring Framework and Aspect-Oriented Programming (AOP) to achieve these goals.

Background

Our application consists of multiple workspaces, each of which serves as an isolated environment for various tasks and projects. These workspaces can have multiple administrators who are responsible for configuring workspace-specific settings, managing members, and performing other administrative actions. To control access to these privileges, we needed to introduce a flexible and efficient authorization concept which has been explained in Section 3.2.4.

Annotations

To mark specific methods and endpoints within our application that require administrator-level access, we have created custom annotations. The primary annotation used for this purpose is `@AdminAccess`. By applying this annotation to controller methods, we indicate that only users with administrative privileges should be allowed to execute these methods.

Listing 4.1: Example of using `@AdminAccess` annotation

```
@DeleteMapping(value = "/settings")
@AdminAccess
public void resetSettings() {
    settingsService.resetSettings();
}
```

AOP Aspect

To intercept and process method calls marked with the `@AdminAccess` annotation, we employ Aspect-Oriented Programming (AOP). We define an `@Around` advice that is responsible for handling this annotation. This advice is executed before the annotated method, allowing us to perform authorization checks.

Listing 4.2: AOP Aspect for `@AdminAccess`

```
@Around("@annotation(AdminAccess)")
public Object handleAdminAccess(ProceedingJoinPoint joinPoint) throws Throwable {
    if (accessService.isAdmin()) {
        return joinPoint.proceed();
    }
    throw new ResponseStatusException(HttpStatus.FORBIDDEN);
}
```

In this code, the `handleAdminAccess` method is executed before the method annotated with `@AdminAccess`. It checks if the current user, as determined by the `accessService`, has administrative privileges for the current workspace. If the user is an administrator, the annotated method proceeds; otherwise, a `ResponseStatusException` with a status code of 403 (Forbidden) is thrown, denying access.

Access Service

The `accessService` is a critical component of our implementation. This service is responsible for determining the user's role within the current workspace. It receives information about the current user and their associated workspace, enabling it to perform database queries to verify the user's role and permissions. By the current implementation, it contains only two methods, *#isAdmin* and *#isOwner*, however with the solution presented, it is easily extendible and ready for a more complex authorization logic.

Conclusion

In conclusion, our implementation of workspace authorization and permission management enhances the security and control of our application. By combining custom annotations, Aspect-Oriented Programming, and a dedicated `accessService`, we dynamically check and enforce user roles within workspaces, allowing us to grant or deny access to specific functionalities based on administrative privileges. This system ensures that our application remains secure and that only authorized users can perform sensitive actions within workspaces.

4.1.3 Internationalization in Angular

Internationalization, often abbreviated as i18n, is a critical feature in software development. Angular has introduced a unique approach to internationalization, which is detailed in their official guide (`https://angular.io/guide/i18n-overview`). Unlike conventional methods, Angular's i18n system builds separate versions of the application for each language, presenting the challenge of serving and managing these different versions effectively.

Implementation and Usage

To utilize Angular's i18n feature, developers must follow specific steps:

- **Marking Text for Translation:** The first step involves marking the text within the application that requires translation. This is done using the i18n attribute. For example, in your HTML templates, you can mark text like this:

```
<h1 i18n>Hello World!</h1>
```

- **Extracting Translation Source Files:** Once the text is marked, the next step is to extract this text into translation source files. Angular CLI provides tools for extracting these marked texts into an industry-standard translation file format (XLIFF, XMB, etc.).
- **Translating Content:** The extracted files can then be handed over to translators who will provide translations for each marked text string. However, this part has been done by us due to the limited financial possibilities. Because of that, we have decided to only offer a German and English version due to the lack of knowledge in any other languages.
- **Building the Application per Language:** After translations are provided, the Angular application is built for each language. This results in different versions of the application, one for each language.

Handling Language Switching

A notable aspect of Angular's i18n feature is how it handles language switching. When a user changes their language preference, the application must redirect them to the appropriate version. This redirection is handled as follows:

```
window.location.href = `${window.location.protocol}//
${window.location.host}/${language}/#${this.router.url}`;
```

Here, the application dynamically updates the window's location URL, navigating the user to the corresponding language-specific version. The `${language}` variable in the URL is replaced with the user's selected language, ensuring they are directed to the correct localized version.

Conclusion

Angular's approach to internationalization, with its unique method of building separate application versions for each language, offers a robust solution for creating multi-language applications. By following the steps of marking text, extracting translations, and rebuilding the application per language, developers can effectively internationalize their Angular applications, catering to a diverse global audience.

4.1.4 Mail Service Integration

Mailing is a central aspect of the most common software systems. Thus, it's a very common task and there are several drop-in solutions out in the market to achieve this goal.

Installation

To implement the mailing solution, we have decided to opt for the default spring solution. Spring offers a designated Maven library for mailing, called *spring-boot-starter-mail*. As soon as the library is being added to a project, the spring property files will notice the change and allow the user to configure the mail gateway. To get started quickly, we generated a Gmail E-Mail account to use for outgoing mail. After generating a new Gmail E-Mail, we were able to configure our Mail Gateway, to use the Gmail gateway.

Example of Configuring a Mail Gateway

```
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=consilium.dev@gmail.com
spring.mail.password=<generate an app password>
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
```

Usage

Now after configuring the E-Mail gateway, you are ready to send E-Mails straight away from your code. To achieve this goal, we have created a *MailService* class, which wraps around the *JavaMailSender* class. The *JavaMailSender* class is being injected into our service and is the one that has been configured by our configuration details. The final step is to use the *JavaMailSender* to your desired needs, in which you define the recipient, followed by subject and text details.

Example of sending an E-Mail

```
SimpleMailMessage message = new SimpleMailMessage();
message.setFrom(workspaceInvitation.getInviterEmail());
message.setReplyTo(workspaceInvitation.getInviterEmail());
message.setTo(workspaceInvitation.getInviteeEmail());
message.setSubject(localizeSubject(workspaceInvitation.getInviteeLanguage()));
message.setText(localizeText(workspaceInvitation));
javaMailSender.send(message);
```

4.1.5 Invoice Swiss QR Code Integration

This section shows how we integrated the Swiss QR Code on our Invoices, allowing for straightforward payment.

As described in Section 3.2.6, we previously decided to use an external library to create the QR Code. The chosen provider (https://www.qr-invoice.ch/) offers a very clean integration. It was imported through the Maven library called *ch.codeblock.qrinvoice.core*. Through the use of a builder the QR Code could easily be generated:

Example of creating an Invoice QR-Code

```
QrInvoiceBuilder
                .create()
                .creditorIBAN(creditor.getIban())
                .paymentAmountInformation(p -> p
                        .chf(totalPrice))
                .creditor(c -> c
                        .combinedAddress()
                        .name(creditor.getName())
                        .addressLine1(creditor.getAddress().getStreet())
                        .addressLine2(creditor.getAddressLine2())
                        .country(COUNRY_CODE)
                )
                .ultimateDebtor(d -> d
                        .combinedAddress()
                        .name(debtor.getName())
                        .addressLine1(debtor.getAddress().get().getStreet())
                        .addressLine2(debtor.getAddressLine2())
                        .country(COUNRY_CODE)
                )
                .build();
```

The library allowed us to output the QR-Code as an Image, which we could then print on the invoice.

Chapter 5

Results

This Chapter highlights the results of this thesis and summarizes it.

5.1 Results

In this Section, we provide an overview of what we achieved during this thesis but also highlight the requirements, which were initially defined, but not addressed in the end.

Overview

This section presents the results achieved in the development of *Consilium*, focusing on key features that were implemented to enhance its functionality and market readiness. These features include multi-user workspaces, versioning of offers, activity-based time tracking, invoicing with Swiss QR code integration, internationalization, and cloud deployment to Amazon Web Services (AWS). Additionally, the results are compared to previously defined requirements.

Login The basis for introducing multi-user functionality was introducing authentication. The login is kept very simple as shown by the login screen:

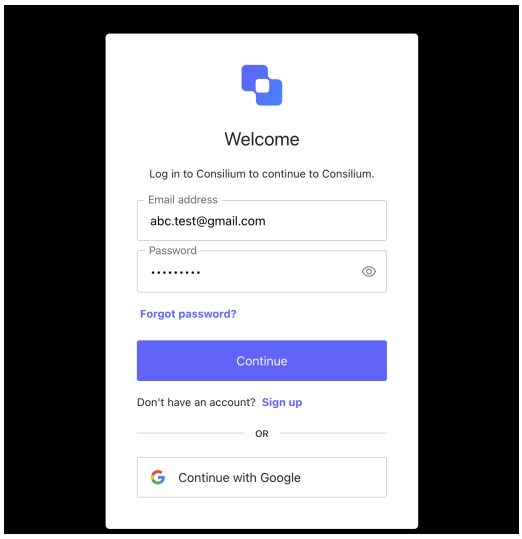

Figure 5.1: Consilium login screen

Users can now easily log into *Consilium*. This addresses the requirement (Security - Login, as it is now possible to secure the data with a login.

Multi-User Workspaces The implementation of multi-user workspaces in Consilium marks a significant advancement in collaborative functionality. This feature allows multiple users to work simultaneously within the same environment, enhancing teamwork and efficiency. It has been observed that with the introduction of this feature, project collaboration has become more streamlined, and the user experience has improved notably, especially in scenarios involving team-based tasks.

Users can create and work with multiple workspaces through a simple menu:

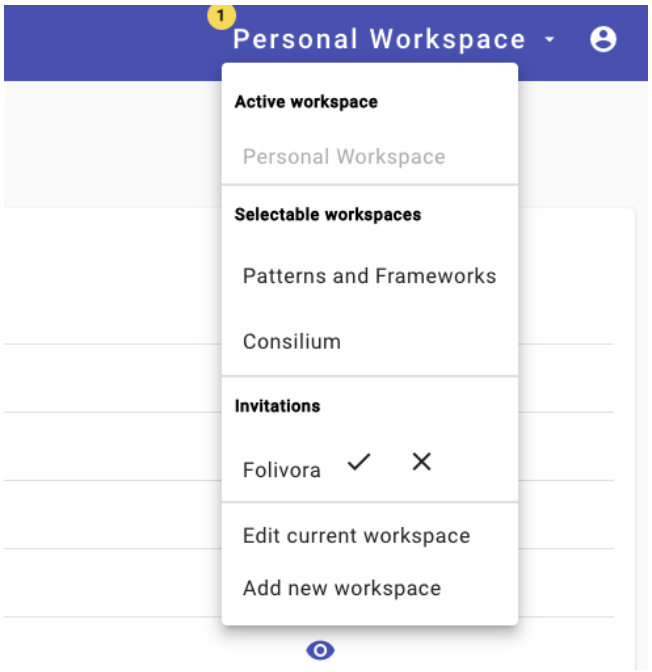

Figure 5.2: Workspace menu component

This screenshot shows the menu component which can be used to switch between active workspaces. It also allows to edit the current workspace, as well as adding a new one.

With the implementation of this feature, we fulfilled the requirement (Collaboration - Access, as it is now possible to work in a team.

Versioning of Offers Version control was integrated into the offer generation process of *Consilium*. This feature ensures that every change made to an offer is tracked and recorded, allowing users to access and revert to previous versions if necessary. The impact of this feature has been substantial in maintaining the integrity of data and enhancing the accountability of modifications made by different users. Previous offers can be reverted to in modification screen of an offer. By using a drop-down, the selected offer version, with the text and the offer blocks is active and displayed.

This addresses the requirement Offers - Versioning, as it is now possible to view old versions of the offers and they can even be reverted to.

Rich Text Editor integration into Offers The introduction of the rich text editor for the offer text allowed a way more flexible creation of offers compared to only having a plain text editor. This allows potential users to be able to model even complex offers with *Consilium*. The following figure gives an example of an offer that can be realized using this feature and shows the editor:

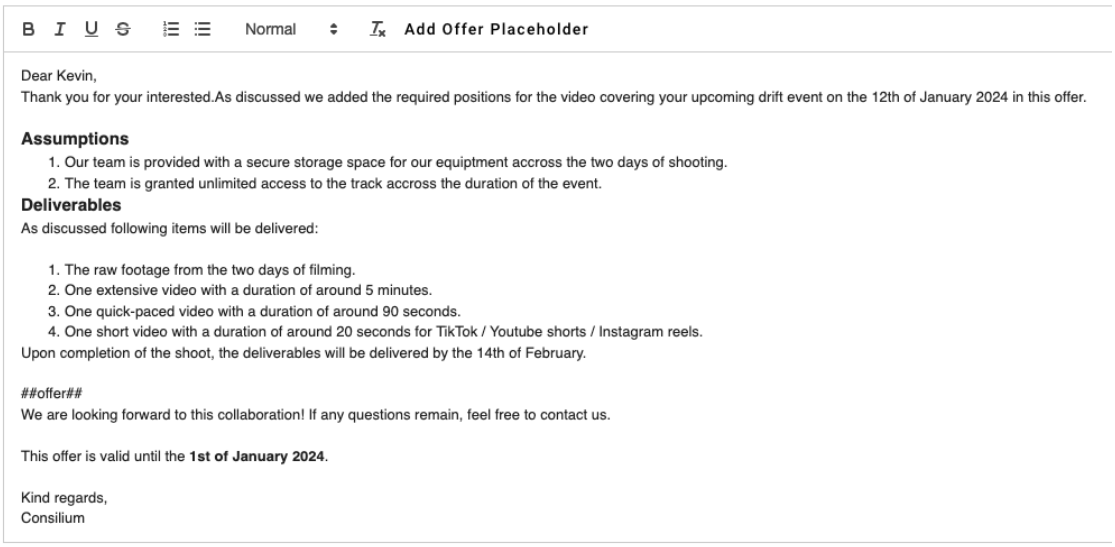

Figure 5.3: Rich text editor

As displayed in the graphic, the rich text editor allows much more flexibility with styling and allows more professional-looking documents to be generated. Through the use of the offer placeholder, the offer blocks can be positioned anywhere the user likes.

Appendix B shows, how this text would look on an exported offer.

The implementation of this feature allowed us to address multiple requirements. Firstly, this allows us to check off the requirement Offers - Customization, as great customization options have been added. Secondly, Requirements - Binding Times as well as Offers - Obligations can easily be modeled using this editor.

Activity-Based Time Tracking The introduction of activity-based time tracking allows users to work very precise. This feature allows users to define categories and specific activities which then can be used to log time. Figure 5.4 is an example of a time-tracking activity.

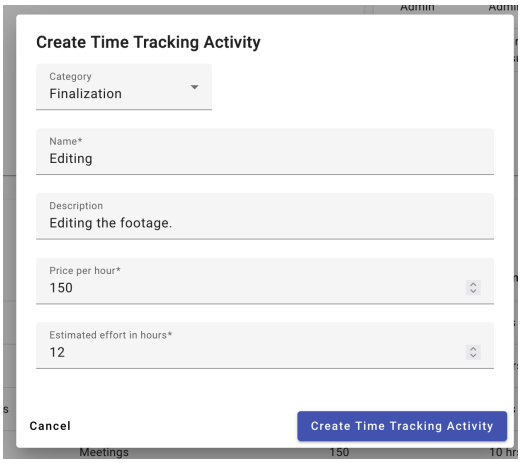

Figure 5.4: Time tracking activity modal

As seen, the activity can be put into a category. In this example, the *Editing* activity is added to the category *Finalization*. Additionally, on each activity, a price per hour is defined as well as an estimate in hours.

This allowed us to fulfil the requirement Billing - Time Log, and bring in monitoring functionality on top.

The categories are now often used to summarize multiple activities, to give a better overview of how the project is going, where the efforts are high, and where the estimates were exceeded for example. To further improve monitor-ability, statistical graphics were added:

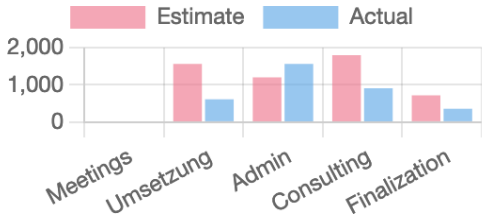

Figure 5.5: Estimate vs actual effort graphic

This allows project managers to see the efforts spent across each category.

Due to the limited time of this thesis, not the full scope described in the technical concept was realized. We decided on implementing the more flexible feature, and not implementing the import feature. This decision was taken, as the import feature was very practical, but not all workflows could be modeled.

Invoicing Based on Time Tracking and Swiss QR Code To allow users to invoice the time spent, an invoicing feature was introduced. This feature is realized, by taking the time spent, in an invoicing period, and adding all work-log entries together. Further, the user can enter a text, which will appear on the invoice above and under the invoice positions. To improve usability a preview was added, and all the entries are grouped by activity and category, making it easier on the eye for the project manager as well as the client:

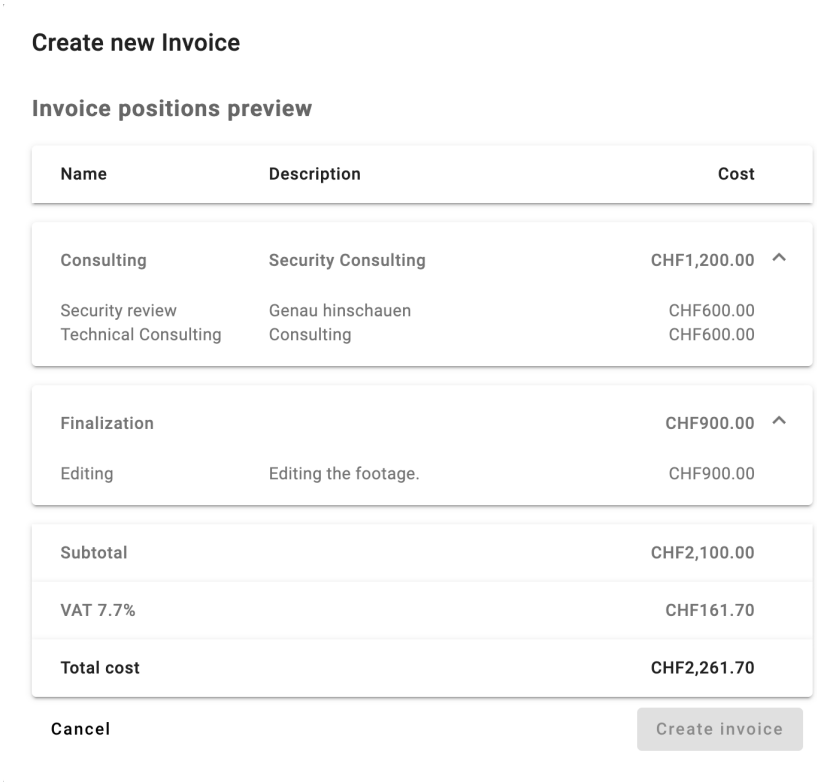

Figure 5.6: Invoice preview

This allows users to exactly see, what they are invoicing.

On top of that, a Swiss QR Code was implemented, which is printed on the bill and can be scanned, allowing for both a professional look, as well as an easy payment process. How an invoice created with *Consilium* could look, can be seen in Appendix C.

This allowed us to fulfill the requirement Billing - Invoice.

Internationalization *Consilium* now supports English and German, to cater to a broader user base. This internationalization effort is a strategic move to make the software accessible to a wider audience and to facilitate its entry into new markets.

This also allows us to check off the non-functional requirement Supportability - Localization.

Cloud Deployment to AWS Finally, the deployment of *Consilium* to Amazon Web Services (AWS) marks a critical step in its scalability and reliability. This cloud deployment ensures that *Consilium* can handle increased user loads and data securely and efficiently. The transition to AWS has shown promising results in terms of enhanced performance, reduced downtime, and improved data security.

Comparison to Requirements

This section aims to compare the results to the requirements, previously defined. While most requirements could be addressed, many new ones came on top and some of those, which were initially defined, were not fulfilled in the end.

We will now go into detail, as to why some of the requirements were not fulfilled in the end.

Offers - Templates: This feature would have allowed users to define offer templates. For multiple reasons we did not implement this feature and therefore did not fulfill this requirement: Firstly, we did not want to implement features only halfway and only wanted to implement features, we had the capacity to implement properly. In this case, we would have wanted to introduce the possibility of using placeholders and variables that can be defined per workspace, per customer, and project for example. This made the feature too big to implement, and other features were chosen instead.

Offers - Cloning: This requirement requires the possibility to clone existing offers. This feature was in the end not implemented as it did not offer something, that is not achievable otherwise. Although it is not the most beautiful workflow, the text of offers can easily be copied manually. This feature would have made the workflows for users with many similar offers easier, but we wanted to focus on making more complex and different workflows possible.

Collaboration - Document Storage: This feature would have enabled users to give access to customers and have document storage to show the customers what they are working on. The reason we did not implement this feature yet is, that we identified *Consilium* becoming a marketplace, as one of the most interesting visions for the future. This would also allow users and their customers to share documents. As we never wanted to implement temporary functionality, which we know will eventually be replaced, we decided against implementing this feature now. If *Consilium* is continued, and taken down the path of becoming a marketplace platform, this feature would be implemented.

5.2 Summary and Outlook

This section summarises this thesis and looks at what could come next for *Consilium*.

Summary

We obtained feedback from a potential customer of Consilium and a previous team member, this feedback can be found in Appendix F.

The *Consilium* project, led by Joel Sauvain and Noah Stalder under the guidance of our advisor at the OST Eastern Switzerland University of Applied Sciences, has made significant strides in addressing the challenges of efficient data management in small to medium enterprises. Employing innovative methodologies such as advanced data analysis, user-centric design, and scalable architecture, the project achieved notable outcomes.

Throughout its journey, the project overcame various technical challenges and adapted to evolving user needs by working agile, contributing to its success and providing valuable insights for future endeavours.

By sticking to our principles, we achieved significant strides into building a tool from which a considerable audience could greatly profit. Feedback from representatives of the target audience highlights the clean integration of end-to-end processes into *Consilium* as the major takeaway.

Future Prospects and Outlook

Looking ahead, the *Consilium* project serves as a foundation for future developments in data management technology. The collected feedback from freelancers shows *Consilium* can already be the best possible solution for part of our customer segment. There are workflows, that can not be modeled better with any other tool available on the market. With some minor tweaks, *Consilium* could be taken even further and really attack this market. With more time at hand, there are many features we would like to implement.

When taking the project further, and trying a market entry, first a pricing model would need to be defined. In addition, we identified, that further work on user experience improvements would be needed, as our research showed. Another step, that would help with the market entry, would be implementing an import functionality, which could see customers moving from another tool to *Consilium*.

A big step we would like to see taken would be transforming *Consilium* into a marketplace platform. First up, *Consilium* would greatly profit from community features, which would allow users to share templates or data to improve each other's experience with *Consilium*. Then, *Consilium* could introduce a customer view, where customers could look for freelancers, which can implement their projects. They could directly see references of their previous work and contact them for collaborations. On top of that, they could publish descriptions of projects, for which they are looking for someone to do them for them. Freelancers on the other hand could see these project descriptions and apply themselves. *Consilium* would be suitable for this kind of application, as a big part of the data required and the views that would be necessary are already present. Through the use of artificial intelligence, we could develop a match-making algorithm, which would bring suitable customers and service providers together even more efficiently.

To summarize, those are the key areas we identified for future exploration:

- **Setting a Price:** Deciding how much to charge for Consilium is important for getting into the market.
- **Advanced Features:** Developing features such as predictive analytics and AI-driven insights to further augment the solution's capabilities.
- **Market Adaptation:** Tailoring the solution for various market segments and industries, showcasing its versatility.
- **User Experience Enhancement:** Continuously improving the user interface based on feedback and emerging trends.

The *Consilium* project demonstrates that small, agile teams, with the right advisors backing them and in close contact with their customer segment, can deliver software, that can compete with existing, established players in a competitive market.

Bibliography

[1] M. Fowler, "Richardson maturity model," Dec 2023. [Online]. Available: https://martinfowler.com/articles/richardsonMaturityModel.html

[2] C. Larman, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2004.

[3] A. Furda, C. Fidge, A. Barros, and O. Zimmermann, *Re-engineering data-centric information systems for the Cloud–A method and architectural patterns promoting multi-tenancy*. Elsevier - Morgan Kaufmann, 12 2017, pp. 227–251.

[4] L. Eder, "Java, sql and jooq." Nov 2023. [Online]. Available: https://blog.jooq.org/

[5] "Cloud computing patterns," 2020. [Online]. Available: https://www.cloudcomputingpatterns.org/tenant_isolated_component/

[6] R. C. Martin and J. O. Coplien, *Clean code: a handbook of agile software craftsmanship*. Upper Saddle River, NJ [etc.]: Prentice Hall, 2009.

[7] R. C. Martin, *Clean Architecture: A Craftsman's Guide to Software Structure and Design*, 1st ed. USA: Prentice Hall Press, 2017.

[8] N. Thiago, "Tutorial: Deploy a spring boot application to the cloud," Jan 2020. [Online]. Available: https://dzone.com/articles/tutorial-deploy-a-spring-boot-application-to-the-c

[9] May 2021. [Online]. Available: https://community.auth0.com/t/how-to-move-from-development-key-to-production-key-for-tenant/62860

[10] D. Syer and P. Webb, Nov 2023. [Online]. Available: https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/

Appendix A

Project Definition

Aufgabenstellung Studienarbeit Joel Sauvain und Noah Stalder

Software-as-a-Service-basierte Projektmanagementsoftware für Selbstständige und Kleinunternehmen

1. Auftraggeber und Betreuer

Diese Arbeit wird in Zusammenarbeit mit dem Cloud Application Lab am IFS durchgeführt.

Ansprechpartner (extern):

n/a

Betreuer (OST):

Prof. Dr. Olaf Zimmermann, OST Dept. I, Institut für Software, olaf.zimmermann@ost.ch

2. Ausgangslage

In einem Dreierteam entstand im Engineering-Projekt im FS 2023 der Prototyp einer webbasierten Projektmanagementsoftware für Einzel- und Kleinunternehmungen. Die bestehende Lösung umfasst Funktionalitäten wie das Erfassen von Kunden, Projekten und Offerten. Zudem existiert eine rudimentäre Zeiterfassung. Während dieser Prototyp das Potential und den Nutzen der Anwendung bereits aufzeigt, reichen Funktionsumfang und Architektur aktuell noch nicht für eine mögliche Markteinführung aus. Der Prototyp soll daher nun zu einem Produkt ausgebaut und dabei als cloudfähig gemacht werden, beispielsweise als Software-as-a-Service (SaaS)-Offering.

3. Ziele der Arbeit und Liefergegenstände

Die Studienarbeit hat zwei Hauptziele. Zum einen soll die Architektur der Projektmanagement-Software so konzipiert werden, dass ein kontinuierlicher, kosteneffizienter Betrieb bei mindestens einem der führenden öffentlichen Cloud-Provider möglich wird (AWS, Azure und/oder GCP). Dabei sollen die definierenden Eigenschaften und Entwurfsprinzipien von Cloud-Native Applications berücksichtigt werden und ein bedarfsgerechtes Konzept für Mandantenfähigkeit entwickelt werden.

Weiterhin sollen neue Funktionsbereiche analysiert und umgesetzt werden, um die Lösung im Markt von bestehenden Angeboten abzuheben. Dazu wird der Stand der Technik in Form eines kriterien-basierten Marktüberblicks ermittelt. Die Stossrichtungen für die funktionalen Neuerungen sollen gemeinsam mit dem Betreuer (als Repräsentanten der Zielgruppe) erhoben und priorisiert werden.

Liefergegenstände. Die zentralen Deliverables der Arbeit sind:

- Source Code und zugehörige Dokumentation (z.B. Schritte für lokale und Cloud-Deployments, Architekturbeschreibung, Domain Model, Programmierreferenz mit API-Design)
- Users Guide (Installationsanweisung, Starthilfe für die Nutzung, Tutorial und Beispiele)

- Studienarbeits-Bericht mit Cloud-Konzept (siehe dazu Punkt 6)

Kritische Erfolgsfaktoren. Für die Bewertung der Arbeit sind folgende Kriterien besonders relevant:

- Zielgruppengerechter Funktionsumfang und sinnvoller Umgang mit Erweiterungsvorschlägen
- Benutzbarkeit der Anwendung, insbesondere Angemessenheit der Einarbeitungsaufwände
- Betreibbarkeit: Installationsaufwand, Robustheit, Kosten und Management von Cloud-Deployments
- Wartbarkeit, insbesondere Erweiterbarkeit und Zukunftssicherheit, beispielsweise im Hinblick auf steigende User- und Entwicklerzahlen
- Software Engineering-Hygienefaktoren wie Versionsverwaltung, automatisierte Builds und Tests, Testabdeckung sowie angemessener Umfang, Inhalt und Stil der Dokumentation

4. Unterstützung

Die erwartete und effektiv erhaltene Unterstützung wird durch die Studierenden protokolliert.

5. Zur Durchführung

Mit dem Betreuer finden in der Regel wöchentlich Besprechungen statt (Treffen auf dem Campus oder Telefon- bzw. Webkonferenz). Zusätzliche Besprechungen sind nach Bedarf zu veranlassen. Alle Besprechungen, bei denen eine Vorbereitung durch den Betreuer nötig ist, sind von den Studierenden mit einer Traktandenliste vorzubereiten. Beschlüsse sind in einem Protokoll zu dokumentieren.

Für die Durchführung der Arbeit ist ein Projektplan zu erstellen. Dabei ist auf einen kontinuierlichen und sichtbaren Arbeitsfortschritt zu achten. Arbeitszeiten sind zu dokumentieren. Sofern nicht in dieser Aufgabenstellung vorgeben, sind die Studierenden für die Auswahl und korrekte Anwendung Ihrer Hilfsmittel (also Werkzeuge, Libraries, Frameworks, SaaS-Angebote, etc.) selbst verantwortlich. Auf Wunsch kann eine an der OST gehostete virtuelle Maschine zur Verfügung gestellt werden.

Die Spezifikation der Anforderungen geschieht durch die Studierenden in Absprache mit dem Betreuer. Bei Disputen entscheidet der Betreuer in Rücksprache mit den Studierenden über die definitiv für die Arbeit relevanten Anforderungen.

Rechercheergebnisse, Anforderungsdokumentation und Architekturbeschreibung sollen im Laufe des Projektes mittels Milestones vom Betreuer abgenommen werden. Zu abgegebenen Arbeitsresultaten wird ein vorläufiges Feedback abgegeben. Die definitive Beurteilung der Arbeit erfolgt auf Grund der am Abgabetermin abgegebenen Liefergegenstände inklusive Dokumentation und Bericht.

Das dritte Teammitglied, das an der Erstellung des Prototypen im Engineering-Projekt beteiligt war, hat dem Projektteam sein Einverständnis mit der Weiterentwicklung im Rahmer dieser Arbeit gegeben. Die Rechte an den Ergebnissen der Arbeit werden in einer separaten Vereinbarung definiert. Der Bericht darf veröffentlicht werden.

6. Dokumentation

Die Arbeit ist nach den Richtlinien des Studiengangs Informatik zu dokumentieren (s.u.). Die zu erstellenden Dokumente sind im Projektplan festzuhalten. Alle Dokumente sind nachzuführen, d.h. sie sollten den Stand der Arbeit bei der Abgabe in konsistenter Form dokumentieren.

Bei der Projektdokumentation und deren Abgabe sind die allgemeinen Informationen zu Studien- und Bachelorarbeiten sowie der "Leitfaden für Studien- und Bachelorarbeiten" des Studiengangs (insbesondere Abschnitt 5.5) zu beachten. Es steht beispielsweise eine "Orientierungshilfe für die Dokumentation einer Studien- oder Bachelorarbeit" zur Verfügung.

7. Termine

Die Termine wurden vom Sekretariat des Studiengangs Informatik in MS Teams veröffentlicht. Ihre Kenntnisnahme soll in einem Sitzungsprotokoll dokumentiert werden.

Erste Semesterwoche, beginnend mit dem 18.09.23	Beginn der Arbeit: Ausgabe der Aufgabenstellung durch die Betreuer und Kickoff-Meeting am 20.09.2023.
21.12.2023	Hochladen aller verlangten Dokumente auf https://avt.i.ost.ch/ bis 17:00. Abgabe des Berichts an den Betreuer/die Betreuerin.

8. Beurteilung

Eine erfolgreiche Studienarbeit zählt 8 ECTS-Punkte pro Studierenden. Für 1 ECTS-Punkt ist eine Arbeitsleistung von 30 Stunden budgetiert.

Es gelten die Bestimmungen des Bachelor-Studiengangs Informatik zur Durchführung und Bewertung von studentischen Arbeiten ("Leitfaden für Studien- und Bachelorarbeiten", siehe oben).

Rapperswil, 20. 09. 2023

Prof. Dr. Olaf Zimmermann
Institut für Software
OST

Appendix B

Example Offer

Kevin Tester
Galaxy Street
15000 Milchstadt

Offer 000202 - Drift event shooting

Dear Kevin,

Thank you for your interested.As discussed we added the required positions for the video covering your upcoming drift event on the 12th of January 2024 in this offer.

Assumptions

1. Our team is provided with a secure storage space for our equiptment accross the two days of shooting.
2. The team is granted unlimited access to the track accross the duration of the event.

Deliverables

As discussed following items will be delivered:

1. The raw footage from the two days of filming.
2. One extensive video with a duration of around 5 minutes.
3. One quick-paced video with a duration of around 90 seconds.
4. One short video with a duration of around 20 seconds for TikTok / Youtube shorts / Instagram reels.

Upon completion of the shoot, the deliverables will be delivered by the 14th of February.

Pos	Description	Quantity	Price	Total
023	Photograph	16 hour	CHF 125.00	CHF 2’000.00
024	Drone rent	2 day	CHF 400.00	CHF 800.00
025	Drone Operator	12 hour	CHF 90.00	CHF 1’080.00
026	Editor	6 hour	CHF 100.00	CHF 600.00
Total				**CHF 4’480.00**

We are looking forward to this collaboration! If any questions remain, feel free to contact us.

This offer is valid until the **1st of January 2024**.

Kind regards,

Consilium

Appendix C

Example Invoice

Kevin Tester
Galaxy Street
15000 Milchstadt

Rechnunbgsdatum 16 December 2023
Rechnungnummer 378247894
Leistungsperiode 03.10.2023 - 16.12.2023
Zahlbahr bis 16.01.2024

My Fancy Invoice

Moin Meister
Schau dir diese Schöne Rechnung an.

Positionen	Menge	Einh.	Einh.-Preis	Betrag CHF
Umsetzung				
Programmieren	0.92	Std	200.00	183.33
Testen	7.00	Std	200.00	1400.00
Integration Tests	2.32	Std	200.00	463.33
Subtotal Umsetzung				**2046.67**
Consulting				
Security review	8.33	Std	300.00	2500.00
Xn abig	6.83	Std	350.00	2391.67
Subtotal Consulting				**4891.67**
Admin				
Projektleitung	9.83	Std	150.00	1475.00
Scrum	16.23	Std	150.00	2435.00
Subtotal Admin				**3910.00**
Subtotal				**10848.33**
MWST				835.32
Total				**11683.65**

Freundliche Grüsse
Consilium

Empfangsschein

Konto / Zahlbar an
CH93 0076 2011 6238 5295 7
Max Muster
Musterstrasse 20
8000 Zürich

Referenz
RF78 123

Zahlbar durch
Kevin Tester
Galaxy Street
15000 Milchstadt

Währung	Betrag
CHF	11 683.65

Annahmestelle

Zahlteil

Währung	Betrag
CHF	11 683.65

Konto / Zahlbar an
CH93 0076 2011 6238 5295 7
Max Muster
Musterstrasse 20
8000 Zürich

Referenz
RF78 123

Zahlbar durch
Kevin Tester
Galaxy Street
15000 Milchstadt

Appendix D

Visual Comparison to Foundation

Dashboard

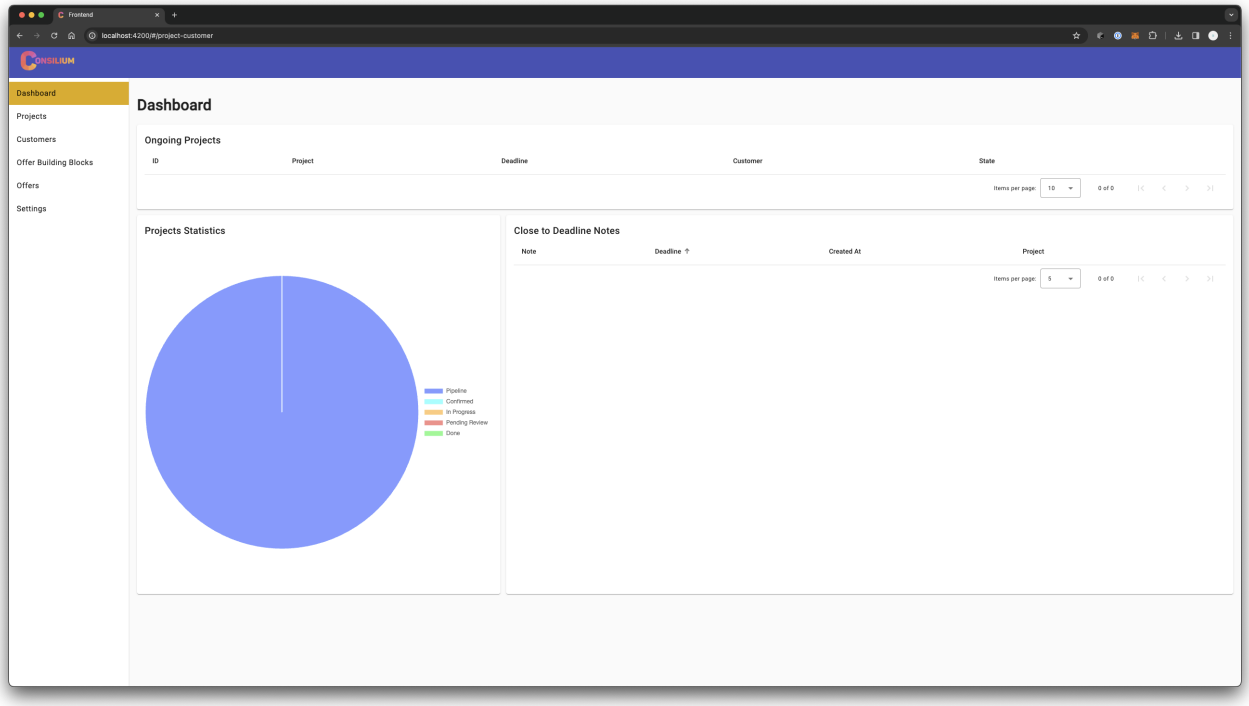

Figure D.1: Dashboard Prototype

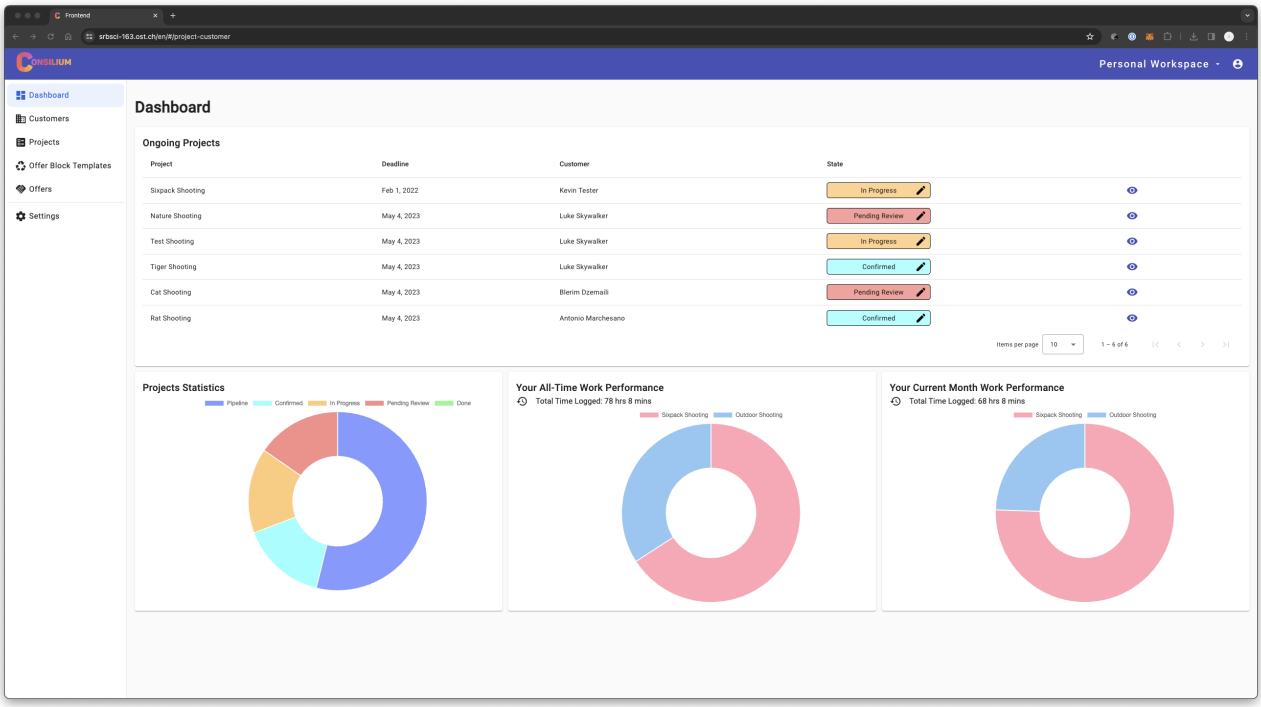

Figure D.2: Dashboard Now

Settings

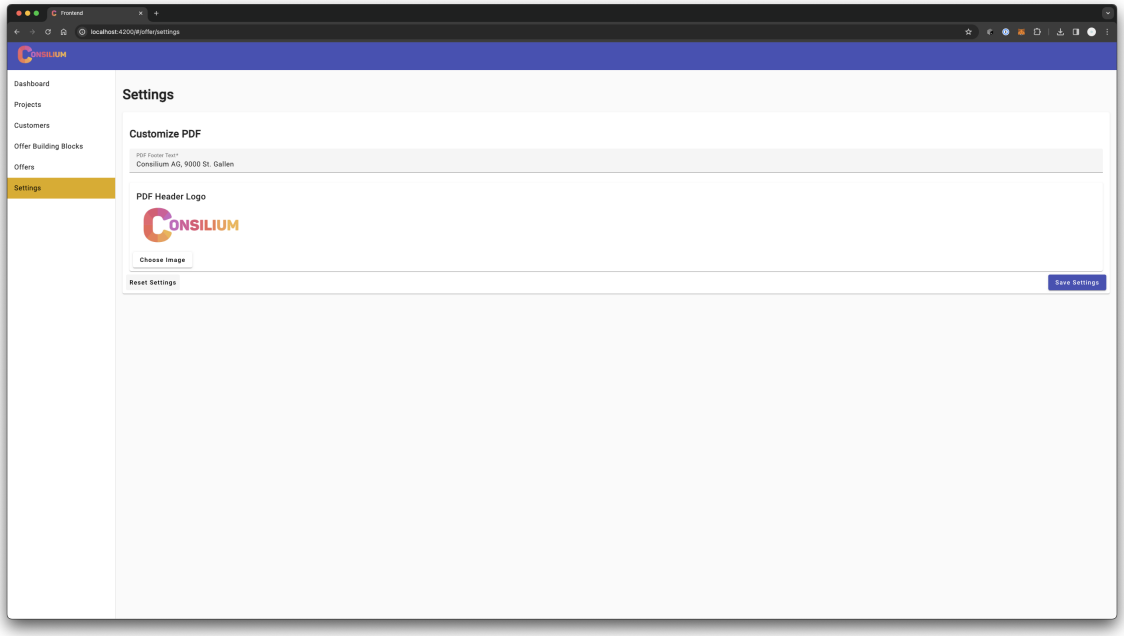

Figure D.3: Settings Prototype

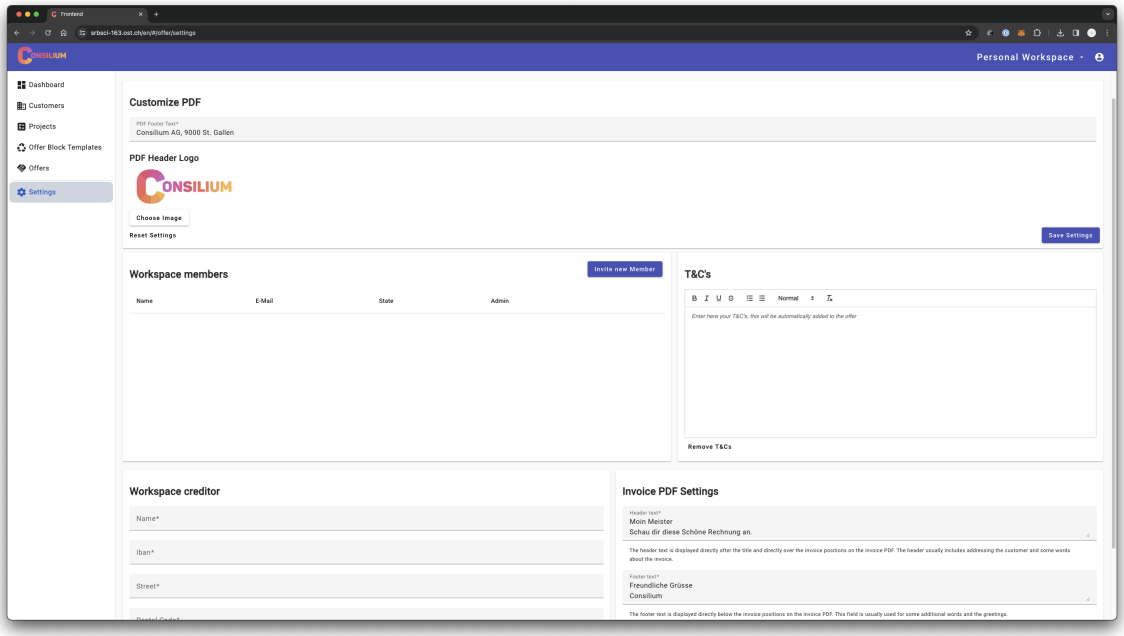

Figure D.4: Settings Now

Project Overview

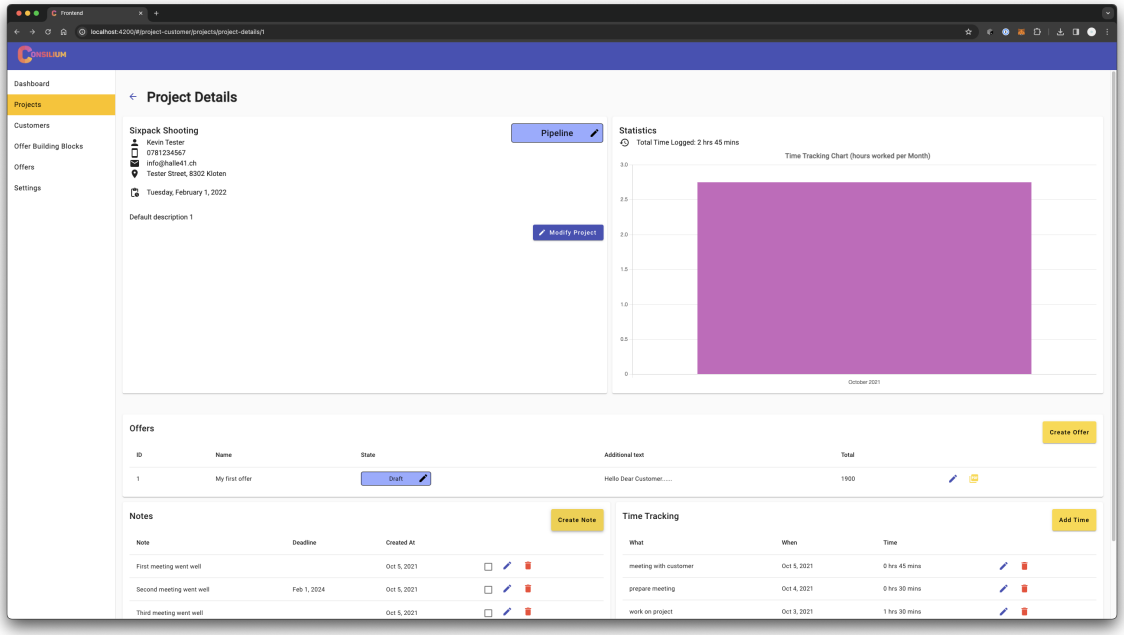

Figure D.5: Project Overview Prototype

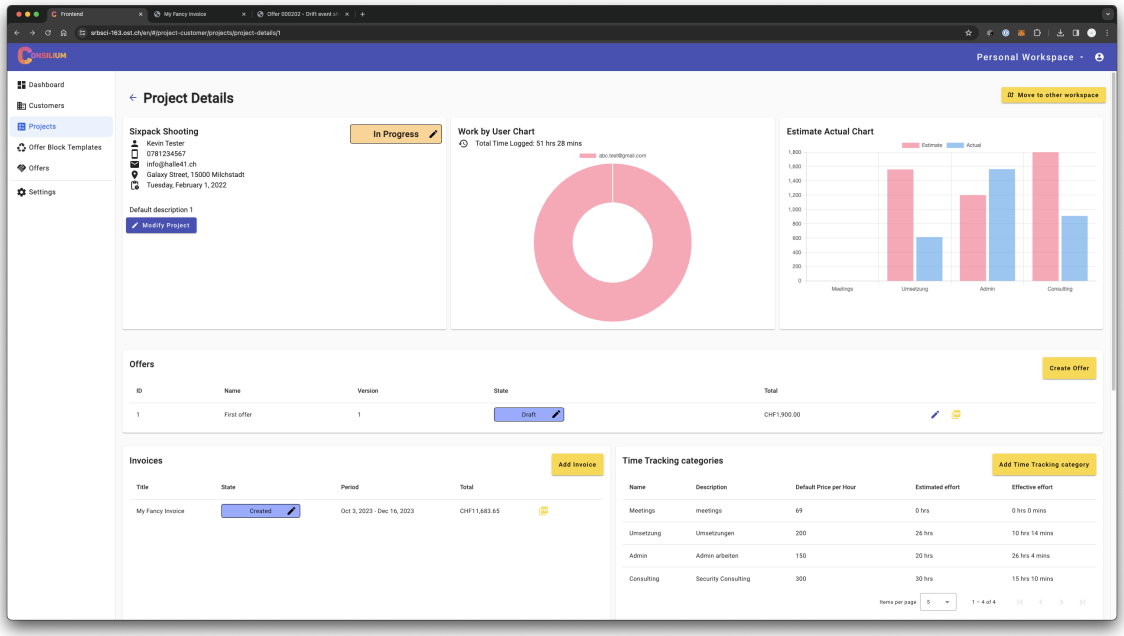

Figure D.6: Project Overview Now 1

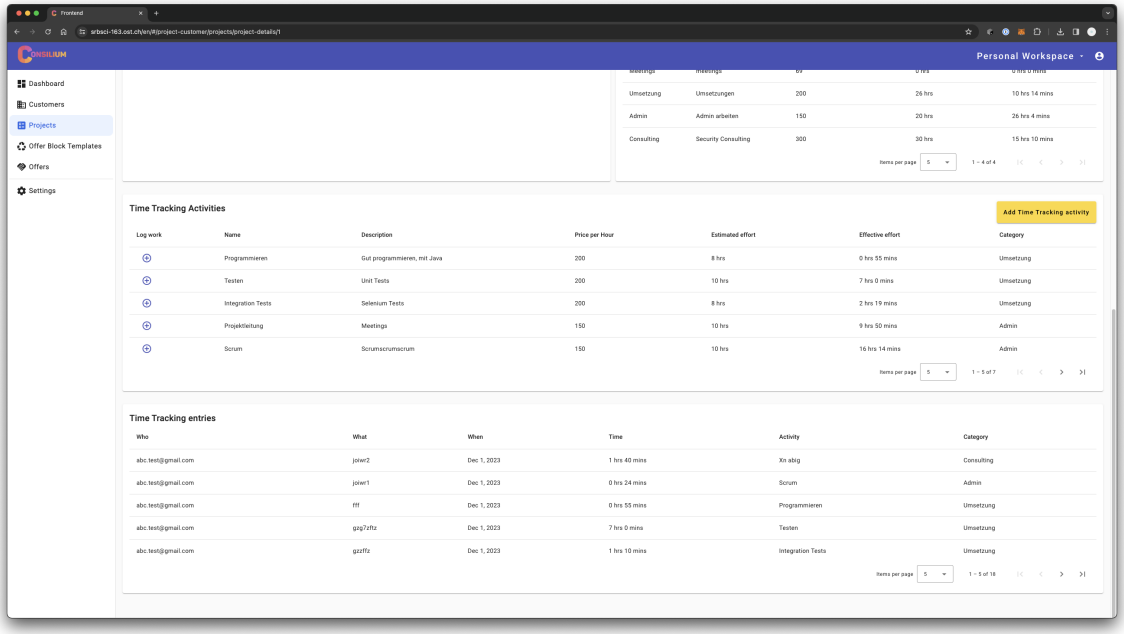

Figure D.7: Project Overview Now 2

Offer

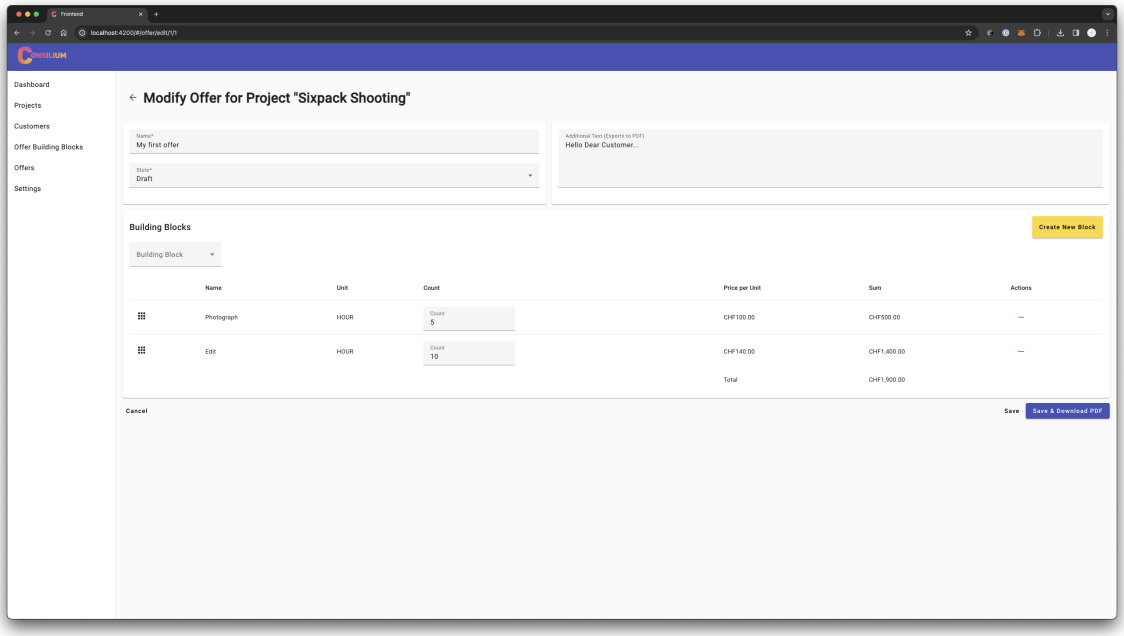

Figure D.8: Offer Prototype

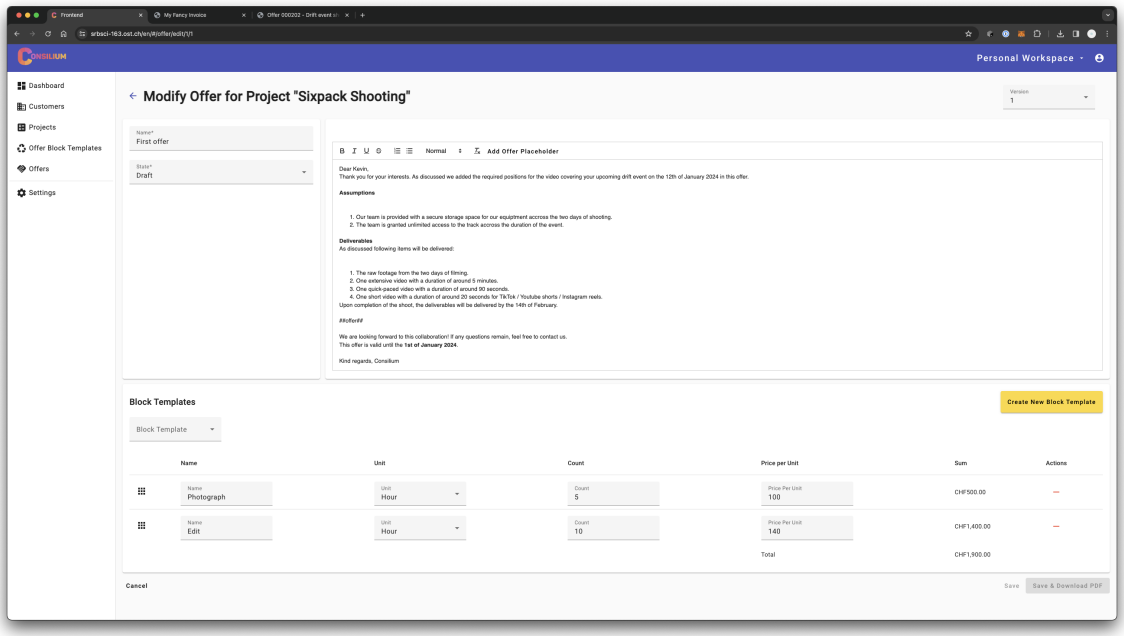

Figure D.9: Offer Now

Invoice

Not contained in the prototype

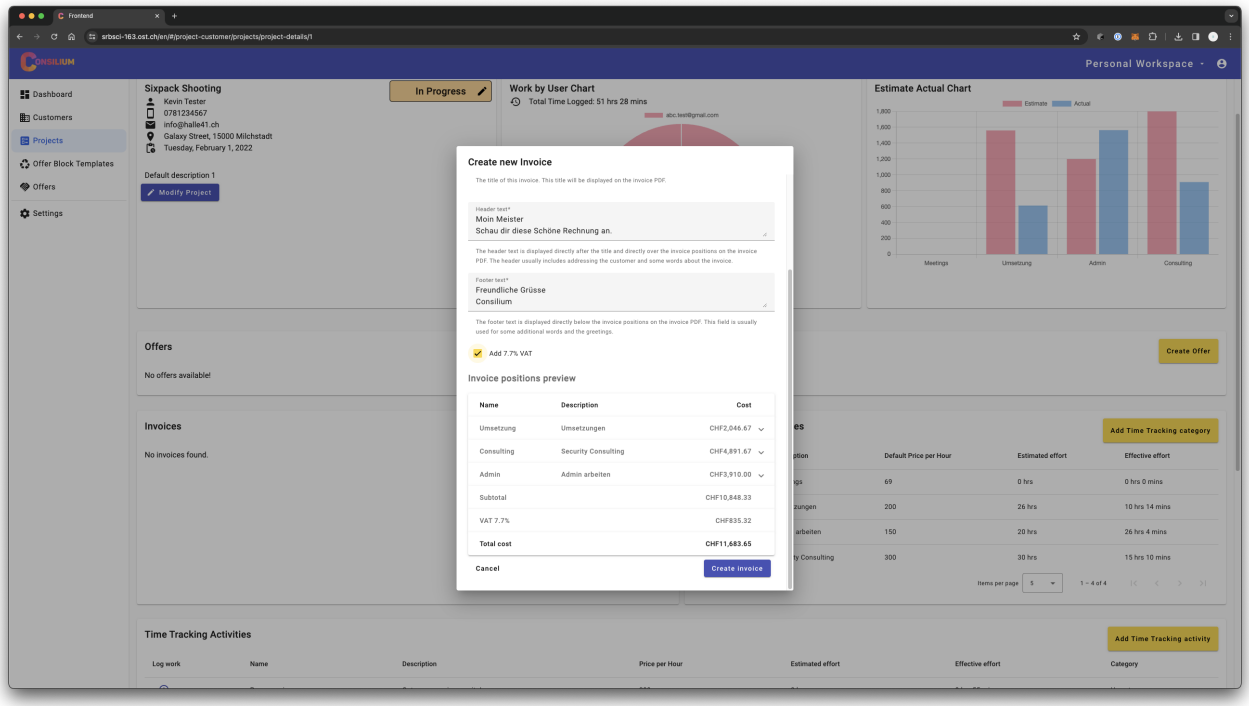

Figure D.10: Invoice Now

User / Auth

Not contained in the prototype

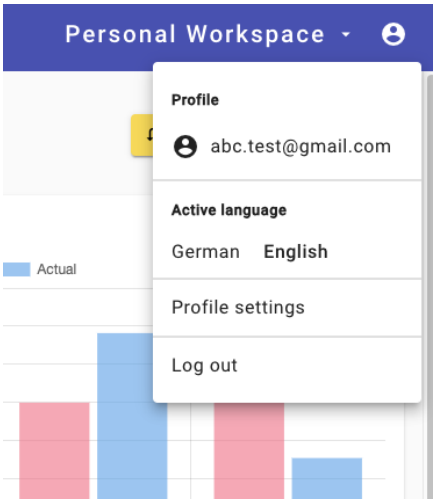

Figure D.11: Profile Now

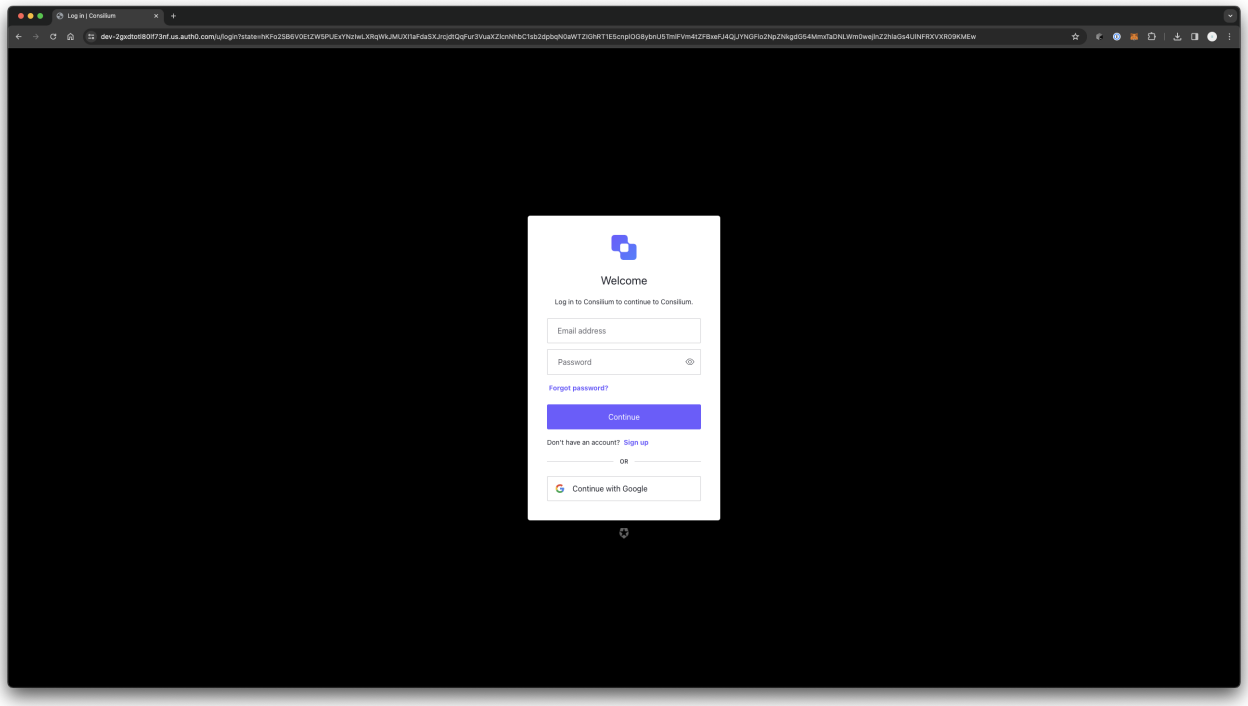

Figure D.12: Authentication Now

Workspace

Not contained in the prototype

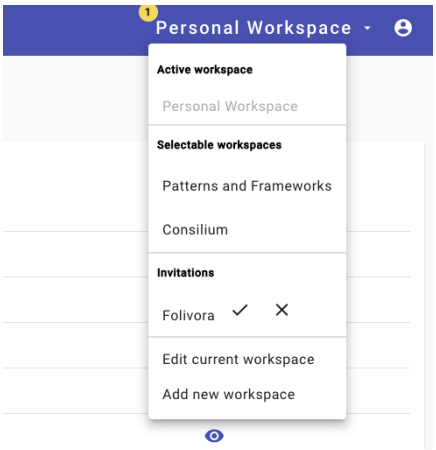

Figure D.13: Workspace Now

Customers

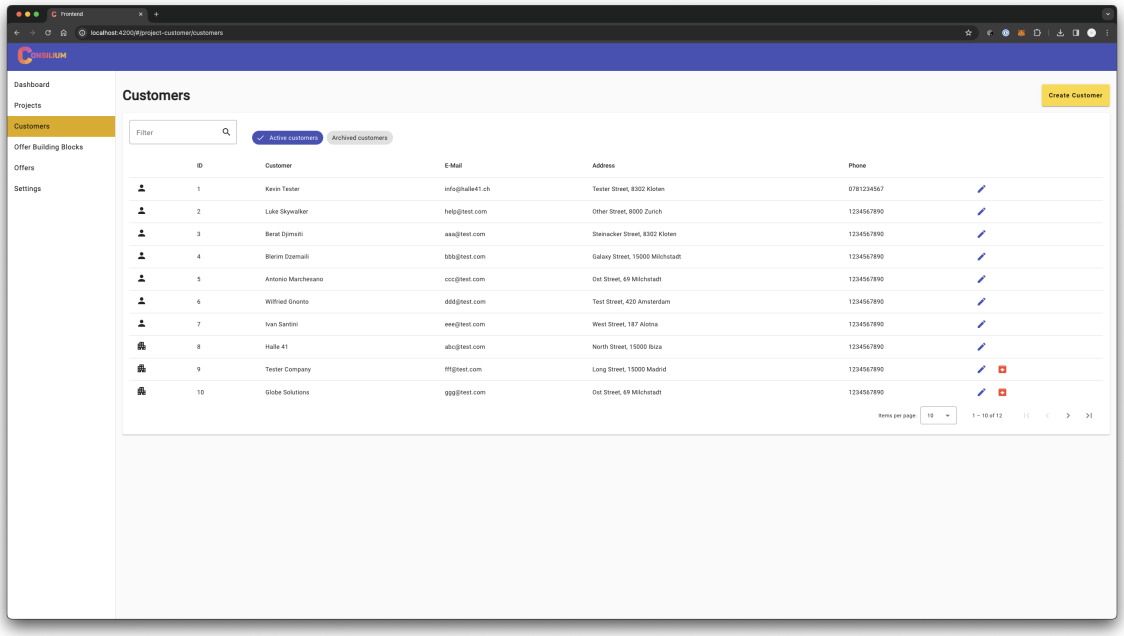

Figure D.14: Customers Prototype

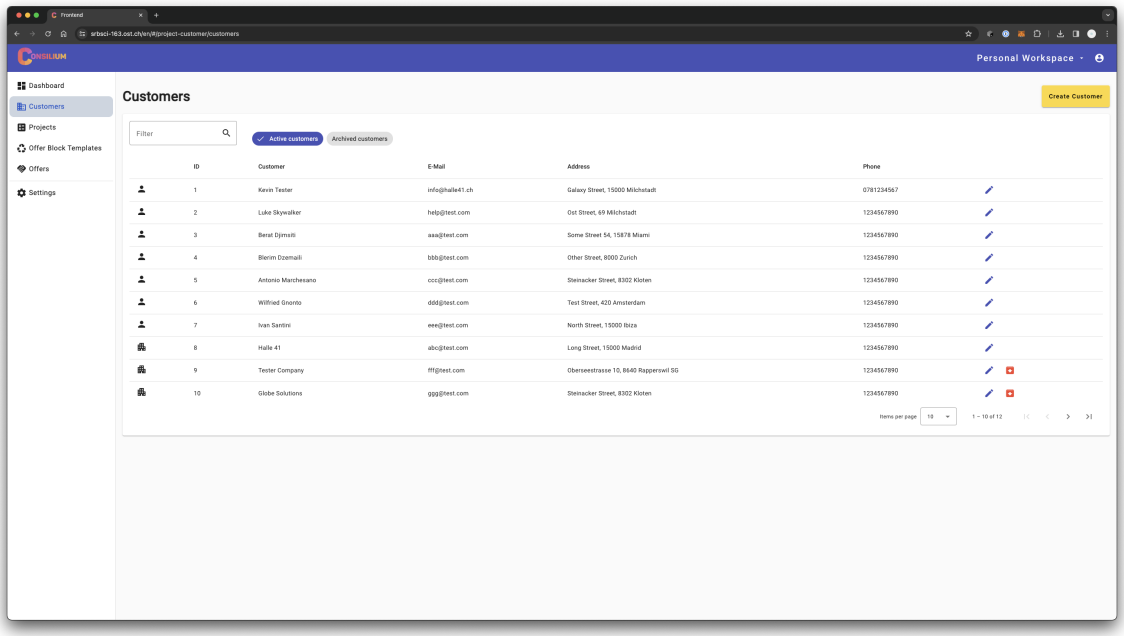

Figure D.15: Customers Now

Projects

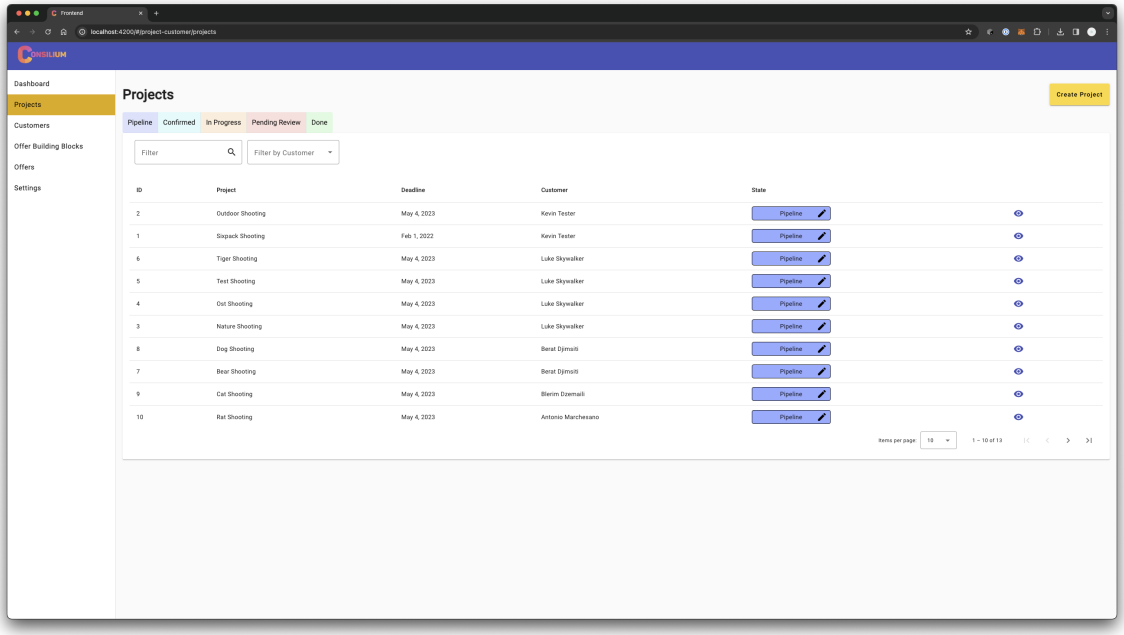

Figure D.16: Projects Prototype

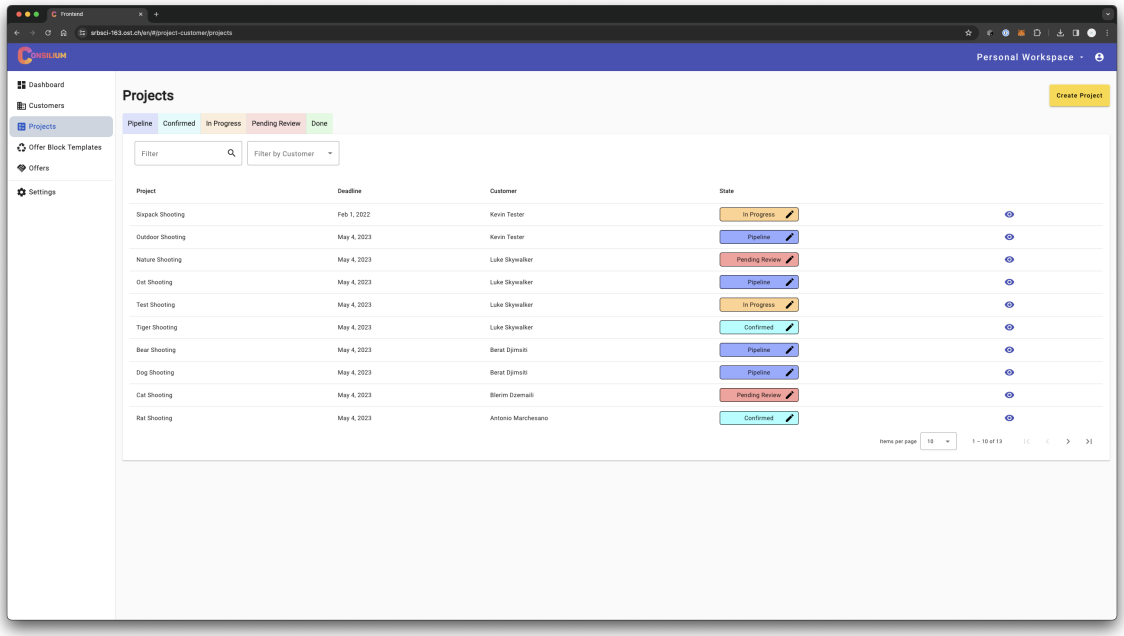

Figure D.17: Projects Now

Appendix E

README File

Consilium Developer Documentation

The following documentation should help developers to be able to help developing on consulium, to be able to adjust consilium to their needs, and to be able to deploy consilium onto any given environment.

Development

The following development guide convers setup instructions, some important code conventions, and some handy commands to keep you on track.

Setup

To start off you need to make sure to complete all the prerequisites. The application generally exists of two parts, a Spring backend and an Angular application, which you can find in the 'frontend' subfolder.

Prerequisites & Tools

- Java 17.
- Postgres' connection, see `startDatabase.sh`
- IntelliJ
- Maven

Very first to get your code compiling, you must generate all the JOOQ classes To achieve this, use the install command from maven.

- `mvn install`

After installing, you will be able to run Consilium right through your IntelliJ IDE. You'll have an existing configuration called 'ConsiliumApplication - Run'.

Note: The Backend must not be running while executing the command!

To run the frontend, first you must generate the node_modules directory. To do so, run `npm install`. Now you will be able to start the frontend by running `npm run start` or use the respective IntelliJ task.

As soon as the backend and frontend is running you can connect to http://localhost:4200. To access the application with test data, you can authenticate using the test user:

```
Username: abc.test@gmail.com
Password: Password1
```

Swagger generation

To generate the swagger interface to access backend calls, you must generate them. This happens by default also on the `mvn install` command, however for convenience, you can also call `npm run api` inside the frontend folder.

Note: The Backend must not be running while executing the command!

This will generate all REST calls into your frontend.

If a controller in the backend is called `ProjectController`, swagger will generate a respective angular service which is being called `ProjectControllerService` which can be found inside the `api` directory. The `ProjectControllerService` contains identical method names as the `ProjectController` does.

Conventions

Conventions which will ease the entry into the project for you.

DTO / Business conversion

Typically, DTO's (Data Transfer Object) will not go beyond the controller layer. As soon as you want to pass data from or to the service layer, you must convert them to business objects. By convention, we have decided to make this conversion inside the DTO.

public static XyDto from(RelativeBusinessObject ...) {}

public RelativeBusinessObject to() {}

Current user context

To get the context of the current user, you have several possibilities. If you just need the active workspace, you can do so by injecting the `WorkspaceService` and calling the `#currentWorkspace()` method. If you however need the current user, you can do so by injecting the `CurrentUser` into any service.

Permission checks

If you want to check if a user has Administrator permission, you can do so by annotating any REST endpoint with the `@AdminAccess` annotation.

If you need to check whether a user is able to access a set of data, you will find helper methods in the `WorkspaceService`. You will find methods such as `handleAccessToProject(int projectId)`. These methods will check if the user can access the given project for example and handle it otherwise. These methods should only be used in the service layer.

Commands

- `mvn clean install` → cleans the build and creates a clean new one
- `mvn install -DskipTests` → Skips the tests to be faster at building
- `mvn test` → Runs the tests
- `npm install` → generates the node_modules folder
- `npm run api` → Generates the swagger api in the frontend
- `npm run start` → Server the frontend on port 4200

Deployment

Consilium is being built as a single jar application, including the frontend. So wherever you want to deploy it to, it should not be a big problem.

Generally to deploy the application you must first build it.

Building Step-by-Step

- `npm run build` (In frontend)
- `cp -R frontend/dist/frontend src/main/resources/public` (In root) ***Important***: Public folder must not exist in resources directory before executing
- `mvn install -DskipTests`

Now your bundled jar file will be available under `target/consilium-version-SNAPSHOT.jar`

Local / Production Deployment

To deploy it locally or on a remote machine, just copy to jar file onto the given machine and start it. You can basically start it by typing `java -jar myfile.jar`. However, most of the time you will be required to change the active spring profile. This can be specified in the command line using:

`-Dspring.profiles.active=prod`.

Now for example if you might be using tmux, you can run consilium with the following command:

```
tmux new-session -d -s consilium 'java -jar -Dspring.profiles.active=prod consilium-0.0.1-SNAPSHOT.jar'
```

Cloud / AWS Deployment

To deploy your application to the cloud. You must get your self comfortable with the given solution you want to use first. We provide a guide to deploy the application on AWS. However, you can do this as well with other providers.

1. Create an account on aws.
2. Go to Beanstalk
3. Create application
4. Step 1, 'Configure environment'
    1. Give your application a name
    2. Select Managed Platform
    3. Select Java as Platform
    4. Select Corretto 17
    5. Upload your Jar file and give it a version
5. Step 2, 'Configure service access'
    1. Select Existing service roles your existing service role
6. Step 3, 'Set up networking, database, ...'
    1. Enable database
    2. Engine: postgres
    3. Engine version: 15.2
    4. Set username and password and remember for future step.
7. Skip step 4
8. Step 5, 'Configure updates, monitoring, and logging'
    1. Go to Platform software - Environment properties
    2. Configure: `SERVER_PORT`, `5000` This is the port used by AWS.
    3. Configure: `SPRING_DATASOURCE_PASSWORD`, `Password you set in step 3`
    4. Configure: `SPRING_DATASOURCE_USERNAME`, `Username you set in step 3`
    5. Configure: `SPRING_DATASOURCE_URL`, `Write any placeholder`
    6. Configure: `SPRING_PROFILES_ACTIVE`, `prod`
    7. Configure further information you want to adjust from the properties file.
9. Now your configuration is done and your Beanstalk should be running. As soon as your database is up and running, adjust the `SPRING_DATASOURCE_URL` environment property to the given address by AWS.

Troubleshooting

You might encounter role permission issues on AWS. You can fix them in the Role management console of AWS.

Configuration

To configure Consilium to your needs, all you can do is adjust the `application-prod.properties` file.

In that file you can basically adjust everything which spring allows you to. But these are the most important ones:

Mailing

To connect to your own mail account you have the following properties to adjust:

```
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=consilium.dev@gmail.com
spring.mail.password=password
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
```

HTTPS

You will be required to provide a certificate to enable HTTPS. To learn how to create such a certificate, take a look at https://letsencrypt.org/. The following properties will be relevant to be adjusted.

```
server.port=443
server.ssl.key-store=/etc/letsencrypt/live/srbsci-163.ost.ch/keystore.p12
server.ssl.key-store-password=password
server.ssl.key-store-type=PKCS12
server.ssl.key-alias=tomcat
```

Appendix F

External Feedback

F.1 Customer Feedback

We have received two pieces of feedback from once a friend of us, who started using Consilium, as well as a colleague of ours, who attended *Consilium* in the prototyping phase. Some of the feedback has now already been implemented in the mean time.

F.1.1 Feedback from Consilium customer

- Consilium ist ein sehr praktisches Tool mit vielen Möglichkeiten, Benutzerdefinierte Prozesse und Lösungen zu erfassen. In einer Branche die sehr unterschiedlich ablaufen kann ist Consilium mitunter einer der angenehmsten und agilsten Lösungen.

- Die Kombination eines Management-Tools welches nicht nur den Projektstatus und den Kunden überblickt, sondern zudem noch als Tool zur Erstellung von Dokumenten und beim hinterlegen von Verträgen helfen kann ist enorm praktisch.

- Als Freelance Filmproduzent bin ich aktuell gerade im Aufbau und der Erweiterung meiner Firma - mit Consilium habe ich eine sehr nützliche Software welche mir die Kommunikation, Organisation und die Skalierung meiner Firma erheblich erleichtert.

- Ich bin sehr gespannt auf zukünftige Ausbauten und sehe viel Potential im beifügen von mehr Funktionalitäten und Optionen im Tool.

F.1.2 Feedback from previous team member

Die Consilium App finde ich ein sehr nützliches Tool für Projektmanagement. Man kann Projekte erfassen und auch unterschiedliche Arbeitsbereiche.

Diese könnten aber noch etwas besser erklärt werden. So wie ich verstehe dient es zur Trennung diverser Projekte. Kunden, etc. gehören auch zu einem Bereich und man kann in teilen mit mehreren Usern. Mir gefällt auch die zweisprachigkeit der Webseite. Ganz toll finde ich die Möglichkeit dynamisch Offerten anzulegen mit den Bausteinen.

Beim Erstellen des Kunden finde ich den mehrteiligen Schritt gut. Doch hier wäre ich toll einen klick sparen zu könen idem der Schritt Fertig nicht mehr gibt und stattdessen automatisch speichert.

Bei den Projekten selbst finde ich die ausführliche Zeiterfassung gruppiert in diversen Buckets mit Sollzeiten sehr gelungen. Doch will ich gerne (solange noch nicht Verrechnet) diese auch bearbeit-

en/löschen können, was aktuell nicht geht. Laut Dashboard sollte es auch eine Notizfunktion mit Deadline geben, doch diese ist auf der Projektseite nicht zu finden. Notizen zu einem Projekt wie Meeting-Protokolle, etc. wären schon noch praktisch.

Weiter wären weitere Statistiken zur Arbeitszeit toll. Aktuell ist ja nur möglich auf Monat gefiltert diese anzuzeigen, statt per Bucket (Arbeitszeiterfassungskategorien).

Im Ausblick wäre toll, wenn es eine möglichkeit gäbe seitens Kunde die Offerten online einzusehen (evtl. direkten Mailversand anbieten). Der Kunde könnte dann die Offerte digital bestätigen oder ablehnen und evtl. unterzeichnen. So wäre der Statuswechsel automatisiert möglich. Rechnungen sollten allenfalls seitens Kunden auch automatisch via Onlinezahlung begleichbar sein. Was wichtig ist ist, dass auf der Rechnung eine Schweizer Rechnungs-QR-Code vorhanden ist (Nicht verifizierbar, da derzeit keine Rechnungen generiert werden).

Zwecks einheitlichkeit wäre im Menu ein punkt mit allen Rechnungen wohl ebenso ganz praktisch.

Die Offer-Blockforlagen sollten in mehreren Workspaces verfügbar sein und nicht nur in einem. Oder zumindest optional kopierbar sein. Dies falls jemand seine Projekte in diverse Workspace stukturiert, aber gleiche Blöcke haben will.

Das Login ist toll das es per E-Mail und passwort möglich ist. Ob jetzt ein externer Auth-Anbieter oder selbst implementiert lässt sich darüber streiten, vorallem da das UI nicht matcht und das Logo der App fehlt.