

Bachelor Thesis

Order Management Tool für 3D-Druck Dienstleistungen

Semester: Spring 2024



Project Team: Joel Sauvain
Noah Stalder

Project Advisor: Frieder Loch

School of Computer Science
OST Eastern Switzerland University of Applied Sciences

Abstract

When a startup is founded, the focus is usually put on getting orders instead of implementing an adequate order management solution. As the startup grows in orders and employees, the initial make-do solution may become insufficient and slow the business down in the long run. This was the case for our partners CHANGE3D, who provide various services in the 3D-printing environment. Their make-do solution consisted of printing out cards for each order. The printed cards were then filled in with details such as order number, title, and client name, and ultimately hung on a whiteboard. As their business grew, the whiteboard became too full, making it more difficult to find orders and keep up with deadlines. This challenge led to the realization that a new solution is needed.

To solve the problems and alleviate the pain points of our partner, we set out to develop a tool that displays their orders on a kanban board. In close collaboration with our partners, we followed the principles of user-centered design that enforces an iterative process and focuses on an understanding of the users and their context in all stages. This enabled us to gain a deep understanding of their problem and allowed us to build an application tailored to their needs. Through a scheduler, written as a microservice in Kotlin, we integrated their Bexio where we fetch the orders from, to eliminate the necessity of manually creating an item for the board. The core of the application is a Backend written in Kotlin, which receives the orders from the scheduler and stores them in a MongoDB. Additionally, the states and state transitions are managed in the Backend, which automatically triggers actions on Bexio, further eliminating manual steps of the previous process. The Frontend is written in Angular, communicating with the Backend through a RESTful API, which allows for live updating using SSE.

The result provides a substantive improvement for the day-to-day business and order management of our partner, as determined by user tests. The core functionality is the board, which responsively displays the items in columns representing the states. Through drag and drop, an item can be moved into another state, which triggers the applicable action on Bexio. To improve the usability of the board and make it more practical for more use cases, there are various view options implemented, such as sorting, filtering, and grouping the items. On top of that, what is shown on each card is configurable and pieces of information can be hidden to further remove clutter from the board. Additional information can be stored and found on the detailed view of each card, such as assigning the item to someone, the history of the item, or an interactive 3D render of the piece that is being produced. After an item has gone through all the steps and the order is completed, it can be moved into the archive, which provides an insight into the orders that were completed across a timeline.

Management Summary

Context

The 3D-printing service provider CHANGE3D was in need of a digital order management solution to visualize the states of orders and entrusted us, Noah Stalder and Joel Sauvain to design and develop it.

Our partners had a make-do solution in the form of a physical whiteboard with cards on it to represent the orders in place which was already starting to get flooded with all the orders they were getting in. When we first got there and got an overview of the situation, it was apparent their solution was outdated and could not keep up with the growth they were and are enjoying. One approach would have been to buy a bigger whiteboard, but we are glad they chose a more future-proof solution, in commissioning a digital solution.

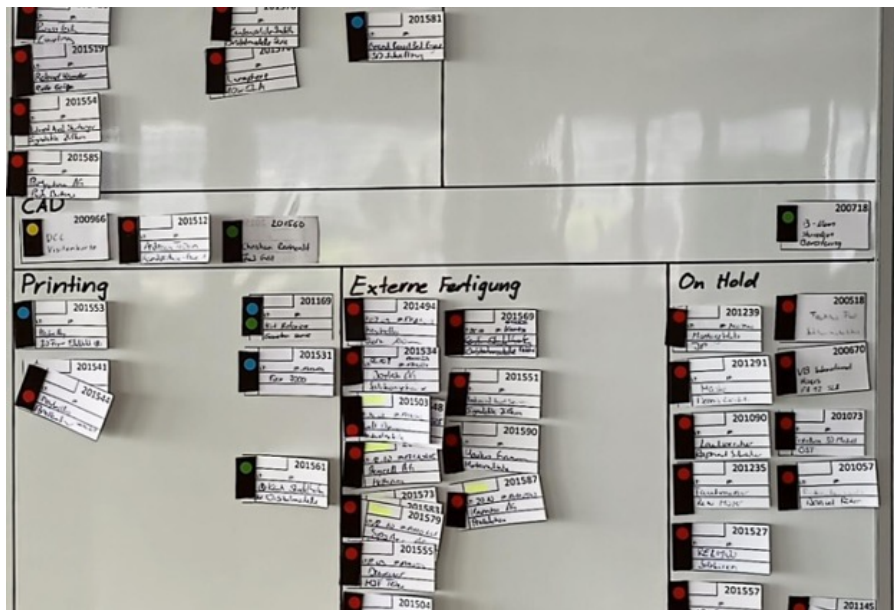


Figure 1: Physical Order Board

Until now, the order board they worked with, had separated areas to represent the states of each order. The cards they put on the board for each order, contain various pieces of information like the order number, title, and the client. The colors represented which member of the team was responsible for the order.

Other indications like the due date could not be put on the cards, and so the deadlines became almost impossible to keep track of. Another weak spot of this solution was that dynamic information which changes during the completion of the order could not be put on the cards without making a mess. As they offer a wide range of services, ranging from consulting to printing, external production, and construction, they had to introduce a mix between the type of order, the technology, and the state onto the board which made it increasingly harder to understand.

From a technological point of view, they manage their offers, orders, and invoices through Bexio and store their files in OneDrive.

Approach & Technologies

Our approach to solving the problem and relieving the pain points of our partner, while introducing various improvements to their process, was to design and develop a dynamic kanban board, displaying the orders. We chose to work in close collaboration with the partner, following a user-centered design process which gives us a deeper understanding of their context and problems and ultimately helps us develop a fitting solution to their problem. A big part of the process is to challenge our and their assumptions to eradicate potential wrong assumptions that would limit the solution.

With an emphasis on an extensive requirements engineering and design process, we held workshops together with the partners, drawing up potential design variants, debating them and alternative solutions, discarding unfitting designs, building up on promising ones, and choosing fitting ones.

We wrote a scheduler microservice in Kotlin to integrate Bexio by periodically fetching the offers, orders, invoices, and deliveries, which are then aggregated and mapped into the different states we want to represent. The scheduler passes the items to our core Backend, which is also written in Kotlin, which then enriches the data set and saves it in a MongoDB. Our core Backend is also responsible for creating the folder structure for each incoming item in their OneDrive. On top of that, state transitions are managed in the Backend, which automatically triggers actions on Bexio, further eliminating manual steps of the previous process.

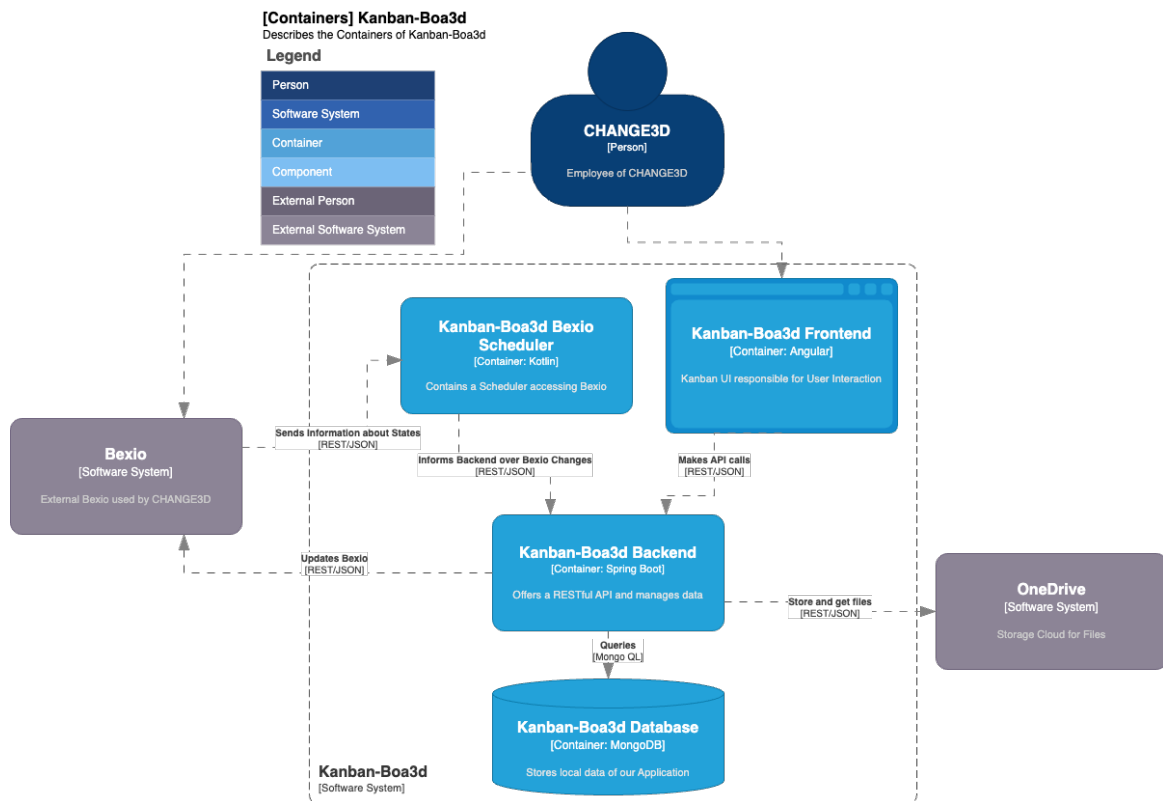


Figure 2: C4 Container Diagram

The frontend is written in angular and is responsible for displaying the items on a board and allowing the user to modify items and moving the items between states through drag & drop. Communication between front and Backend is done via a RESTful API which also allows for live updating using SSE.

Results

Based on the feedback of our partners and as a result of the exploitative user tests we can call this project a success. The resulting tool managed to address the initial problem fittingly, introducing substantial improvements compared to their previous solution.

The heart of the finished tool is the board, which displays the orders as cards across columns representing the different states:

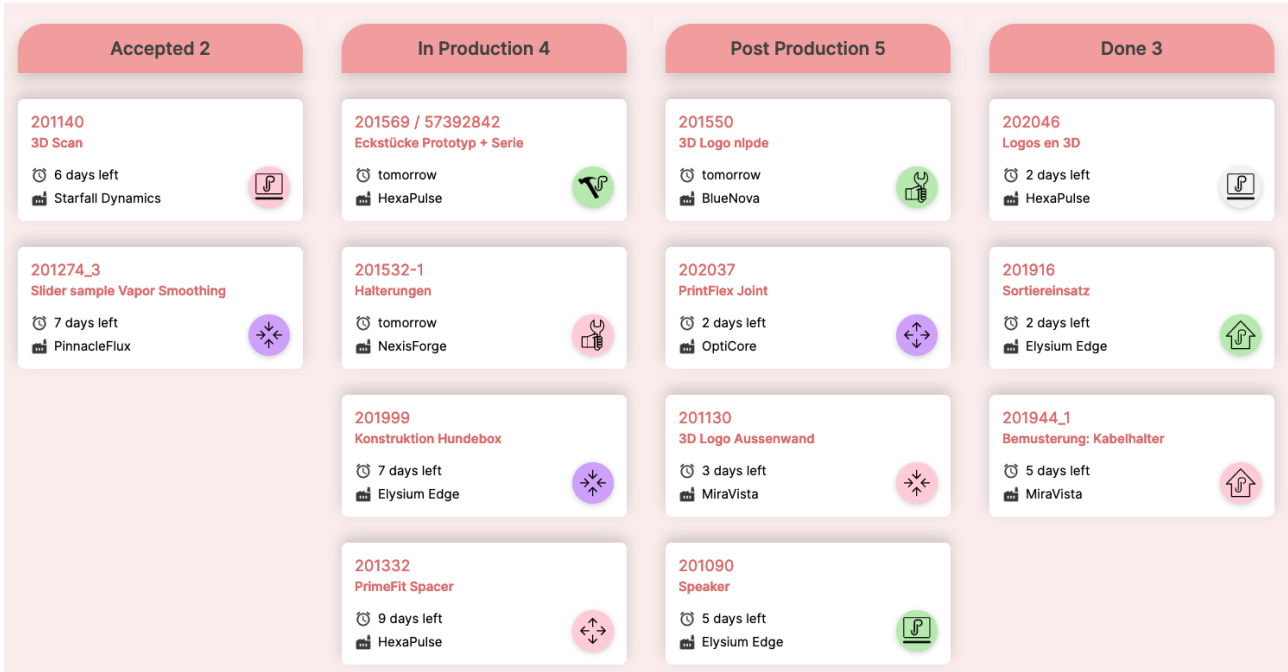


Figure 3: Board Cut-Out

The card design includes the order number and the title as well as the days left until the deadline and the customer. Additionally, we created icons for each type of service they offer which are displayed inside a sphere. The color of the sphere represents which user is currently assigned to the item.

The board offers a wide range of view customization options like sorting mechanisms and filters but also allows the items to be grouped by assignee or item type. Additionally, the viewer can hide pieces of information from the card, only showing the pieces of information he is interested in.

In the detail view, users can make changes to the details of an item. Among other things, the users can add a description, set and change the assignee, and view the history of an item.

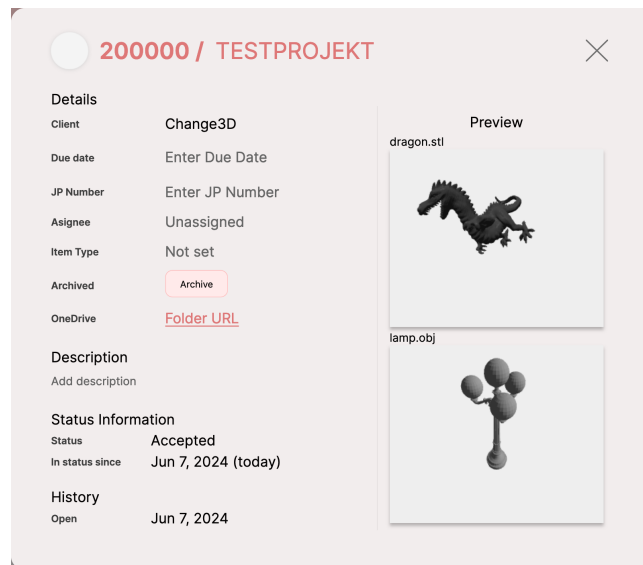


Figure 4: Detail View

On top of the information displayed, the 3D render of the object that should be printed is displayed to help associate the order to the object. When an item has run its course, it can be moved to the archive, where all completed orders are displayed across a timeline.

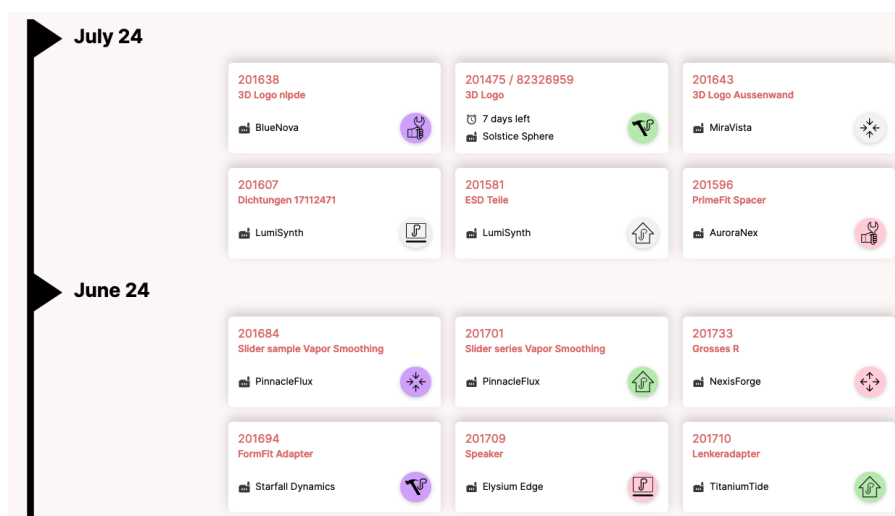


Figure 5: Archive Cut-Out

With those and more features, we managed to exceed the initial project definition, which only included a running prototype as the running project could successfully be deployed and our partner is already working with the new solution. As written in the testimonial by our partner, we were able to substantially improve their efficiency by automating a lot of production processes making tiring manual labor a thing of the past, and enabling them to spend more time on more important tasks. Our solution is already indispensable for their day-to-day business and far exceeded their expectations.

Acknowledgements

First of all, we would like to thank our advisor Frieder Loch, for his support and his insights. No question went unanswered and his web engineering expertise, creative ideas, and suggestions were invaluable to the success of our project.

Furthermore, we would like to extend our gratitude to Markus Grimm and Kevin Seliner at CHANGE3D with whom it was a pleasure to work with. From the start, we felt welcomed and could feel the trust they put in us. They helped us gain a deep understanding of their problem and context and provided invaluable inputs and feedback in the workshops, sprint plannings, and user tests.

Contents

1	Introduction	1
1.1	Problem	1
1.2	Project Definition	2
1.3	Context	3
2	Analysis and Requirements	4
2.1	Requirements Analysis Process	4
2.2	Requirements Analysis	5
2.2.1	Kickoff Meeting	5
2.2.2	Requirement Analysis and Design Process	6
2.2.3	Design Iteration 1: Card Design and Arrangement	6
2.2.4	Design Iteration 2: Finalization Card Design and Archive	15
2.3	Requirements	20
2.3.1	Functional Requirements / User Stories	20
2.3.2	Non Functional Requirements	22
2.4	Market Analysis	24
2.4.1	Jira	24
2.4.2	Miro	25
2.4.3	Summary	27
3	Solution Design	28
3.1	Architecture overview	28
3.1.1	Kotlin over Java	30
3.1.2	Spring Boot as our Backend	31
3.1.3	Database Selection	32
3.1.4	Logging	33
3.2	Security Considerations	33
3.2.1	Problem	33
3.2.2	Future Considerations: Implementing an IAM System	34
3.2.3	Conclusion	34
3.3	Live Board Solution Design	35
3.4	Bexio Transitions Solution Design	35
3.4.1	Accepted State Transition	36
3.4.2	Done State Transition	36
3.5	OneDrive Integration Solution Design	37
3.5.1	Creating Folder Structure for New Projects	37
3.5.2	Uploading PDFs for Accepted or Done State Transitions	38
3.5.3	Querying and Rendering 3D Files in Project Detail Card	39

4	Implementation	41
4.1	Communication between Services	41
4.2	Deployment	42
4.3	Bexio Communication	44
4.3.1	Authentication at Bexio	44
4.3.2	Secure Management of JWT in Deployment and Development	44
4.4	Bexio Data Export and Transformation	44
4.4.1	Relevant Endpoints	44
4.4.2	Filtering Algorithm	45
4.5	Bexio Updates	45
4.5.1	Implemented Solutions for Document Number Management	45
4.5.2	Persistent Challenge with Shipment Creation	46
4.5.3	Future Outlook	46
4.6	SVG generation for Archive	46
4.7	Integration of OneDrive	47
4.8	3D Model Visualization Integration	48
4.8.1	Integration Process	48
4.8.2	Challenges and Solutions	48
5	Results	50
5.1	Results	50
5.1.1	General	50
5.1.2	Board	52
5.1.3	Detail View	57
5.1.4	Archive	59
5.1.5	Full Screen	60
5.1.6	Interactions	61
5.1.7	OneDrive Integration	62
5.1.8	Differences to Design and Requirements	62
5.2	Summary and Outlook	64
	Bibliography	65
	A Project Definition	67
	B Testimonial	70
	C Slides First Design Iteration	71

List of Figures

1	Physical Order Board	iii
2	C4 Container Diagram	iv
3	Board Cut-Out	v
4	Detail View	vi
5	Archive Cut-Out	vi
1.1	Physical Order Board	1
2.1	Initial Card from Customer	7
2.2	Colored Type Icons	7
2.3	Initial Card Design (extended option)	8
2.4	Initial Card Design (minimal option)	8
2.5	Simple Columns Arrangement	9
2.6	Simple Columns Grouped by Assignee	10
2.7	Double Filed Columns	11
2.8	Simple Rows Arrangement	12
2.9	Island Arrangement	13
2.10	Detail View	14
2.11	Quantified Detail View	15
2.12	Before vs After: Finalized Card Design	16
2.13	Full Page Layout	18
2.14	Archive Design	19
2.15	Jira Kanban Board	24
2.16	Jira Custom Columns	25
2.17	Miro Kanban Board	26
2.18	Miro Item	26
3.1	C4 Context Diagram	29
3.2	C4 Container Diagram	30
3.3	Interaction Diagram: Server-Sent Events	35
3.4	Sequence Diagram: Accepted State Transition	36
3.5	Sequence Diagram: Done State Transition	37
3.6	Sequence Diagram: Creating Folder Structure	38
3.7	Sequence Diagram: Uploading PDFs	39
3.8	Sequence Diagram: Querying and Rendering 3D Files	40
4.1	GitLab CI Configuring Variables	43
4.2	Archive Beam implementation	47
5.1	Header bar	50

5.2	Settings	51
5.3	User Color Picker	51
5.4	Board	52
5.5	Scrolling Behaviour Board	53
5.6	Card on Board	53
5.7	View Bar	54
5.8	View Settings Expanded	54
5.9	Board Grouped-By Assingee	55
5.10	Board Grouped-By Item Type	55
5.11	Filtered Board	56
5.12	Board with Minimal Cards	56
5.13	View Settings Expanded: Urgent View	57
5.14	Detail View	58
5.15	Detail View with Preview	59
5.16	Archive	60
5.17	Full Screen	61
5.18	Archive Shortcut	62

Chapter 1

Introduction

This chapter gives an overview of the character of this thesis and narrates based on the information available, before starting to work on the project.

1.1 Problem

This thesis aims to provide a product that improves the day-to-day management of orders for a 3D printing service provider. The client is willing to expand and identified their current solution for their order management as a bottleneck which slows them down.

Currently, they receive their orders from *Bexio* where they manually print out little notes and write additional information on them. They use a physical whiteboard to maintain an overview of their order (see Figure 1.1). The whiteboard consists of different areas that represent the different states of an order.

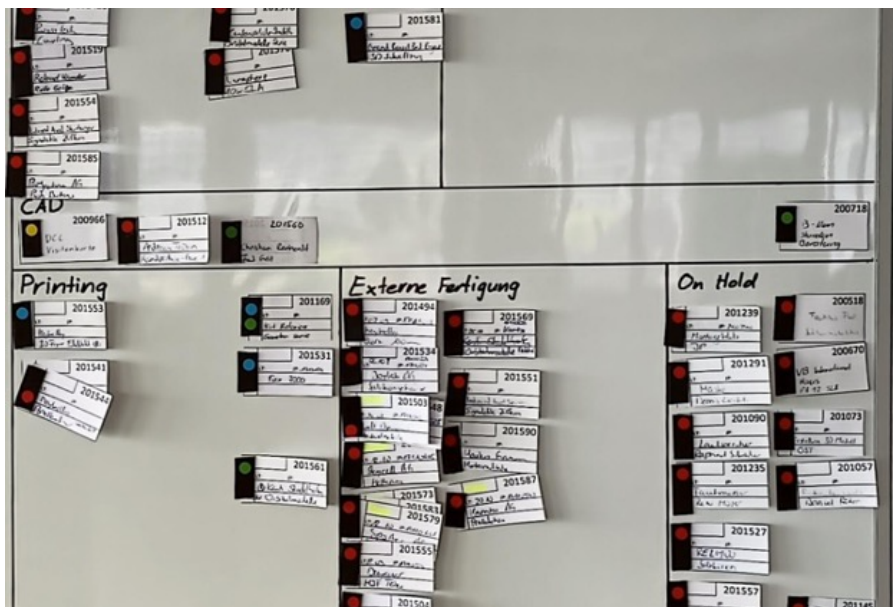


Figure 1.1: Physical Order Board

This board shows the different states an order can be in as well as how the cards currently look. Each

card displays the order number and varying items of additional information. The color eclipses indicate the responsible team member for each order.

This way of working slows them down significantly as every new order that comes in, requires manual labor to prepare a card and place it on the physical board. Additionally, it made it hard to work with deadlines as there is limited space on the cards, and rearranging them based on the delivery date is tiring work too. If there were additional remarks that might be helpful for someone working on the order, there was no space to place them either. The billing process was also more challenging, as it is hard to see which orders have already been invoiced and which ones have not.

1.2 Project Definition

This section gives an overview of the project definition, states the framework conditions, and aims to provide further insights into the character of this thesis.

The current solution and its problems highlighted in section 1.1 led to *CHANGE3D* contacting OST for help which ultimately led to their problem becoming the topic of our Bachelor Thesis. As our Bachelor Thesis, this project has a required effort of 360 hours per person, granting 12 ECTs, and is done in a team of two.

Our challenge was to improve their workflow by developing and introducing a new web Frontend which is designed around relieving their key pain points while integrating seamlessly into their existing infrastructure.

This means, our tool should integrate Bexio to consume the new orders and help our client manage them. Our tool should include a Kanban board that models the orders as cards and allows for them to be moved via Drag and Drop to manage the orders and their respective states. Ultimately, our task is to implement a Proof-of-Concept while being free to choose whichever technologies we want.

Our thesis starts with a requirements analysis with the client where the technical ramifications as well as the processes are established. A market analysis, comparing our tool against existing options will also be part of the thesis. Based on the requirements, fitting technologies should be selected. The whole project is meant to be done in a user-centric way, which includes working together with the customer to get real-world, real-time feedback.

The full project definition in German can be found in Appendix A.

1.3 Context

This section aims to provide an overview of the system context we are working within as well as give some insights into the domain-specific details of them as 3D printing service providers.

As a 3D printing service provider they offer a wide range of services. They offer services such as printing and construction but also consult clients in this field. Various printing technologies require different types of printers. Depending on what kind of object should be printed, different technologies are best suited. *CHANGE3D* has printers for one technology in-house which they print themselves but offer services for other technologies, for which they outsource the production to partners.

They get their orders through *Bexio*, where they create their offers and handle their accounting. In *Bexio*, they input the customer and order details, create an order and manually create a folder on their file system. Through an Excel sheet, they calculate the prices and send out the offer. Upon acceptance of an order, they send out the confirmation and put it in the created folder. For out-sourcing orders they can not do in-house, they use the platform *Jellypipe*. Once an order is printed and completed, they package it up, print out the shipping etiquette, and ship it.

Chapter 2

Analysis and Requirements

This chapter goes through the analytical process and why it was chosen, summarizes the requirements it generated, and compares our solution to existing solutions on the market through a market analysis.

2.1 Requirements Analysis Process

Our goal is, to understand the problems and pain points of our clients as well as possible so we design and implement the best solution for them we can. With that in mind, we decide to work closely with the client gathering their input, coming up with solution concepts, and discussing those with them. We chose to not have separate requirements engineering and design processes and instead merged them and applied the user-centered design methodology to profit from the synergies of both processes. We based our user-centered design approach on the article [1].

By doing so we will generally split our process into iterations which consist of the following four phases:

1. **Understand the context of Use:** In the first phase we use their existing solution as a baseline to understand how they work. Additionally, through workshops and meetings, we try to further identify ways to improve their workflow.
2. **Specify user requirements:** In this step, the requirements are loosely defined as a basis for the design.
3. **Design solutions:** In this step, we create the designs to fulfill the requirements. In most cases, this includes a real design of how the feature would look on the page. Usually, there are multiple variants, as there is always more than one way to solve a problem.
4. **Evaluate against requirements:** In the last step of the iteration, we get together with the client again present the solutions, discuss variants, and select what will be kept in place and what will be worked upon again in the next iteration.

Another advantage we gain by doing that it is way easier to achieve a good match between the System and the Real World which is the second heuristic highlighted by [2]. This improves the user experience, as a user will recognize the terms used in the tool from the real world. Other than getting a better feeling for what the users want we can also much better determine what they do not want which prevents us from implementing features they will not need. On one hand, this allows us to invest our time in more features, and on the other hand, it helps us achieve heuristic eight Aesthetic and Minimalist Design

from [2], which states that irrelevant extra information diminishes the relative visibility of relevant units of information.

2.2 Requirements Analysis

This section shows how we followed the process we chose in section 2.1. The goal of this section is to not only show the finished designs and established requirements but also give insights into how they were generated,.

2.2.1 Kickoff Meeting

To achieve our goals we saw a meeting with our client as the optimal first step. We wanted to talk to them face to face to get a feeling for who they are, how they work, what their surroundings are, what their pain points are, and how we could best involve them in the process to achieve the best possible outcome. By sitting down together and understanding where certain requirements or ideas come from, we hoped we would be able to comb out incorrect assumptions on our and their part. When we understand the reason for certain requirements we can come up with alternatives that solve the same problem even better.

So in the first week of working on the thesis, we set up a Kickoff meeting and were invited to the *CHANGE3D* office space. There they showed us how they currently work with their physical board and they could further elaborate the limitations of their existing solution. They then presented us with an initial set of must-haves in our digital solution and listed some nice-to-haves as well which we discussed and modified together. This resulted in the following initial must-haves which will be later further elaborated on and established in more detail:

- Display all orders as a Kanban board
- Model and display order process from start (offering) to finish (shipping).
- Status of the order cards is modifiable through Drag&Drop
- The information that needs to be displayed on each card
- Integration of Bexio
- Must support various devices
- Must contain dynamic items that can be extended.

Another key outcome of the meeting was, that we established together, how we will structure our collaboration together and mainly how we want to involve them in our design process and generate the requirements together.

2.2.2 Requirement Analysis and Design Process

In the kickoff meeting, we agreed to work closely together especially during the design process, to get their opinions and feedback on our suggestions and variants, to get a result best suited to their needs. We will involve them in our scrum process to regularly and iteratively get their inputs. During this project, we will put the user interface and design first, by starting with creating screens, which we will then discuss with the customer and work on in iterations to shape the end product together. After contemplating using mock-ups for the pages, we decided to use actual page designs. This decision was based on the fact, we only have a few pages and the content on those pages is what we want to prioritize. Mock-ups would have been a better fit if we were designing a tool with multiple pages and complex flows. The idea is, to draft up variants, discuss them, make initial observations together, and decide which variants should be kept for further consideration and which ones can be discarded in iterations. Each variant does not need to be perfected to be presented and discussed, instead the chosen variants will be refined and perfected in the following iterations. Through this approach, we hoped to have a clearer vision of the complete product early on and avoid the risk of our thesis resulting in a half-baked end product that is not well thought through. When working with designs instead of developing the user interface step by step, we can also make changes quicker with a fraction of effort, draft variants better, and react faster to inputs and feedback.

After looking for different tools to create the screens we landed on *Figma*. We previously evaluated *InVision* as well, but what convinced us of *Figma* was mainly the great material they offer for beginners. This was a key point for us, as we want to get off the ground quickly and do not want to spend the first few weeks of the project, learning a new tool. Another factor that steered us in the direction of *Figma* is the community and the extensive resources that are available. On top of that, we heard from people, whom we have worked together with, having found great success with this tool.

Once we agree on the designs and functionality of the tool with our customers, we will then use them as input to define the requirements. With this approach providing a clearer vision of how the product should look in the end, early on, we give ourselves a much better basis for our selection of technologies and architecture.

2.2.3 Design Iteration 1: Card Design and Arrangement

In the first iteration, we focused on designing the cards and drafting up possible ways we could arrange them on the Kanban board in their respective states. We decided to start with those items, as we view them as the heart of the application, and by starting early we have the most time to improve them gradually. The board needs to give a clear overview of the state of things and provide key information as well as possible.

Card Design

A card represents an order that can go through different states during its lifetime. The key challenge when designing this card is, to provide the most important information while not becoming too packed and crowded as that would make finding key information harder and could make the whole page look overloaded.

To start off with, the client provided us with an example of how a card could look like and what information they need on it, which is shown in Figure 2.1:

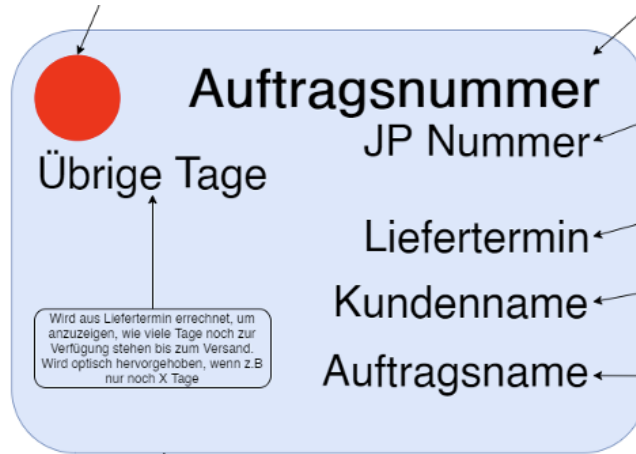


Figure 2.1: Initial Card from Customer

They stated that the color of the card should represent which type of order it is and the color of the sphere, which is red in this example, should indicate who is responsible for the order. Of the additional information, the order number (Auftragsnummer) is the most prominent but the, JP Number, the remaining days (Übrige Tage), the delivery date (Liefertermin), the name of the client (Kundenname) as well as the order name (Auftragsname) should also be displayed.

Initially, we saw that this already made the card quite full and loaded while the colors and their combinations when displayed on the board could be confusing. To tackle that potential problem, we came up with the idea to create icons for each type of order and then give them a background color, which represents who is responsible for the order, which is shown in Figure 2.2.

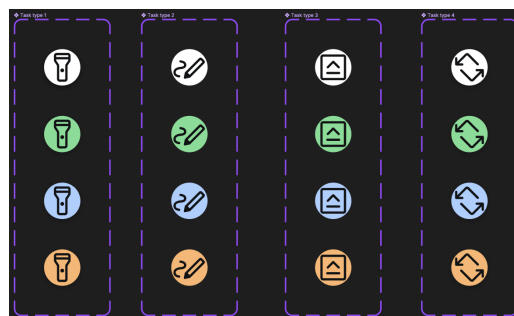


Figure 2.2: Colored Type Icons

Through this change, we hoped to reduce the noise on the full page while still providing the same information.

From there we created two versions of potential card designs which vary in detail. The first one is more extended and provides more information which is displayed in Figure 2.3.

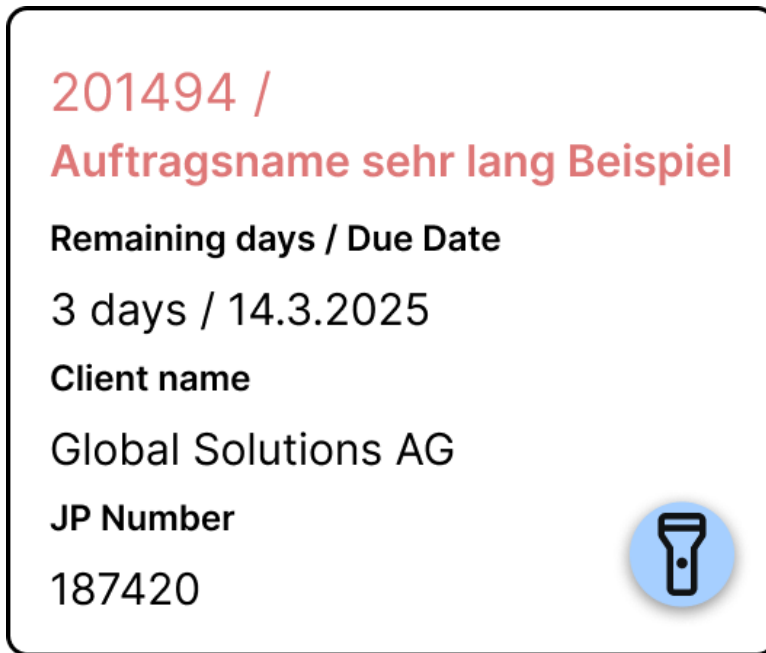


Figure 2.3: Initial Card Design (extended option)

This design option displays all the stated items while explaining what each field means to help someone new understand the tool.

The other option we came up with is more minimalistic and displays less information and is shown in Figure 2.4.

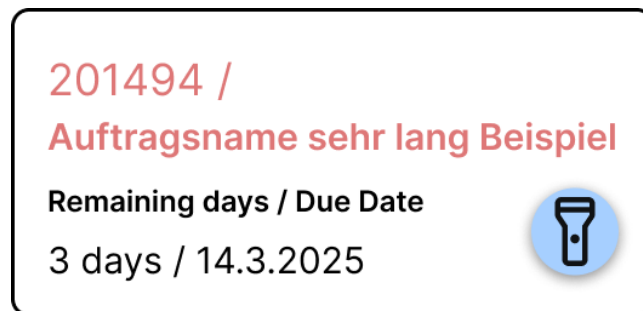


Figure 2.4: Initial Card Design (minimal option)

This card design focuses on not being too crowded by only providing minimal information. We identified the remaining days as a key element, as they have a hard time with their current solution to keep track of what needs to be done by when.

Card Arrangement

How the cards will be arranged provides the structure for the whole board and it is therefore instrumental to look at different possibilities and to land on the best fit. Additionally to the key improvement we want to provide for them to get and maintain a better overview, we also want to provide a way for them to customize the views so they can see specific cards at the time while hiding clutter. On top of providing the best possible overview for the whole team, we also wanted our tool to assist each user on its own while working on the orders as well. Therefore we wanted the Kanban board to allow the user to customize the view to best present the information he is looking for.

For the arrangement possibilities we defined different qualities we are looking for to measure the different approaches. First up, an efficient use of space is important to provide as much space for cards without overflowing. We want to prevent any unnecessary scrolling while seeing vertical scrolling as less of an evil than horizontal scrolling. The arrangement also needs to model the states which the cards go through and should therefore mimic the flow accordingly. We also added how well each option would be suited for features like sorting, grouping, and filtering to the equation.

Card Arrangement Variant 1: Columns

The first option, which is also the arrangement found in most comparable tools, is columns. With columns, each column represents a state and the cards move from left to right. Figure 2.5 shows our initial layout using columns:

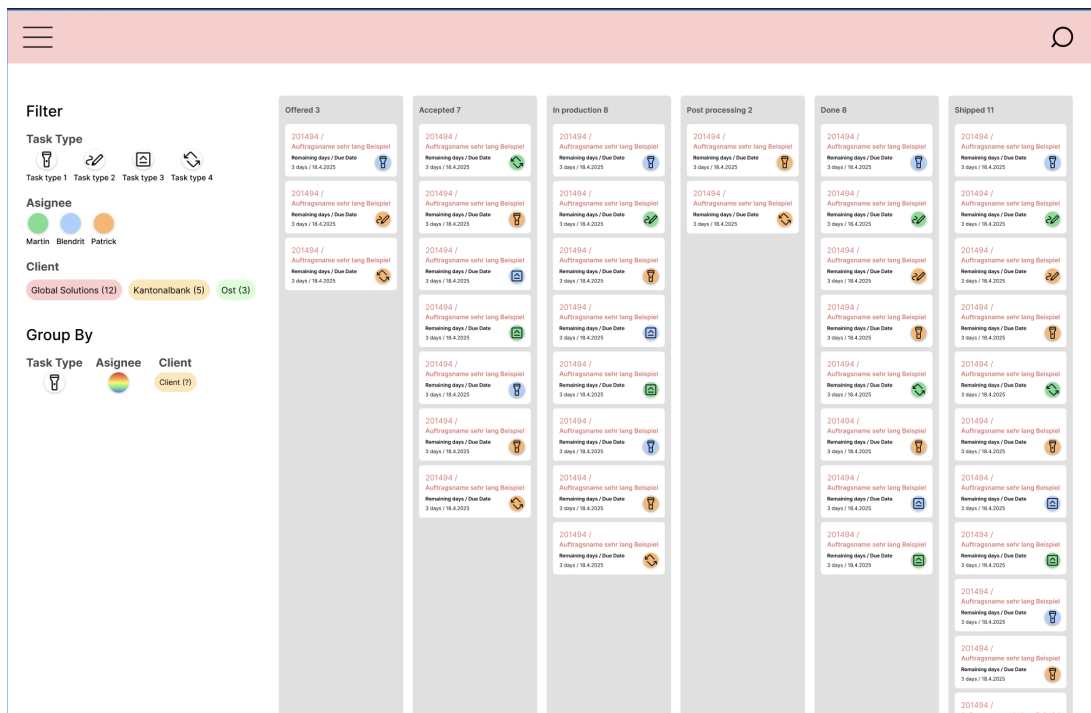


Figure 2.5: Simple Columns Arrangement

The use of columns is simple and easy to adapt to for users as "Jakob's Law" states "Users spend most of their time on other sites, and they prefer your site to work the same way as all the other sites they already know." as cited in [3]. They work well with small, wider cards as the height of a card determines how many cards fit into one column. Simple columns are also not complex and keep the look of the page clean.

A downside is the usage of space, as a column that only contains a few cards, takes up the whole space leaving much empty space, as illustrated by the "Post Processing" column which only has two entries. This is not strictly a bad thing though, as some empty, or more precisely negative space, can be refreshing and make the page not look too packed as stated by [4] by saying "All good visual artists understand the importance of negative space, the empty area that draws attention to, and accents, the actual subject."

However, as the goal is to prevent the need to scroll as much as possible, imperfect space usage, could be counterproductive. They also work well with additional functionality like filtering or sorting and group by, which is illustrated in Figure 2.6 where the cards are grouped by the assignee.

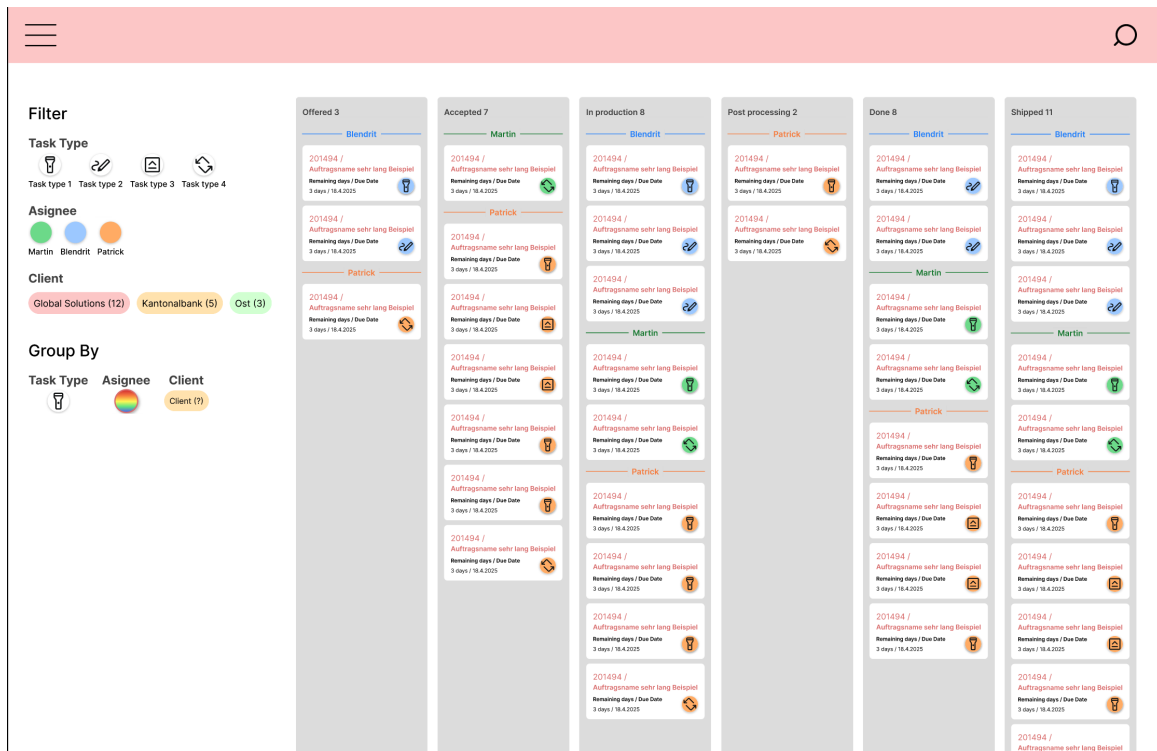


Figure 2.6: Simple Columns Grouped by Assignee

The grouping of the cards by assignee can be done very cleanly, as through single file columns, an exact line can be drawn to separate the issues. Additionally, cards are separated horizontally, which allows to add text without taking up too much space. It allows users to see their own tasks quickly while still maintaining an overview of all orders.

To improve on the space usage of simple single-filed columns, we introduced an adapted variant that splits the fullest columns into double columns as shown in Figure 2.7.

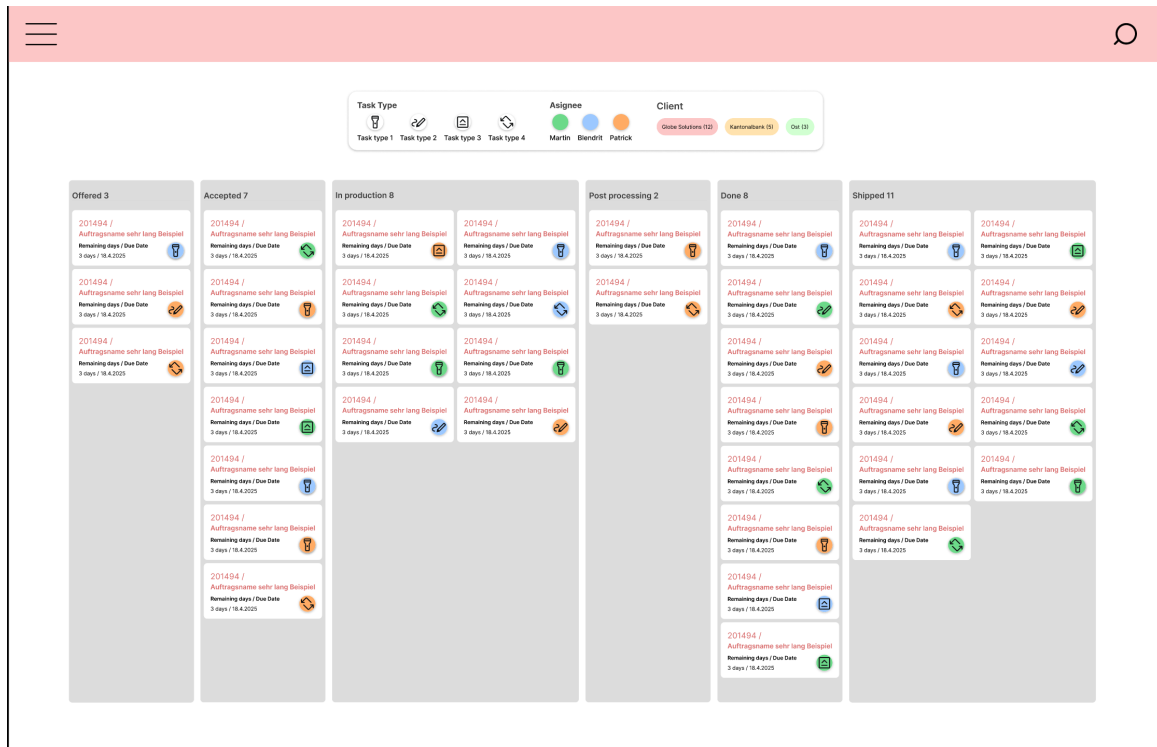


Figure 2.7: Double Filed Columns

In this example, it leads to "In production" and "Shipped" being split into a double column. This variant could either split the fullest columns or predefined columns into double or even multi-columns. This improves the usage of space and enhances which columns contain the most cards. It could however become more tricky with grouping and ordering.

Card Arrangement Variant 2: Rows

The second variant we considered and made designs for is using rows for the different states which Figure 2.8 displays.

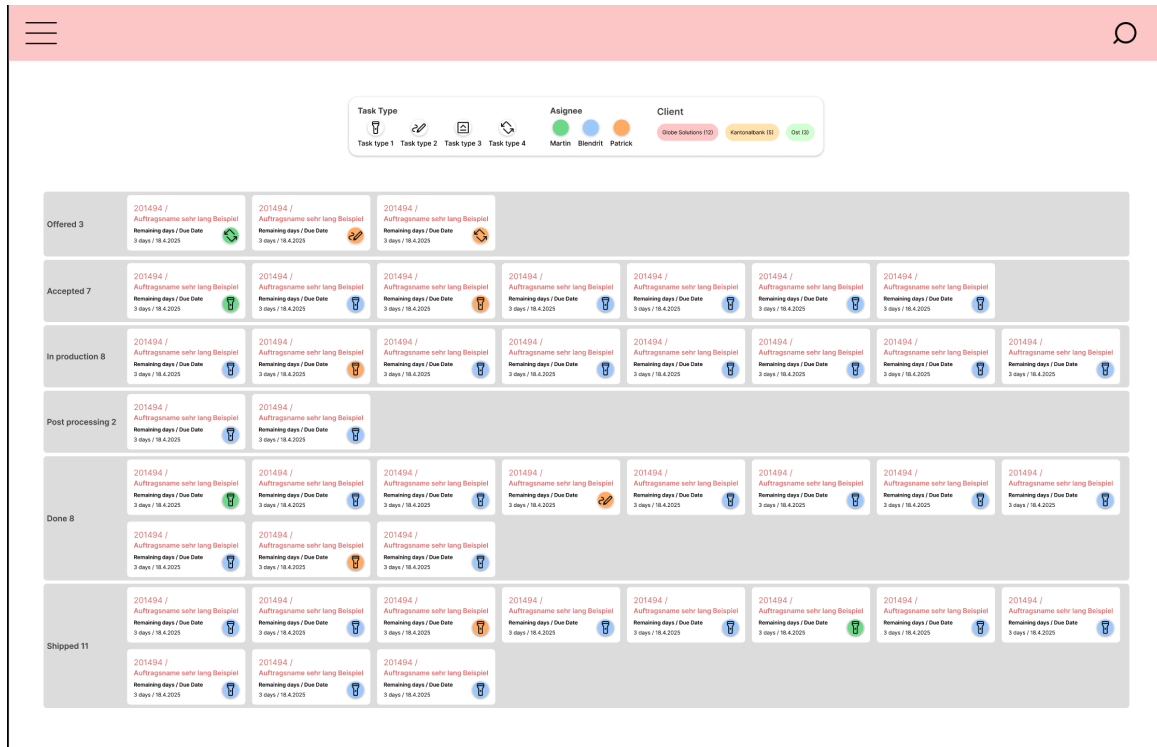


Figure 2.8: Simple Rows Arrangement

Using rows brings out the effect of the cards gradually wandering down, which represents the lifetime of an order. The limiting factor for how many cards can be fit on one screen vertically is the width of the cards, which means narrower cards are better suited, while the height of the card determines how many rows fit.

A big downside of this approach is the handling once there are more cards in a particular state than can fit in one row. In this scenario, either horizontal scrolling is used, which we generally want to avoid or the row becomes a double column, as shown in the states "Done" and "Shipped" in this example. This is potentially bad for space usage, as even if there is only one more card than how many can be displayed in one row, it instantly takes up a whole new row, and therefore the occupied space doubles. While sorting and filtering can be done nicely while using rows, the grouping gets worse. The example shown in Figure 2.6 grouping by assignee would look a lot worse using rows, as the text would take up a lot of space this way. A similar effect can be observed with the row titles which take up more space compared to the column titles. An alternative would be to just use a colored line but grouping by customer for example would still pose a problem.

Card Arrangement Variant 3: Islands

The third and final approach we considered initially was inspired by their physical boards which we called islands. While with columns and rows, each state takes up the whole vertical and horizontal space respectively, the island view does neither and sort of combines both approaches. Our interpretation of the island view is shown in Figure 2.9.

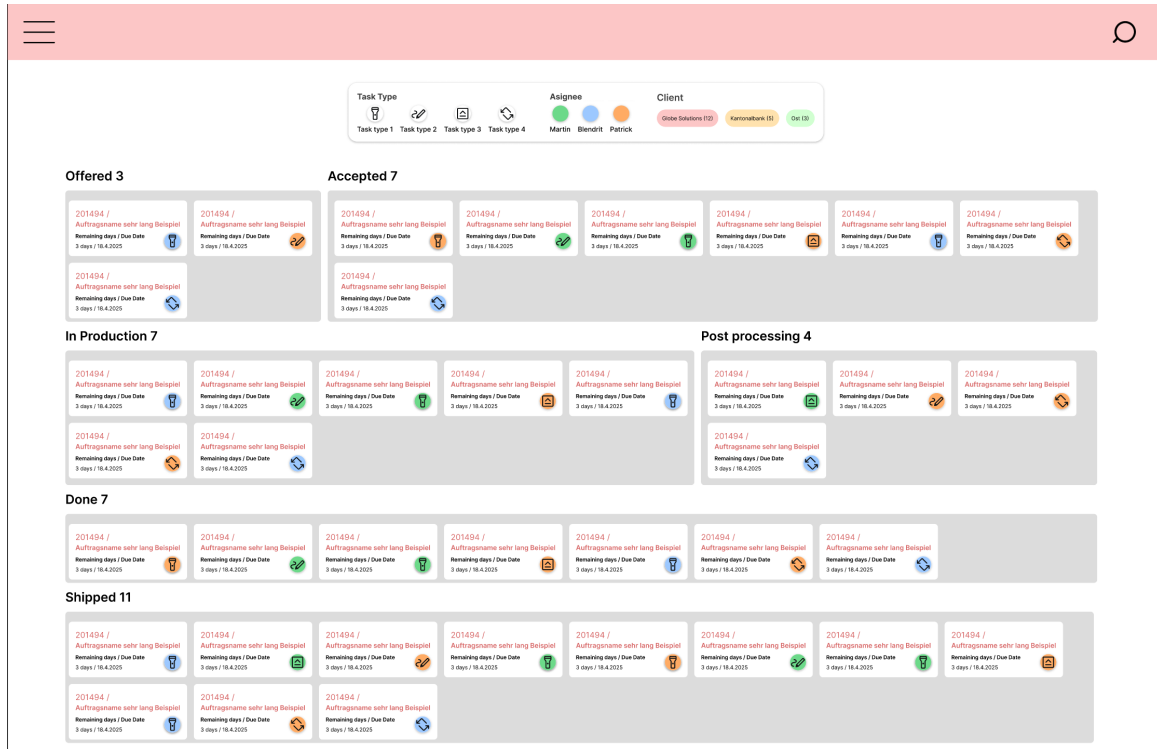


Figure 2.9: Island Arrangement

This example shows a potential arrangement that uses the island approach. Multiple states can be on one row but the flow of the cards is generally still from top to bottom. Islands could be a great choice for scenarios where the cards do not go through the same states each time.

For example, in case a card could be moved from state 1 into either one of states 2, 3, or 4, state one could be displayed as its own row and the other three could be displayed below together in one row. As this is not the case for us, the benefit is limited. With dynamic forming of islands when needed, the space usage could be the best of all variants but it would also be the most difficult to implement. Sorting and grouping however could become horrendous, as the viewer needs to adapt to all the different arrangements at once.

Detail View Design

Another component for which we made some initial design was the more detailed view of orders, which are displayed when clicking on a particular card. The initial design we came up with is shown in Figure 2.10.

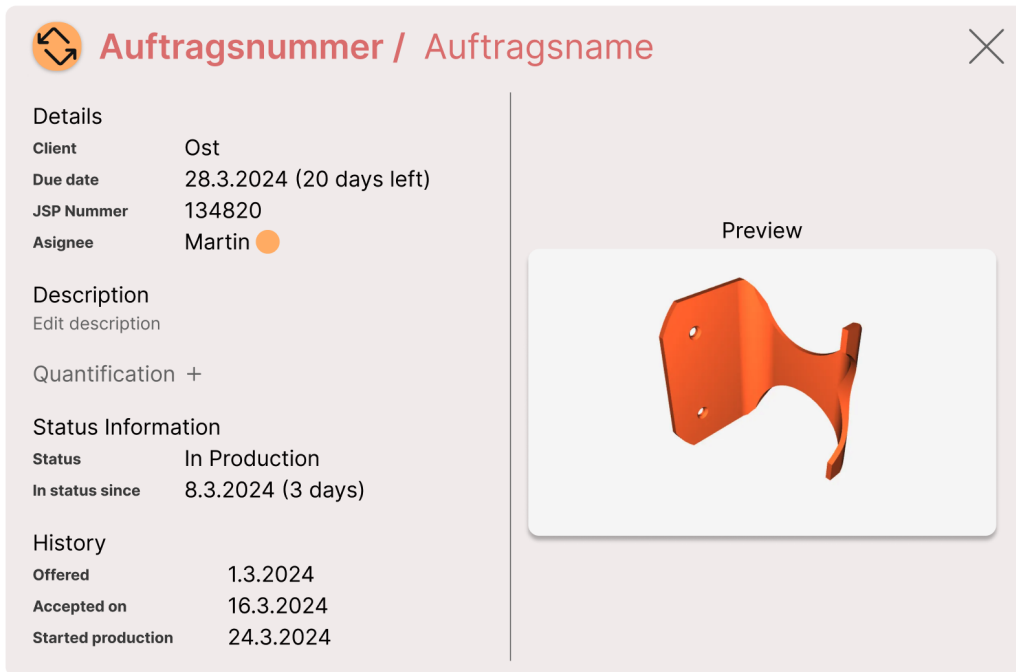


Figure 2.10: Detail View

We decided to group the general information as details. Additionally, to provide the customer with the ability to leave notes that are important for a particular order, we added an editable description field which is especially useful when more than one person is involved in the process of an order so they can share this information.

We decided to add another group called "Status Information" where additional information concerning the specific state an order is in can be displayed. Another addition is the history which has the goal to provide more traceability of the process an order went through. For example, if a deadline is missed, the customer has a better chance to see what went wrong and in which state an order spent too much time. To improve the recognizability of an order, we added a preview of the item that should be printed to this card as well.

As the customer has orders, that consist of multiple parts or parts that are needed more than once, we wanted to provide further support for that. We did so, by adding a section called "Quantification" to the card which is displayed in Figure 2.11.

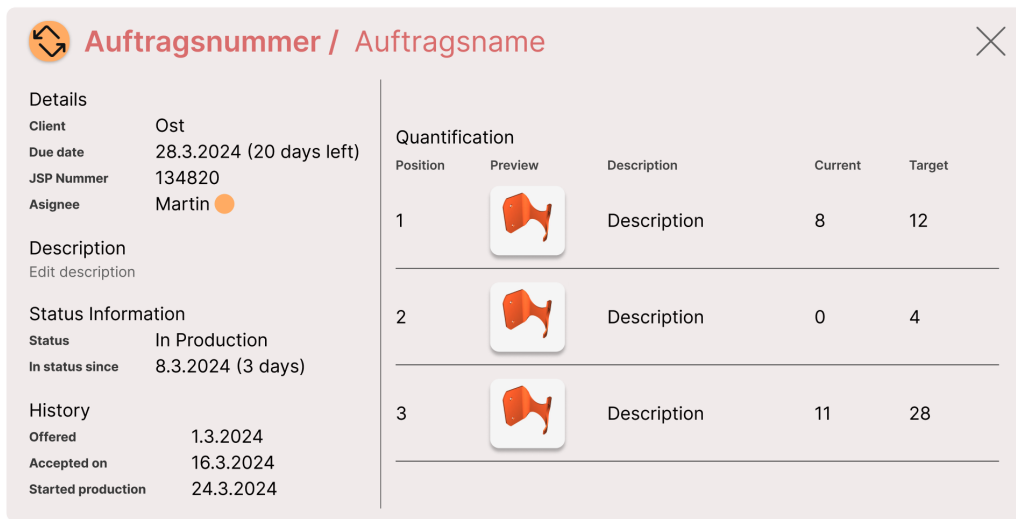


Figure 2.11: Quantified Detail View

The quantification section consists of a table that displays the different positions that an order consists of. In this table, each part has its own description and a counter for the number of parts that have been completed as well as the total amount needed.

Design Iteration 1: Review

After preparing all the designs mentioned in subsection 2.2.3, we sat down with the customer and discussed them. As an introduction we prepared a few slides to give them context to what we will present, which can be found in Appendix C. From there, we switched to *Figma* where we presented the options in motion. The most notable outcomes were the following:

Colored Icons

The client appreciated the concept of the colored icons which we will therefore reuse in future designs.

Card Arrangement

Variant 1 and especially the sub-variant (Figure 2.7), where the fullest states take up more space, was their preferred option, as it provides a good usage of space while maintaining a good overview. Therefore this variant will be used for further refinement.

Detail View

The possibility to have a description on the cards as well as the history was appreciated and will be maintained in future iterations.

2.2.4 Design Iteration 2: Finalization Card Design and Archive

After addressing the most important components in the first iteration, the goal for the second iteration was to incorporate the feedback from the first iteration and finalize the card designs. On top of that, a new page, the archive is designed, where completed orders are displayed.

Finalization Card Design

The client generally appreciated the design we presented at the first design review meeting. So in this iteration, it was not about redesigning the whole card, but to finalize the design and implement the feedback. In our initial design, we aimed to have the card as simple as possible and only display the most important information to limit the size of the cards which allows us to place more cards on a page without scrolling. They highlighted, however, that the JP Number, as well as the client name were a must-have for them as they use those to identify and differentiate between orders. Additionally, we suggested visualizing whether a card has been invoiced or not, so the users see which items need to be invoiced without expanding a card. Those inputs and considerations led to the design highlighted in Figure 2.12.

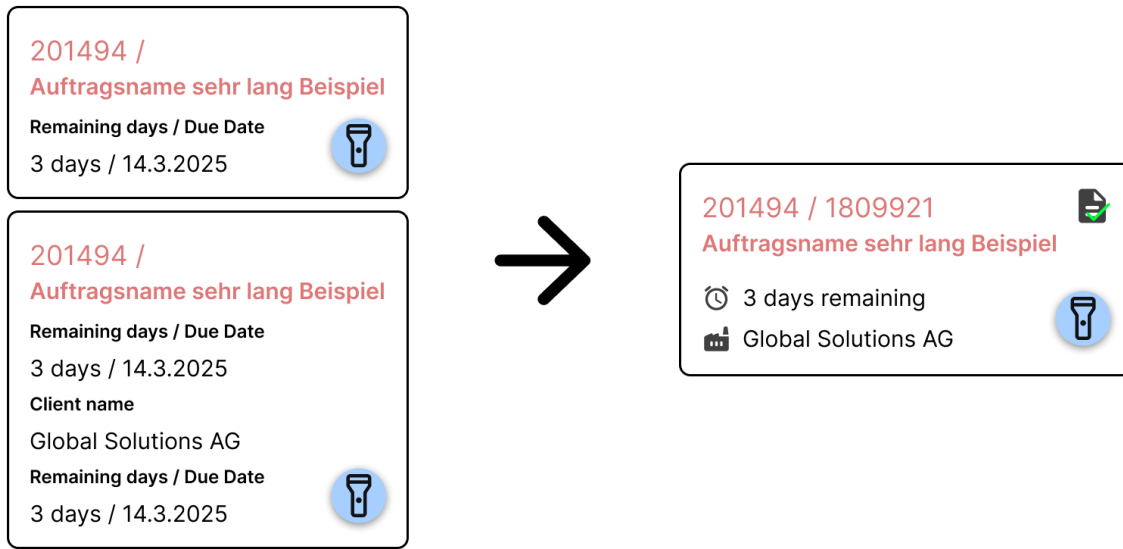


Figure 2.12: Before vs After: Finalized Card Design

The new card design shows the JP Number on the same line as the order ID separated by a slash, to not take up more space vertically. Although from the client's perspective, no label at all was necessary for the due date or the customer name, we opted to use icons to indicate, what the meaning of those fields is to keep the card design intuitive. This means we implemented metaphorical signifiers in the form of icons instead of explicit signifiers in the form of words to have more efficient space usage. Here, we were confronted with the challenge that even though we are not building a product to bring on the market and are building a custom solution for a specific client, that uses the tool daily, we still can't neglect design principles.

We saw the use of icons as the best compromise between no label at all and using a text label, which blows up the design. Additionally, we added an icon on the top right, which indicates whether an order has been invoiced or not. We opted to only display an icon when an order has been sent and just leave it out when there was not because this field is only relevant in certain states.

Another option was to display either an invoiced or a "not invoiced" icon in certain states, but we saw that as confusing. By going with that compromise of using icons, we intend to use tooltips that give the user context when hovering over the field. Another, more playful, idea would be to have the clock icon in a different color or even slightly pulsating when the deadline is close.

For this design, we implement the hints from [4] by taking advantage of human reading patterns. This

influenced where we placed each piece of information on the card as well as the text size and color. Through discussions with the customer, identified the item number, and the JP Number as the most important pieces of information for our users to identify an item. To prioritize the items we tried to look at it from their perspective. We envisioned during a stand-up for example when user A could as user B something about a specific item he would be most likely to use item ID (201494). So therefore we placed that piece of information at the top left. For user B to now get context of what user A is talking about, the next most important item would be the title of an order which is therefore placed just below.

We defined those two pieces of information, alongside an optional JP Number, as the card header, which is in a different font size and weight and color to the body, to distinguish clearly and emphasize the importance of those items. Additionally, we put the client's name on the card as well, to make the cards more identifiable as the title can sometimes be ambiguous where the client's name comes in as an additional distinguishable factor. As this is only the case for some and not all items, the client name was not put in the card header.

Full Page Design

The next item we tackled, was to generally decide on the whole page layout. For this item, our goals were, to everything around the Kanban board as simple as possible to not overload the user. One key criterion for us was, that the user always knows exactly where he is and where he can go from here.

With the page design, we tried to adhere to "Jakob's Law" here as well and familiarly designed the page based on the citation in [3]. By doing that, we want to make use of the value of familiarity which ultimately makes the whole page easier to process and understand. We wanted to use elements already found in a lot of web pages to reduce the mental effort required for a user and ultimately ensure a lower cognitive load. An example would be to display the search bar or the settings icon in a familiar place as the users will most likely look there first when looking for that. We did not want this law to limit our creativity and interpreted it in a way that not everything needs to be generic and look the same but we still want to leverage previous experiences of users in our favor when possible.

With those considerations in mind, we came up with the design shown in Figure 2.13.

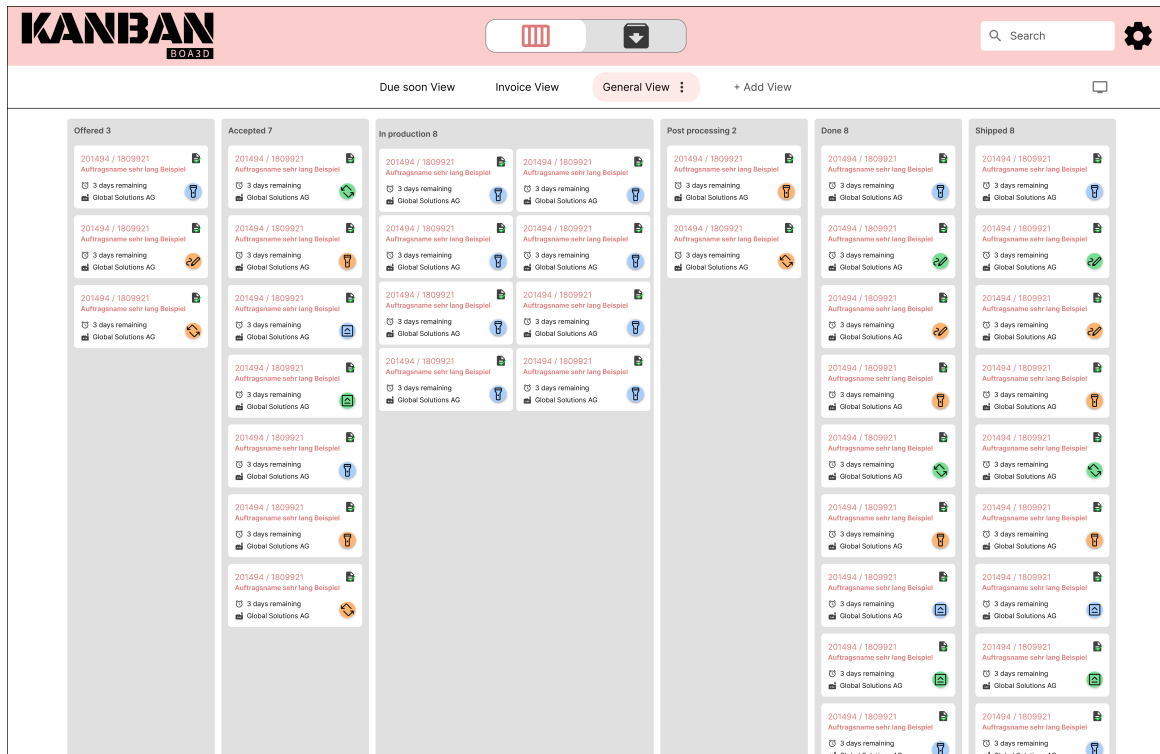


Figure 2.13: Full Page Layout

The general layout we selected is from the top down a header bar, another optional page-specific bar, and then the content. We chose to only work with horizontal separation with each level becoming increasingly more specific.

The header bar is generally present on every page within the tool and strictly does not contain any page-specific functionality or data. It serves the purpose of giving the user a sense of familiarity across all pages. Through the use of the logo, we wanted to provide an additional visual recognizability factor to distinguish our page.

As we only use two main pages, the board, and the archive, we elected to go with a more playful form of navigation by using a switcher. With the switcher, we followed the guide from [4] which emphasizes providing a clear visual orientation & navigation and states "A user browsing the web is not unlike a migrating bird. With all that open space and only a vague direction of where they want to go, measures must be taken to orient themselves to avoid getting lost.". With the switcher, we can always signal the current location of the user, using a different color. With the simple switcher with only two options, we want to provide the user with a sense of comfort as he always knows where he is, and where he can go, and the navigation is always in the same place to allow him to get back.

The search bar is also placed in this bar as the search allows the user to find both archived as well as active items. The reason to place the search functionality here and not in the page-specific bar is to indicate to the user that the search allows to look up all items. As we opted to not use any form of sidebars and the settings are not page-specific, we placed the settings in the header bar as well.

The second bar provides space for any page-specific settings or data, which is not the content itself. Through the use of the same background color as the content area, we indicate the connection between this area and the content. As loads of content are displayed in the content area, we opted to leave anything but the content itself out of it, to not overstimulate the user.

Archive Design

This page displays all items that have been completed. They are displayed on a separate page to not overload the main, Kanban board page.

For the design of the archive page, we looked at what the intent of the user is, when he views this page. In contrast to the board page, where the user wants to get an overview of all in-progress items, here the key factor is to have items displayed in a clear relation to the time they were completed. With this in mind, we landed on the design shown in Figure 2.14.

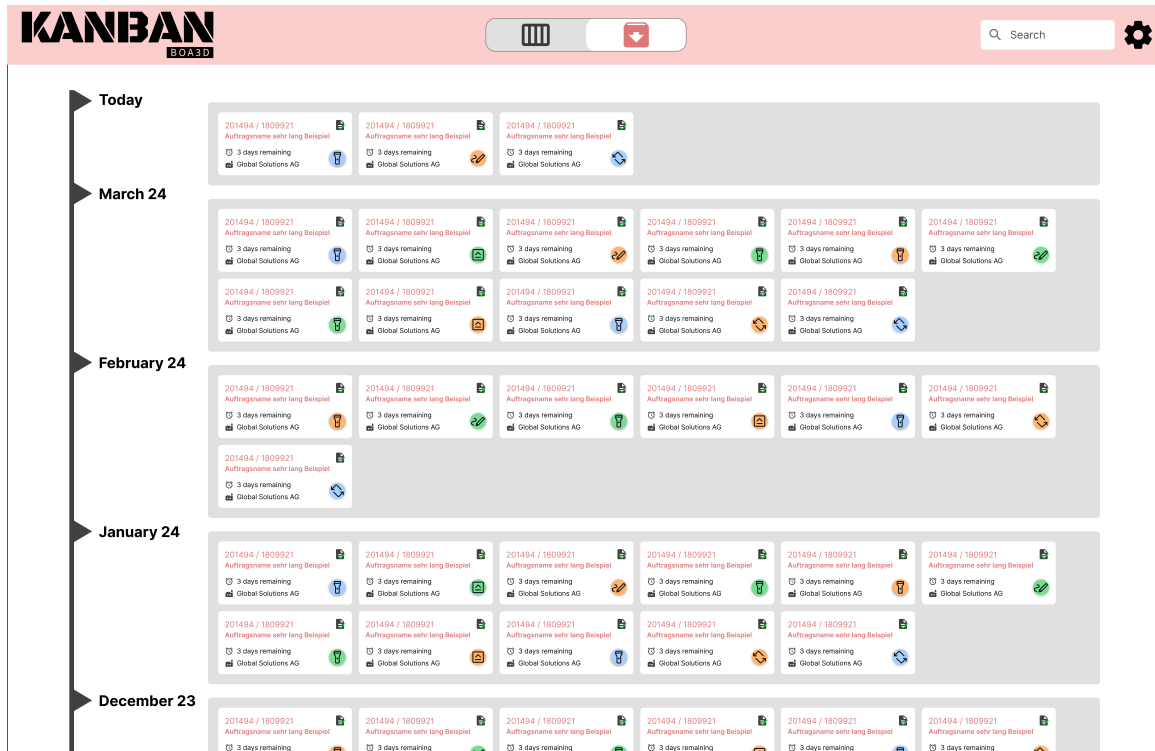


Figure 2.14: Archive Design

To give a clear sense of when an item was completed, we introduced a timeline that groups and separates the items based on the month of completion. The timeline is displayed vertically and the items are aligned in rows to ensure the archive fits on the screen without the need of horizontal scrolling.

2.3 Requirements

This section covers the requirements that were mainly generated through the iterative process shown in section 2.2.

2.3.1 Functional Requirements / User Stories

The user stories cover the functional scope of the tool from the perspective of different potential users.

Kanban Board – Overview

As the team leader,

I want to view all the active orders and see their state and assignee

so I can keep track of day-to-day business.

Kanban Board – Full Screen

As the team leader,

I want to view the Kanban Board in full screen on the TV

so we can discuss the active orders together.

Kanban Board - Live Updating

As the team leader,

I want the TV to always be up to date

so we can discuss the latest orders together.

Kanban Board – Drag & Drop

As someone working on orders,

I want to modify the state by dragging and dropping the orders

so I can keep the state of the orders up to date.

Kanban Board – State Updates

As someone working on orders,

I want the state to be updated on Bexio when an order is moved to accepted or done

so the Kanban Board and Bexio are kept in sync.

Kanban Board – Filtering - Assignee

As someone working on orders,

I want to filter out any orders not assigned to me

so I gain an overview of my tasks.

Kanban Board – Filtering / Sorting - Due Date

As the team leader,

I want to view the orders close to the deadline

to prioritize the work accordingly.

Kanban Board – Invoice

As the team member responsible for invoicing,
I want to see which orders have not been invoiced
to see which orders I need to invoice.

Search

As someone looking for details of a specific order,
I want to search for the order by order number
to find it quickly.

Card Details – Description

As someone having important information on an order,
I want to document that piece of information
so my colleagues have access to this information.

Card Details – JP Number

As someone working on orders,
I want to record the JP Number
so the orders correlate to the correct number.

Card Details – Assignee

As the team leader,
I want to assign an order to a coworker
so everybody knows their tasks.

Card Details – Reassign

As someone who wants to pass the order to a coworker,
I want to reassign an order
so the responsible person is modeled correctly.

Card Details – Due Date

As someone working in sales,
I want to set the deadline for an order I promised to the customer
so the work can be prioritized accordingly.

Card Details – Quantification

As someone working on an order with multiple positions,
I want to see how many parts for each position are required
so I know how many I have to print.

Card Details – Quantification Update

As someone who printed one part of an order with multiple positions,
I want to adjust the counter of the completed parts
so everyone knows how many have to still be printed.

Card Details – History

As the team leader,
I want to view the history of an item
so I can determine why it was delayed.

Archive – Overview

As the team leader,
I want to view the already completed orders
so I can look up details of previous tasks.

OneDrive - Folders As a team leader,

I want the system to create folder structures for each project
so I can store relevant data in these folders and don't have to create them by myself.

OneDrive - PDF As a team leader,

I want the system to automatically store the project PDFs in OneDrive
so I don't have to do it by myself and it can't get forgotten.

OneDrive / Card Details - 3D Models As a team member,

I want to view 3D Models which were uploaded to OneDrive
so I can easily associate the object that is printed with the order.

2.3.2 Non Functional Requirements

We define non-functional requirements, to make sure the software is not only functional but is also easy to use, conforms to security standards, and performs well. Here, we use the in-place solution to determine the improvements to the processes we want to achieve.

Performance - Time Behaviour

For efficiency, we defined the following requirements which previously were manual steps in the in-place solution and should now be automated steps:

- When an item is created on Bexio, it does not need to be created manually.
- When an item moves states on Bexio, it does not need to be moved on the board manually.
- When an item is moved on the board, it does not need to be kept in sync on Bexio manually.

Efficiency

For efficiency, we defined the following requirements which previously were manual steps in the in-place solution and should now be automated steps:

- When an item is created on Bexio, it does not need to be created manually.
- When an item moves states on Bexio, it does not need to be moved on the board manually.
- When an item is moved on the board, it does not need to be kept in sync on Bexio manually.

Usability – Consistency

After each item is moved to a different state, the item should be moved on every open client viewing the board.

Usability – Feedback

After an item is moved on the board the user should get a positive visual confirmation.

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 board view, he should reach the archive with one click.
- Given the user is on the archive, he should reach the board view with one click.
- Given the user is on the board view, he should be able to view the details of an item with one click.
- When the user is either on the board view or the archive he should be able to reach the settings with one click.

Reliability – Recoverability

Each item that was moved to a different state or the archive can be moved back to the original state, or recovered from the archive.

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. We have opted for a limitation of max 100 lines per method.

Security

Authentication parameters to third-party services should not be leaked. This includes the access credentials to Bexio and OneDrive. These should be put in environment variables and should be stored neither on a device nor the GitLab repository.

2.4 Market Analysis

In this section, we will compare our application against other tools that offer Kanban Board functionality. By doing so we aim to identify strengths and weaknesses of our tool. We will look at the features of other tools and establish for which use cases they would be fitting. For the competition we want to compare ourselves with, we tried to choose options that offer Kanban Board functionality but in as different forms as possible.

2.4.1 Jira

atlassian.com/software/jira

Jira is a project management and issue-tracking software developed by *Atlassian*. It is widely used for agile project management, allowing teams to plan, track, and release software efficiently. Even though *Jira* has a software management background, it could also be used for non-software projects like the 3D printing services Change 3D offers. One of the features *Jira* offers out of the box is a Kanban Board, which is the reason we compare our tool to *Jira* and looks as shown in Figure 2.15.

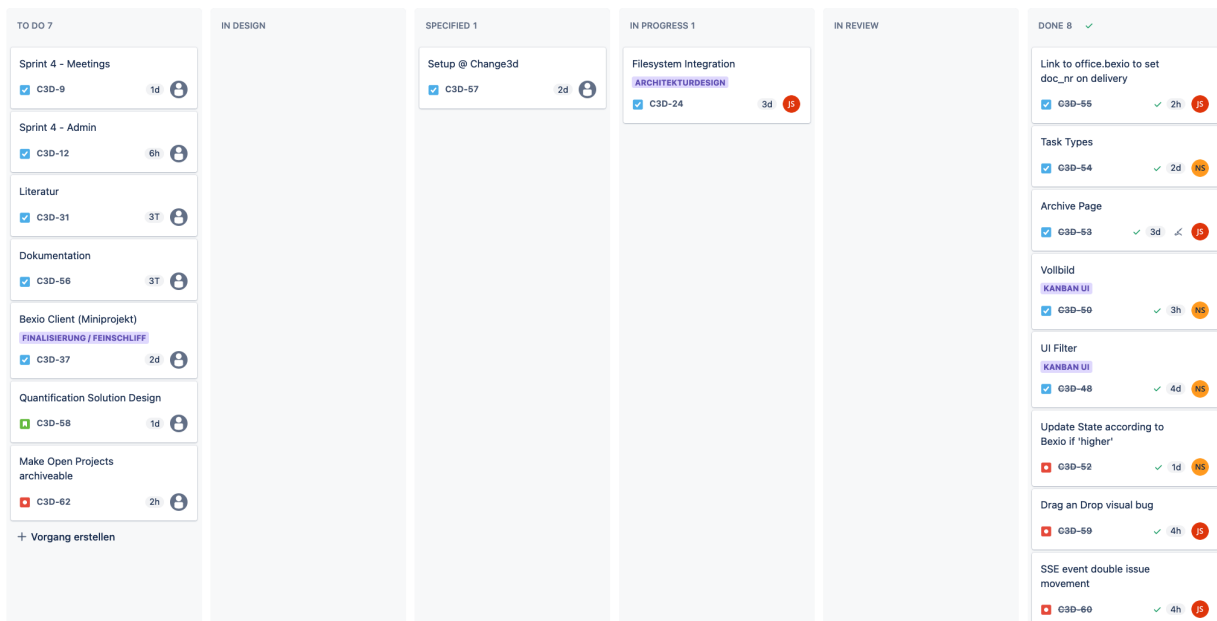


Figure 2.15: Jira Kanban Board

The screenshot shows the Kanban board we are using while developing this tool. As shown, the board looks simple but offers the necessary functionality for a basic Kanban board. In comparison to our tool where we can design the board so it best fits our customer's needs, the *Jira* board caters to a broader audience and is therefore more generic. The *Jira* board is very functional and can be customized as shown by Figure 2.16.

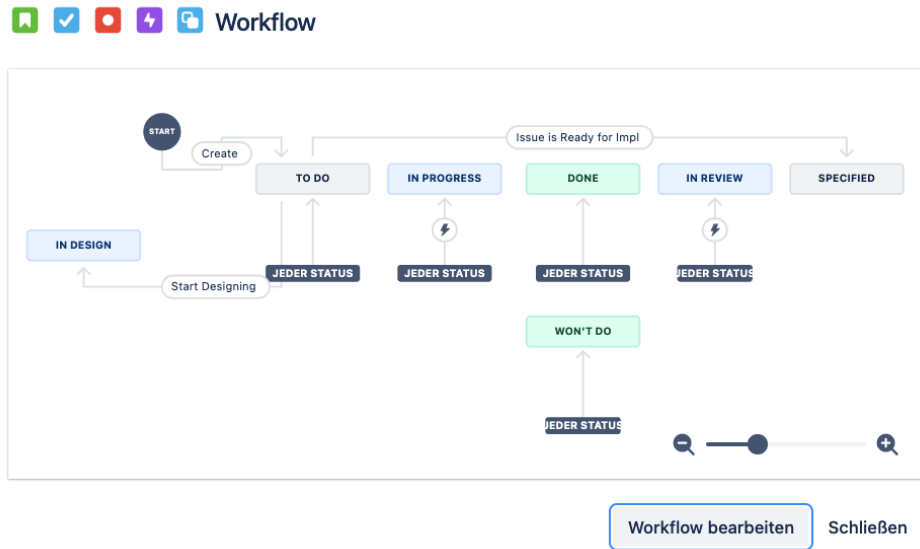


Figure 2.16: Jira Custom Columns

This figure shows the *Jira* feature which allows you to define the workflow of your issues and allow you to define restrictions, which state for example, from which state an issue can be moved to which other state. Another feature that makes *Jira* useful for a lot of use cases is, that it enables you to define states and columns.

What *Jira* does not allow is customizing how an issue is displayed on the board. For example, the use case to display the order number from Bexio, the JP Number, the title as well as when an item is due would not be possible on the board view. Additionally, while a group-by feature is offered the filters are rather limited. The board can only be filtered by assignee, epic, or issue type, while the issue can not be sorted.

Orders that have multiple positions could be modeled, using sub-tasks with each sub-task representing a position. Another approach could be using just text which would also feel makeshift. At least, those sub-tasks could be hidden from the board view so they do not show up on the board.

Overall while *Jira* is the best fit for Software Projects, it could be applied to other industries as well. *Jira* brings out-of-the-box support for features like assigning an issue to a user, setting deadlines, and leaving comments. With that in mind, there would be extra effort needed to synchronize the issues with the Bexio orders, which would either have to be done by hand, or by writing a plugin. A compromise could be setting up the workflow in a way that requires a checkbox to be checked before moving the issue to a state which requires an action on Bexio.

For a smaller team like Change 3D, using the Premium Plan, the *Jira* license would cost \$16 per user per month.

2.4.2 Miro

miro.com

Miro is a collaborative online whiteboarding platform that enables teams to brainstorm, plan, and collaborate visually in real-time. It offers a flexible canvas where users can create diagrams, mind maps, wireframes, and more using a variety of pre-built templates and customizable elements. With

features like sticky notes, shapes, and integrated video conferencing, Miro fosters remote collaboration and creativity among teams across different locations.

While in Jira everything evolves around Issues and the board is just a presentation option for the Issues, Miro is built around the board functionality. With the Kanban Framework Template, Miro can be used as a Kanban board as well as shown in Figure 2.17.

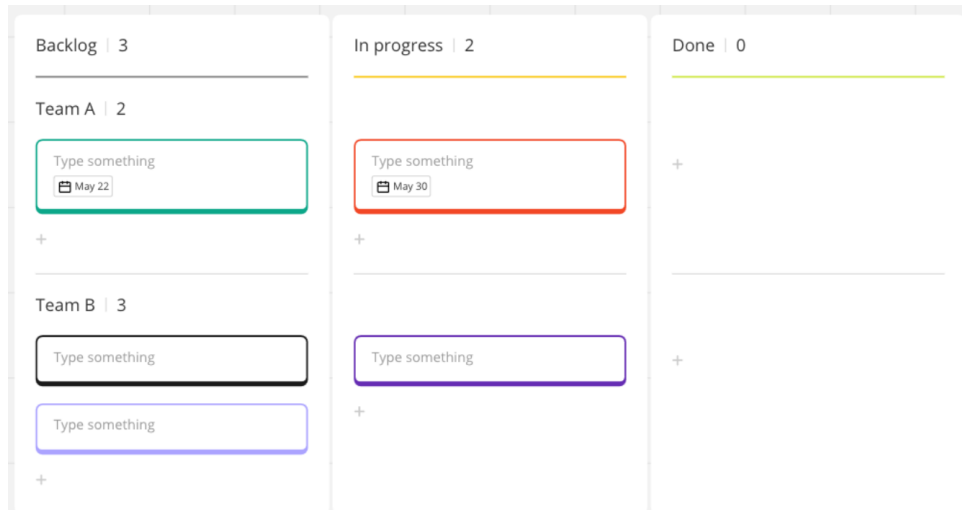


Figure 2.17: Miro Kanban Board

The Miro Kanban Board feels less restricted than the Kanban Board from *Jira* but also offers loads of functionality, like adding comments, assigning an issue, and setting deadlines. Overall, the Miro board feels free-flowing by allowing the user to quickly add new swim lanes, and columns without enforcing a lot of restrictions.

As the board generally is very free-flowing, no additional view options are offered. The items on the board can neither be filtered, sorted nor grouped. Additionally, no further fields, as an order number or a JP number could be defined on an item in Miro. However generally, everything other than the description is displayed on the board as shown by Figure 2.18

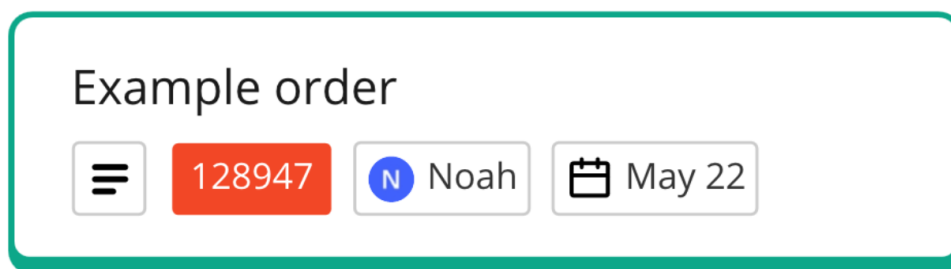


Figure 2.18: Miro Item

On this item, a title, and assignee as well as a due date have been defined and are displayed.

In contrast to *Jira*, there would not be a possibility to define sub-tasks to model positions of an order, so the only way to achieve that functionality would be using the description field.

This makes the Miro board easy to get started with and could be a great fit for startups and other smaller ventures. As with the *Jira* board, this board could not be integrated with *Bexio* out of the

box and the orders would need to manually be kept in sync. The downside of that is, that for more complex workflows, the lack of restrictions could lead to the board becoming messy, and in our case, a big risk would be Bexio getting out of sync with the board.

Using the starter pack, which offers sufficient features for this use case, Miro would set back a customer \$8 per user per month.

2.4.3 Summary

In summary, we see the *Jira* kanban board as the perfect fit, for teams that already use *Jira* to visualize their issues. Miro on the other hand caters to teams, that seek a less restrictive board to quickly visualize their items. A downside they both share is, that they cater to a broad audience, they have to be generic so they match as many potential users and use cases as possible while introducing their terminology. Especially *Jira* could be confusing for users who do not have a software development background and are not used to the terminology. This is also a big benefit of a custom product as we can use the terminology as it's lived by our customer and follow their real-world conventions as we do not have the restriction of having to cater to a broad audience and thereby adhering to the second heuristic as highlighted by [2].

Not having this restriction allows us to design the cards on the board in a way that they offer information that is relevant to the users. On top of that, we can specifically target pain points, like deadline management, by implementing filters to only display items that are due soon. Another benefit we see is, that by not catering to a broad audience we can keep a minimalist design by having less clutter around the features that this use case requires, compared to *Jira* or *Miro*. With a custom implementation, we can therefore follow the eighth heuristic of [2], which states "Interfaces should not contain information that is irrelevant or rarely needed. Every extra unit of information in an interface competes with the relevant units of information and diminishes their relative visibility".

Where both tools are more flexible than our application is that they quickly allow you to introduce new states (columns) on the fly, whereas, with the custom-developed solution, code changes would be needed. This downside however is connected to the fact that state transitions are connected to additional business logic like triggering actions on Bexio and simply introducing a new column would not solve the issue sufficiently.

Overall the biggest benefit of developing the board from scratch is, that we can offer a complete Bexio integration which could only somewhat be achieved with *Jira* and not at all with Miro. By developing a tool specifically for their needs, we can build around their processes and solve their specific problems without having to care for other features that would make sense in a different context but are not relevant to our partner. Furthermore, by closely working together with the client, we can get real-time feedback and tailor the solution exactly to their needs.

Chapter 3

Solution Design

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. This chapter is on a different level compared to the chapter Chapter 4 which will highlight key implementation details.

3.1 Architecture overview

Our application consists of three fundamental containers. On one hand, we use Angular for the web application, which visualizes the Kanban board. For the Backend, on the other hand, we use two containers. The core container provides a RESTful API, which communicates with the other containers and the data storage. The second part of the Backend consists of a scheduler service, which is solely responsible for communicating with the Bexio ecosystem. It also delivers data via RESTful API to the main Backend. This, in turn, occasionally informs the Frontend about updates.

To visualize our architecture designs, we use C4 diagrams to provide further insights.

Level 1: Context

The outermost level, the container level, shows the placement and interactions between the application and the external system, and is displayed in Figure 3.1.

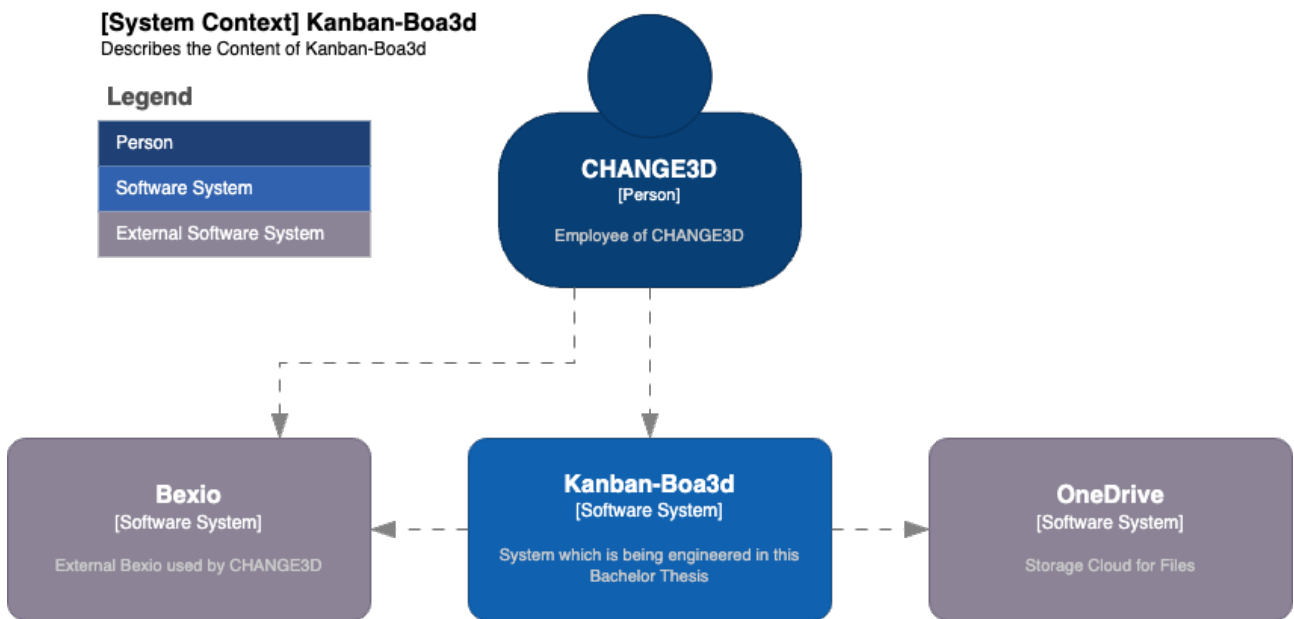


Figure 3.1: C4 Context Diagram

Person This is the primary user of the system, denoted by a large blue circle. CHANGE3D represents either an individual or a collection of individuals within CHANGE3D who interact with the software systems depicted. It's noted as an employee of CHANGE3D.

Kanban-Boa3d This is the system being engineered as part of this Bachelor Thesis. It is a variant of a kanban board, which is a popular project management tool to visualize work and optimize the flow of the work among the team.

OneDrive OneDrive for storing files, which can be accessed through an API.

Bexio Bexio is a third-party software, used by CHANGE3D to manage their orders and contracts.

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.

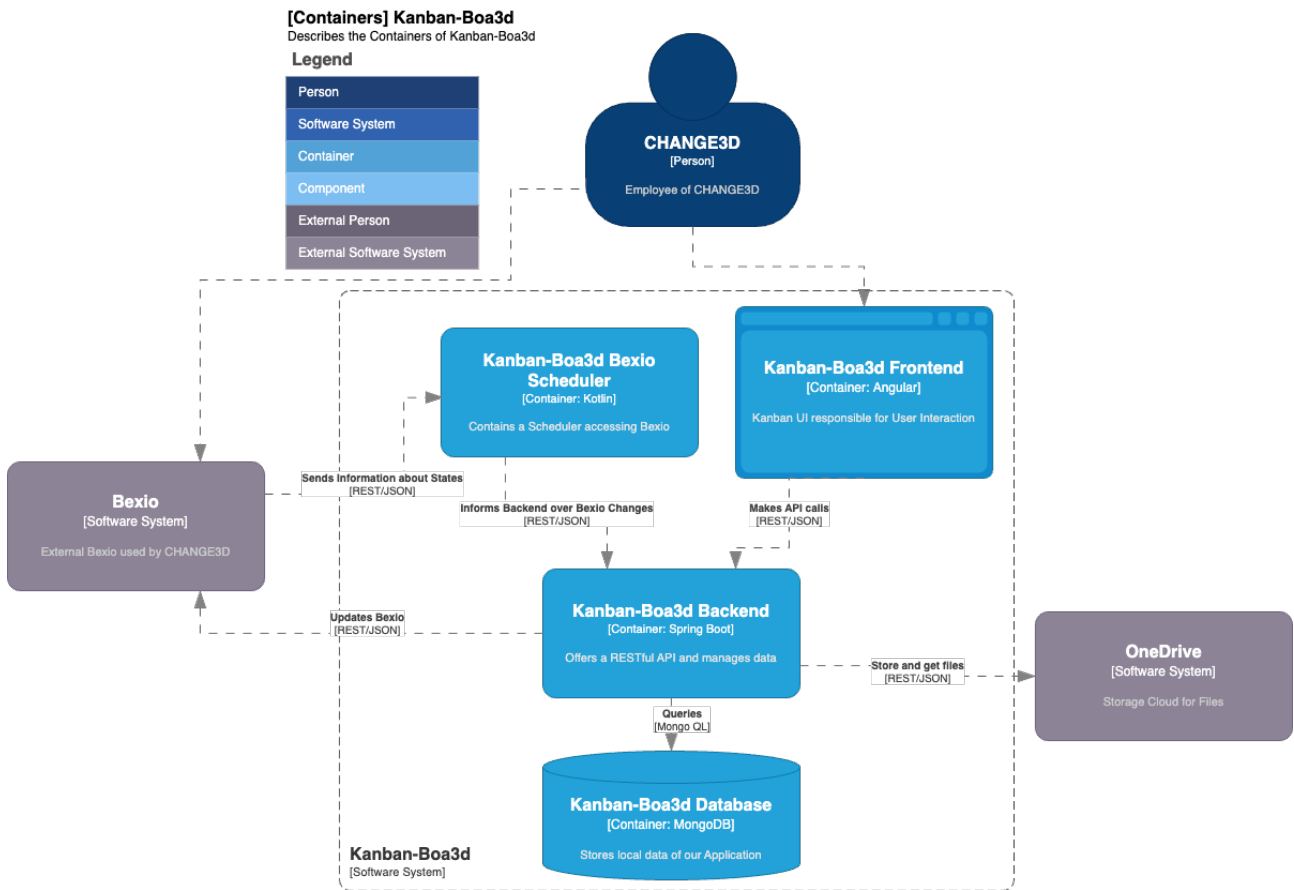


Figure 3.2: C4 Container Diagram

Kanban-Boa3d Backend The Backend, responsible for interactions with Bexio and OneDrive, as well as delivering data to the Frontend, eg. the end user.

Kanban-Boa3d Frontend Angular application responsible for communication with the Backend and displaying information to the user.

Kanban-Boa3d Bexio Scheduler The Scheduler which is responsible for fetching and converting data from Bexio and sending them to the Backend.

Kanban-Boa3d Database The database required to store all Kanban related data.

3.1.1 Kotlin over Java

For our bachelor's thesis, we chose Kotlin over Java to help us learn and use modern programming features. This brief analysis explains why we chose Kotlin over Java.

Advantages of Kotlin Over Java

1. **Clearer and Safer Code:** Kotlin allows us to write less code that's also easier to understand. It helps prevent common errors, like those caused by null values, making our software more reliable.
2. **Works Well with Java:** Kotlin and Java can work together seamlessly. This lets us use all the Java tools and libraries we're already familiar with while we learn Kotlin.

Challenges of Adopting Kotlin

1. **Learning Curve:** Learning Kotlin is a challenge because it's new to us. However, this fits our goal of gaining new skills and knowledge during our thesis project.
2. **Fewer Resources:** There are fewer tutorials and community support for Kotlin compared to Java. But the quality of available resources is good and growing as more people start using Kotlin.

Conclusion

Using Kotlin offers a great chance to improve our programming skills with its modern features and strong safety measures. Despite some initial challenges like a steeper learning curve and fewer resources, these aspects ultimately contribute to our educational goals, making Kotlin a smart choice for our project.

3.1.2 Spring Boot as our Backend

We opted for Spring Boot as our Backend framework over alternatives like Dropwizard or Node.js. This decision was driven by several factors including its seamless integration with Kotlin, ease of use, existing knowledge of the Spring framework, and efficient integration with MongoDB.

Reasons for Choosing Spring Boot

1. **Integration with Kotlin:** Spring Boot offers excellent integration with Kotlin, providing features that are well-suited to Kotlin's concise and expressive syntax. This compatibility lets us use Kotlin's safety and clear features, improving our development experience and making the code easier to maintain with fewer errors.
2. **Ease of Use:** Its auto-configuration capabilities automatically manage boilerplate code for database connections, security configurations, and other backend needs. This makes it particularly attractive for rapid development cycles and for those who prefer to focus more on application logic rather than on configuration details.
3. **Existing Knowledge in Spring Boot:** Our familiarity with Spring Boot significantly influences our choice. Leveraging existing knowledge allows us to be more efficient in development without the steep learning curve associated with adopting a new framework like Dropwizard or a different framework like Node.js. This familiarity also reduces potential risks and issues during the development process, as we are already use to the typical challenges and best practices associated with Spring Boot.
4. **Integration with MongoDB:** Spring Boot seamlessly integrates with MongoDB, facilitated by Spring Data MongoDB. This integration simplifies the implementation of database operations, providing automatic mapping between database documents and Java objects, and managing database connections. This is especially beneficial for our project, as we have to handle lots of data which we can easily store in the database.

Conclusion

Spring Boot stands out as the optimal choice for our project's Backend due to its strong support for Kotlin, user-friendly nature, the advantage of existing expertise within our team, and efficient MongoDB integration. These factors collectively ensure that our project progresses smoothly, efficiently, and with a high standard of quality.

3.1.3 Database Selection

The decision to use MongoDB over a relational database such as PostgreSQL in our project is based on an analysis of our application's specific needs, data consistency requirements, and setup efficiency. This is especially true in the context of leveraging Spring Boot and MongoTemplate for querying. Below is a comparison highlighting the advantages and disadvantages which ultimately were the basis for our decision.

Data Consistency

MongoDB: Our project benefits from MongoDB's flexibility in environments where eventual consistency is acceptable. This is due to the fact that our data originates from an already consistent database (Bexio), thus reducing the need for strict transactional consistency. MongoDB's schema flexibility allows for rapid iteration and changes to the data model without significant overhead.

PostgreSQL: Offers strong consistency, ensuring reliable and predictable transaction processing. This level of consistency is crucial for applications where strict maintenance of data integrity and relationships is necessary. However, for our project, this feature is less critical due to the consistent nature of our data sources and the lack of complicated relationships we need to model.

Setup Efficiency

MongoDB: Stands out for its ease of setup and schema-less nature. Unlike relational databases that require predefined tables and schemas, MongoDB's collections do not enforce a strict structure, significantly reducing initial setup time and allowing for dynamic changes as the project evolves.

PostgreSQL: In contrast, setting up a PostgreSQL database involves defining tables, columns, and relationships upfront, which can be time-consuming and less adaptable to changes. While this structure supports complex queries and data integrity, it may introduce setup delays, particularly for projects with evolving data requirements.

Querying with Spring Boot and MongoTemplate

MongoDB and Spring Boot: The integration of MongoDB with Spring Boot, facilitated by MongoTemplate, offers a streamlined approach to querying and database interaction. MongoTemplate simplifies complex queries, data manipulation, and aggregation operations, aligning well with rapid development practices.

PostgreSQL and Spring Boot: Though Spring Boot supports PostgreSQL efficiently through JPA, jOOQ, and other abstractions, the relational model sometimes requires more verbose configurations and queries. This can be particularly true when dealing with complex joins and relational mappings, which, despite being powerful, may not offer the same level of simplicity and speed as MongoDB for certain applications.

Conclusion

The selection of MongoDB for our project is motivated by its flexibility, rapid setup time, efficient development, and querying capabilities offered through the integration with Spring Boot and MongoTemplate, but also by our interest in trying out new technologies. This choice suits our application's context, where the consistency provided by traditional relational databases is less of a concern. While PostgreSQL has its advantages in scenarios requiring complex relational data modeling and strict consistency, MongoDB's schema-less design and compatibility with modern development frameworks present a compelling case for projects prioritizing development velocity and adaptability. After making

our shortlist of Database possibilities in quite a few projects in the past, MongoDB was finally the best fit this time.

3.1.4 Logging

As we do not provide any form of authentication it is essential to have the best possible logging, to keep track of what is happening. To do so, we introduced logging in our Backend as well as in our Scheduler application. Both applications use Logback to execute logging tasks. To strictly log every operation on the Scheduler application, we benefit from an interceptor in our OkHttp Client.

```
.addInterceptor {
    val logger = KotlinLogging.logger{}
    val request = it.request()

    logger.info { "Sending request ${request.url}" }
    val response = it.proceed(request)
    logger.info { "Received response for ${response.request.url}" }

    response
}
```

As we are using Spring Boot in our Backend, we benefit from Logging from Spring. We added a `CommonsRequestLoggingFilter` which does the same as our Interceptor introduced in the Scheduler application. As logging itself is not very useful, we create log files for both of our services. Managing log files has the advantage that we can track actions in the past and trace back potential bugs, vulnerabilities, and user actions.

3.2 Security Considerations

Introduction

In our development, we faced a challenging decision regarding the hosting and security model. We considered two main options: cloud hosting with user authentication and internal hosting without user authentication. After careful evaluation, we decided on the latter. This document details the reasons for our choice, as well as the advantages and potential drawbacks of this approach.

Chosen Solution: Internal Hosting Without Authentication

3.2.1 Problem

Our product is hosted internally on the company's servers, accessible only through the office's local area network (LAN) or WiFi. This setup excludes external internet access, thereby limiting access to employees who are physically present in the office or connected to its network. Furthermore, we have integrated the application's display with the office's TV, which is used to showcase real-time information relevant to all members.

Advantages

- **Simplified Access:** By hosting internally and removing authentication, we reduce the complexity for users. This is particularly beneficial for communal displays like the office TV where users require quick and easy access to the application without the hassle of logging in each time.

- **Enhanced Security Within the Network:** The restriction of access to only the internal network adds a layer of security. External threats are considerably mitigated as attackers outside the physical office environment cannot access the application.
- **Reduced Development Time and Cost:** Avoiding the implementation of an authentication system simplifies the development process, reducing both time to launch and associated costs. This also lowers ongoing maintenance requirements related to user account management and authentication troubleshooting.
- **Ease of Use:** Employees can access the application seamlessly from any connected device within the office without undergoing authentication procedures, fostering a smoother user experience and encouraging usage.

Drawbacks

- **Limited Remote Accessibility:** With access restricted to the internal network, remote work scenarios are not supported unless VPNs or other secure remote access tools are utilized, which could reintroduce complexity and potential security risks.
- **Dependence on Physical and Network Security:** The security of our application is heavily reliant on the physical security of the office and the integrity of the internal network. Any breach in these areas could expose the application to unauthorized access.
- **Scalability Issues:** As the company grows or if it opens new locations, the internal hosting approach may pose challenges in scalability and management, requiring additional infrastructure and possibly leading to fragmented data silos.
- **Compliance and Audit Concerns:** Without authentication logs, it may be difficult to perform audits. However, we try to perform a non-repudiation nonetheless, using information such as internal IPs.

3.2.2 Future Considerations: Implementing an IAM System

The introduction of an Identity and Access Management (IAM) system could be considered to enhance security measures, especially in scenarios involving more complex access requirements or remote access capabilities. An IAM system would enable:

- **Centralized Management of User Identities and Access:** Simplifying the control over who accesses what within our systems and applications.
- **Enhanced Security Features:** Such as multi-factor authentication, which can significantly mitigate potential security breaches.
- **Regulatory Compliance:** By providing necessary audit trails and user management capabilities to meet compliance requirements.

Keeping the possibility of an IAM in mind, we designed the application in a way, that Identity Access Management could be introduced in the application, with minimal to no adjustments.

3.2.3 Conclusion

The decision to host our product internally without user authentication was driven by the need for easy access within a controlled environment, the desire to eliminate the complexities of managing authentication systems, and the suitability of our office's security measures. This approach aligns with our current operational needs and office setup, providing a straightforward and secure solution for in-office use.

3.3 Live Board Solution Design

This solution design fulfills the requirements specified in 2.3.1, enabling the software to update in real-time. Throughout the solution development process, we explored two potential technologies: Server-Sent Events (SSE) and WebSockets.

Both technologies are well-suited for our needs, with the primary distinction being WebSockets' capability to receive data from the Frontend. However, since our system handles all write operations through REST requests, the bidirectional communication feature of WebSockets offers no additional benefit to us. Consequently, we have opted for SSE.

SSE, or Server-Sent Events, is a web technology that enables a server to send updates to the client browser via a single, persistent connection. This method is particularly efficient for sending notifications or live updates to the Frontend without requiring the client to make repeated HTTP requests.

SSE allows for the transmission of various message types. We have designed our system to use the following message categories:

ONE: This message type is used when a property of a single project is updated, or a project is retrieved from the archive to be displayed again in the main view.

MANY: This type updates all projects simultaneously. It is triggered when the scheduling service retrieves a new set of data from Bexio.

ONE_REMOVAL: This message type is utilized when a project is archived and should no longer be visible in the main view.

Although it is feasible to use only the **MANY** message type, this approach has two significant drawbacks. Firstly, our chosen method allows the Frontend to visually indicate specific updates to projects, enhancing user experience by providing immediate feedback on the changes. Secondly, it significantly improves system performance as it avoids the need to reload all projects continuously.

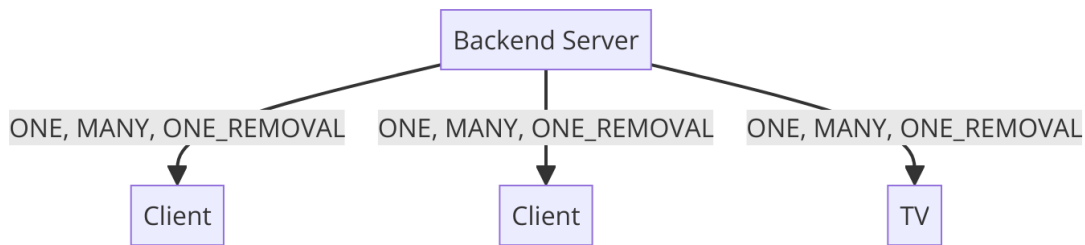


Figure 3.3: Interaction Diagram: Server-Sent Events

3.4 Bexio Transitions Solution Design

When a state transition to "Done" or "Accepted" happens, we need to interact with Bexio, our order management tool, to accept the offer or handle further processes. Specifically, for the "Accepted" state, we must accept the offer on Bexio. For the "Done" state, we need to create both an invoice and a delivery. The following sequence diagrams illustrate the technical solutions to achieve these use cases.

3.4.1 Accepted State Transition

This diagram shows the sequence of REST requests made to Bexio when a project is moved to the "Accepted" state. The process involves searching for the quote, accepting it, creating an order from the offer, and setting the document number on the order.

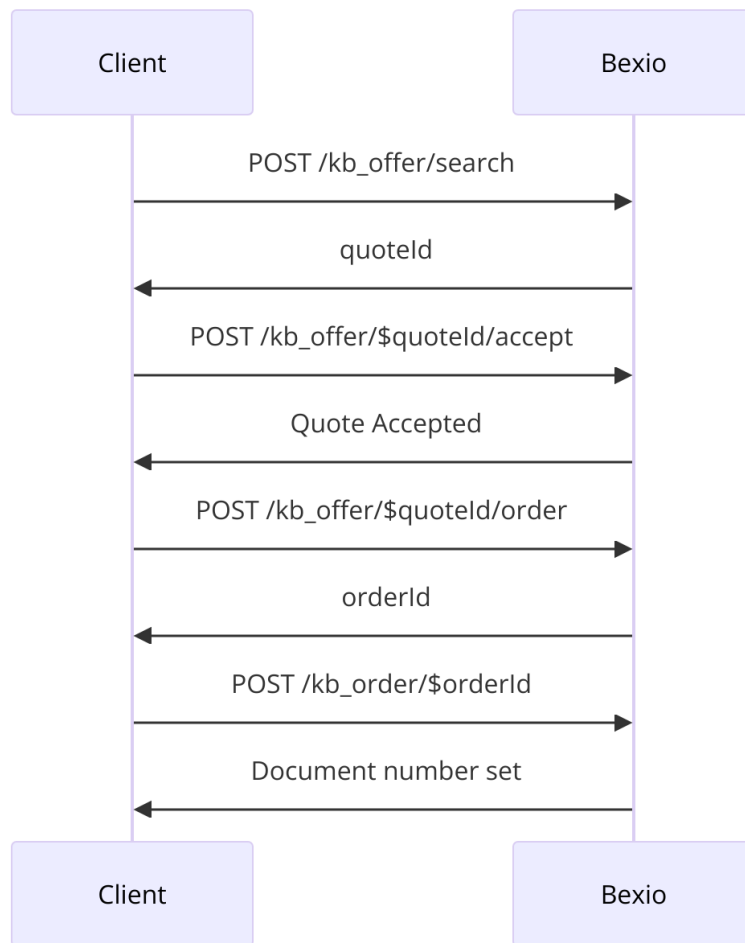


Figure 3.4: Sequence Diagram: Accepted State Transition

3.4.2 Done State Transition

This diagram shows the sequence of REST requests made to Bexio when a project is moved to the "Done" state. The process involves searching for the order, creating an invoice from the order, and creating a delivery from the order.

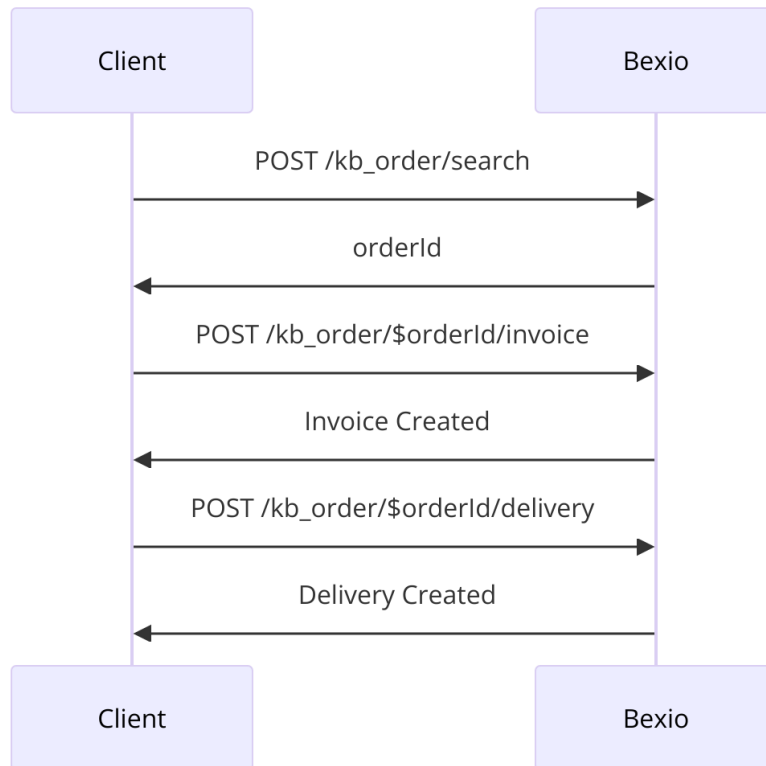


Figure 3.5: Sequence Diagram: Done State Transition

3.5 OneDrive Integration Solution Design

Our solution for integrating OneDrive with our project management system involves several key processes to ensure seamless management and accessibility of project-related documents. The functional requirements which are covered by this solution design are:

OneDrive - Folders, OneDrive - PDFs, OneDrive - 3D Models

Below are the sequence diagrams illustrating the technical solutions for each use case.

3.5.1 Creating Folder Structure for New Projects

When a new project is inserted into the database, a folder structure is created on OneDrive to organize project documents effectively. The process includes creating a root folder named after the document_number, and within this root folder, creating subfolders named "Dokumente" and "GCode". These requests are executed as batch requests to OneDrive.

1. Create a root folder with the document_number
2. Create a folder named "Dokumente" inside the root folder
3. Create a folder named "GCode" inside the root folder
4. Batch request endpoint: POST [https://graph.microsoft.com/v1.0/\\$batch](https://graph.microsoft.com/v1.0/$batch)

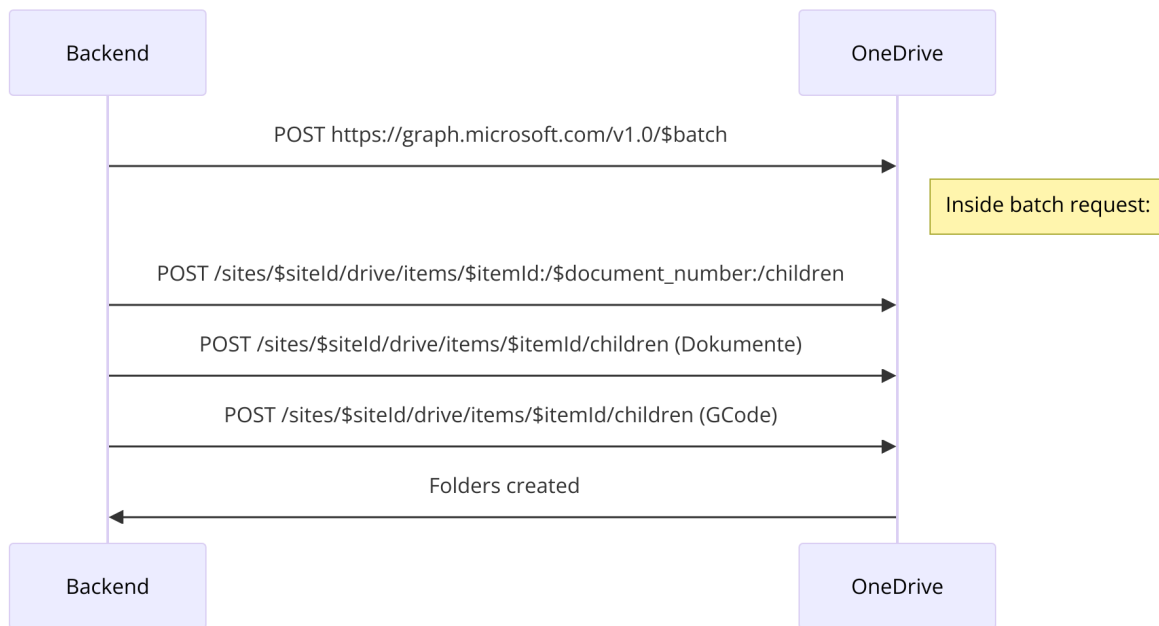


Figure 3.6: Sequence Diagram: Creating Folder Structure

3.5.2 Uploading PDFs for Accepted or Done State Transitions

When a project transitions to the "Accepted" or "Done" state, the respective PDF is downloaded from Bexio and uploaded to the root folder of the project on OneDrive.

For Accepted State:

- Backend requests PDF from Bexio: GET /kb_order/\$orderId/pdf
- Bexio responds with a PDF file
- Backend uploads PDF to OneDrive: PUT ../\$itemId:\$document_number/\$pdfName:/content

For Done State:

- Backend requests PDF from Bexio: GET /kb_invoice/\$invoiceId/pdf
- Bexio responds with a PDF file
- Backend uploads PDF to OneDrive: PUT ../\$itemId:\$document_number/\$pdfName:/content

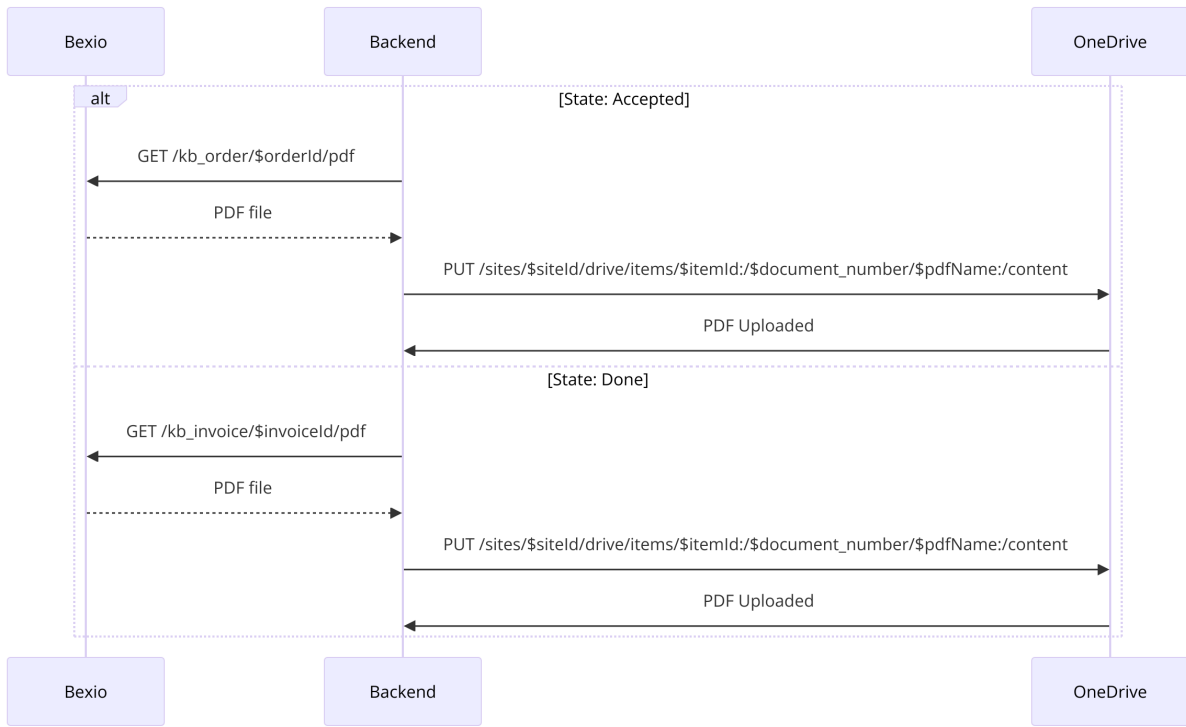


Figure 3.7: Sequence Diagram: Uploading PDFs

3.5.3 Querying and Rendering 3D Files in Project Detail Card

When the detail card of a project is opened, OneDrive is queried for the "Dokumente" folder inside the project folder to check for *obj* or *stl* files. If these files are available, they are downloaded and displayed as 3D renders in the project detail card.

1. Query OneDrive for obj or stl files:

- Endpoint: GET /sites/{siteId}/drive/items/{itemId}/{document_number}/Dokumente:/children

2. Download and render 3D files

- Endpoint: GET /sites/{siteId}/drive/items/{fileId}/content

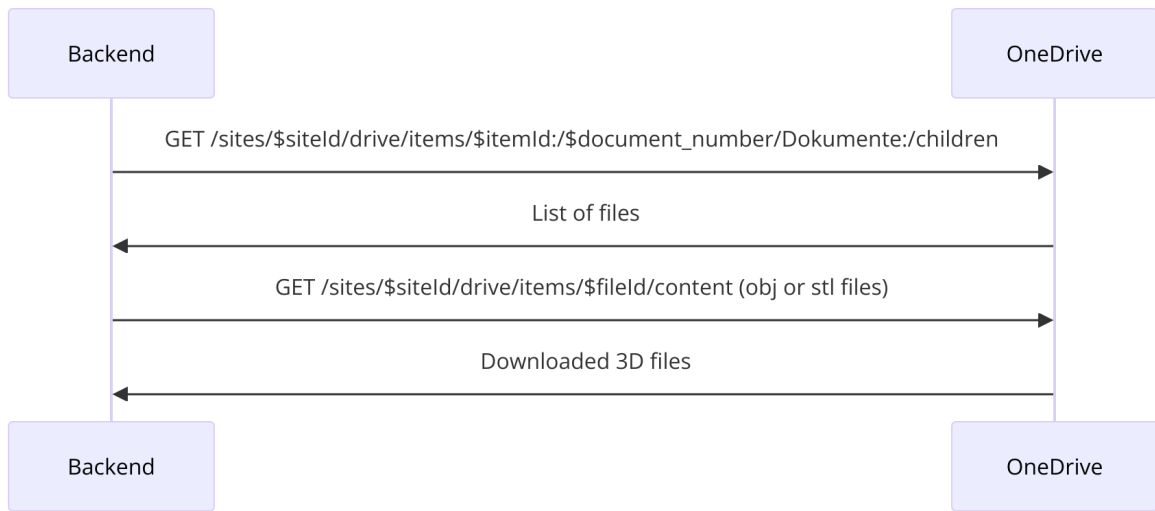


Figure 3.8: Sequence Diagram: Querying and Rendering 3D Files

Chapter 4

Implementation

In this section, we talk about our biggest challenges during the project's implementation. We cover interesting solutions we came up with and detail the steps we took, including some specific code. This part of the thesis shows how we turned our initial concepts into a working project, highlighting the practical steps and creative problem-solving that got us there.

4.1 Communication between Services

To streamline our services' integration, we utilize OpenAPI. Specifically, we generate an OpenAPI specification directly and automatically from our Backend's RESTful HTTP endpoints. To do so, we use a maven plugin.

```
<groupId>org.springdoc</groupId>
<artifactId>springdoc-openapi-maven-plugin</artifactId>
```

Leveraging our build system, Maven, we also automatically produce API clients for both our consuming services. Firstly, for our Frontend, we create an Angular HTTP client, and secondly, we generate one for our Scheduler Application, which is developed in Kotlin.

```
<groupId>org.openapitools</groupId>
<artifactId>openapi-generator-maven-plugin</artifactId>
...
<executions>
  <execution>
    <id>angular-client-code-generation</id>
    <phase>integration-test</phase>
    <goals>
      <goal>generate</goal>
    </goals>
    <configuration>
      <inputSpec>${project.build.directory}/openapi.json</inputSpec>
      <generatorName>typescript-angular</generatorName>
      <output>${project.basedir}/../kanban-boa3d-client/src/app/api</output>
    </configuration>
  </execution>
  <execution>
    <id>java-client-code-generation</id>
```

```

    <phase>integration-test</phase>
    <goals>
      <goal>generate</goal>
    </goals>
    <configuration>
      <inputSpec>${project.build.directory}/openapi.json</inputSpec>
      <generatorName>kotlin</generatorName>
      <output>
        ${project.basedir}/../kanban-boa3d-scheduler/target/generated-sources/api
      </output>
    </configuration>
  </execution>
</executions>

```

As we can see, there are two executions defined. Once for the Angular code generation and then for the Kotlin code generation. By defining the output of the clients, we can automatically add them to the other projects.

This build logic enables us to seamlessly connect all our Containers simultaneously. By doing so, we boost our productivity and flexibility since we no longer need to manually commit our REST clients—everything can be automatically generated from our Backend.

4.2 Deployment

We achieve flawless deployment through Docker Compose, which enables us to deploy rapidly. This approach allows us to deploy successfully at the end of every sprint.

Our Docker Compose setup orchestrates three key containers: Backend, Scheduler, and MongoDB. The Backend also directly serves the Frontend, removing the need for a reverse proxy.

For deployment, we leverage GitLab CI to craft Docker images. Specifically, we have a publish job that builds images and pushes them to the GitLab Container Registry. This setup allows us to pull the images onto any target machine.

The publishing step in the *gitlab-ci.yml* file is as follows:

```

publish:
  image: docker:latest
  variables:
    IMAGE_TAG_BE: $CI_REGISTRY_IMAGE/kanban-boa3d-server:latest
    IMAGE_TAG_SCHEDULER: $CI_REGISTRY_IMAGE/kanban-boa3d-scheduler:latest
    # Instruct Docker not to start over TLS.
    DOCKER_TLS_CERTDIR: ""
    # Improve performance with overlayfs.
    DOCKER_DRIVER: overlay2
  services:
    - name: docker:dind
      # explicitly disable tls to avoid docker startup interruption
      command: [ "--tls=false" ]
  stage: publish
  only:
    refs:

```

```

- master
before_script:
- docker login $CI_REGISTRY -u $CI_REGISTRY_USER -p $CI_REGISTRY_PASSWORD
script:
- docker build -t $IMAGE_TAG_BE kanban-boa3d-server
- docker push $IMAGE_TAG_BE
- docker build -t $IMAGE_TAG_SCHEDULER kanban-boa3d-scheduler
- docker push $IMAGE_TAG_SCHEDULER

```

Once we publish the images to the Container Registry, we actively pull and launch the service orchestra using Docker Compose. We've deliberately kept this process manual to maintain flexibility.

In a further GitLab CI job, deploy, we deploy the just produced docker images onto our test system. This step is essential for our goal of Continuous Delivery. It closes the circle and makes the latest version available all the time. To do so, we use the feature of remote deployments, provided by docker. In particular, we deploy using the following script snippet

```

script:
- DOCKER_HOST="ssh://ins@srbsci-29" docker login $CI_REGISTRY -u $CI_REGISTRY_USER
- DOCKER_HOST="ssh://ins@srbsci-29" docker-compose down
- DOCKER_HOST="ssh://ins@srbsci-29" docker-compose pull
- JWT_TOKEN="$JWT_TOKEN" DOCKER_HOST="ssh://ins@srbsci-29" docker-compose up -d

```

As we can see, using the DOCKER_HOST variable, we can define a remote deployment location. However, as security is a concern, we can not just deploy it onto another machine. Due to that, we are required to configure an Asymmetric key pair, through which GitLab CI can connect to our test host. To do so, you have to store the public key in the `authorized_keys` file which resides in the `.ssh` folder on the test host. The private key, on the other hand, must be configured in the CI Settings of GitLab. This can be achieved as follows:

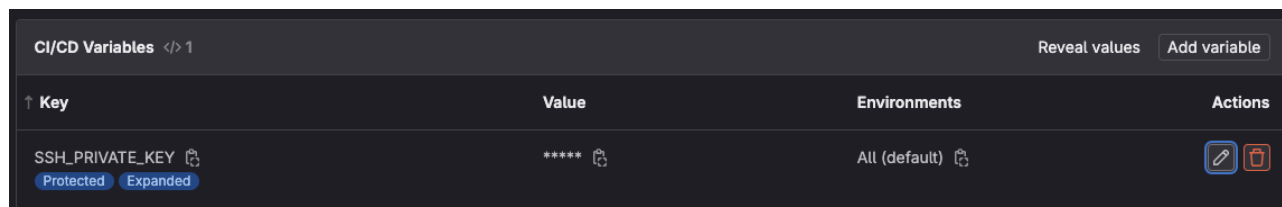


Figure 4.1: GitLab CI Configuring Variables

After configuring the private key in the GitLab environment, we can access it in our build file. By using the `before_script` section in `gitlab-ci`, we configure to private key:

```

before_script:
- eval $(ssh-agent -s)
- ssh-add <(echo "$SSH_PRIVATE_KEY")
- mkdir -p ~/.ssh

```


4.3 Bexio Communication

4.3.1 Authentication at Bexio

To connect to Bexio, we need to authenticate with their service, for which there are two options. The first and most prevalent method is authentication via OpenID Connect. Setting up this method requires following the steps outlined in their Documentation. However, the complexity of this flow, including the need to renew authentication periodically, presents a significant drawback. Recognizing this complexity, we decided against it and opted for the second option.

The alternative involves using a JWT (JSON Web Token), which an Administrator can generate. Once we receive this JWT Token from CHANGE3D, we can access Bexio by including it in the Authentication Header as follows:

```
Authentication: Bearer <My JWT Token>
```

Given its simplicity, we chose this second option. You can find more details about the setup [here](#).

4.3.2 Secure Management of JWT in Deployment and Development

To ensure secure handling of the JWT and maintain best practices for application deployment, we have adopted a method that avoids storing the JWT directly in the source code, thereby mitigating significant security risks. Instead, we use environment variables to supply the JWT, which helps to separate sensitive data from the codebase.

Deploying our application using Docker containers via GitLab CI presented a challenge due to the isolated nature of Docker containers. In response, we introduced a secure variable within GitLab CI specifically designated for the JWT. This approach ensures that the JWT is securely stored and only accessible during the deployment process. As Docker containers are initiated, the JWT is automatically populated from this secure GitLab CI variable, ensuring the token remains protected and is never exposed in the code or the Git repository.

For local development, we configure the IDE to run the service by manually adding the JWT as an environment variable in the run configuration settings. This setup allows us to work securely on the application without needing to alter the core security measures.

4.4 Bexio Data Export and Transformation

As stated before, we have created a scheduling service, which is solely responsible for the export of data from Bexio. The following chapter covers this export in depth.

4.4.1 Relevant Endpoints

Data that we want to export appear across multiple endpoints as their state progresses from offering to invoicing. Specifically, the endpoints involved are:

- `/kb_offer` – returns all offered projects
- `/kb_order` – returns all accepted projects
- `/kb_delivery` – returns all projects with a delivery note created
- `/kb_invoice` – returns all projects for which an invoice was created

A project initially appears in the `/kb_offer` endpoint and can subsequently appear in `/kb_order` once accepted. This overlapping presence across endpoints necessitates a filtering mechanism to ensure that each project is uniquely identified according to its most advanced state.

4.4.2 Filtering Algorithm

The goal of the filtering algorithm is to map projects to their respective states: Open (from `/kb_offer`), Accepted (from `/kb_order`), and Done (from `/kb_delivery` and `/kb_invoice`). Here is the step-by-step breakdown of the algorithm:

1. Fetch the project lists from each endpoint.
2. Process the endpoints in the order of the project's lifecycle: `/kb_offer`, `/kb_order`, `/kb_delivery`, and `/kb_invoice`.
3. For each project in an endpoint:
 - If the project ID exists in the following endpoint, remove it from the current one.
 - If the project ID does not exist in the following endpoint, keep it in the list.
4. After processing all endpoints, all lists will accurately reflect the highest state of each project.

4.5 Bexio Updates

In the integration with Bexio, we are also required to create state transitions on the Bexio System itself. To do so, we utilize the API provided by Bexio. However, the API poses big challenges when it comes to setting the document number through all those stages. Managing these document numbers effectively across business transactions is crucial, especially since these identifiers link various stages of a transaction cycle together. Document numbers, set by CHANGE3D at the onset of an initial offer, must be consistently copied to subsequent transaction stages, such as offer acceptance, shipment creation, and invoice generation. This consistency is important for the system to work and to track a project effectively through all stages.

4.5.1 Implemented Solutions for Document Number Management

In the solution design of Bexio (section 3.4), we have visualized a sequence diagram for a state transition. This section will go to the in-depth problems and solutions. Our system ensures the document number from the initial offer is programmatically propagated through subsequent transaction stages:

Accept an Offer: Upon acceptance of an offer, the document number is programmatically retrieved from the initial offer and applied to the order. This step involves a second API call to edit the order to set the document number, leveraging an undocumented feature that we have managed to utilize effectively.

Create an Invoice: When generating an invoice, the document number from the initial offer is required to be included directly in the API call, however, also leverages an undocumented feature. Our system automates this inclusion, ensuring the invoice is correctly linked back to the originating transaction.

4.5.2 Persistent Challenge with Shipment Creation

Create a Shipment: The most significant ongoing challenge exists in the shipment creation process. Currently, the Bexio API does not support the inclusion of the document number in the shipment creation request. Moreover, it also lacks the capability to edit the shipment post-creation to add the document number. This limitation poses a considerable hurdle as we are not able to set the document number in any programmable way. Attempting to modify the shipment record to include the document number post-creation is not possible, as the API returns a "not implemented yet" status. **Workaround:** Until these API functionalities are implemented, we are required to instruct CHANGE3D to manually adapt those changes in Bexio itself. To ease that workaround, we open a new tab in the browser, which redirects them directly to the page where you can set the document number.

4.5.3 Future Outlook

The inability to set the document number in shipments remains a significant gap in our integration with Bexio. We are hopeful that with the communication with Bexio, a solution will be developed that allows for complete and efficient document number integration across all stages of our transactions.

4.6 SVG generation for Archive

To display the time beam in the archive view, we have opted to create these visuals with self-multivated SVGs. This time beam consists of three properties; a line, a triangle, and a text. By giving all these three properties the appropriate relative X and Y coordinates, it creates a time beam, which should look as close as possible to the *Figma* design in Figure 2.14.

```
<svg width="220" height="100%" class="timeline-graph">
  <line x1="30" y1="0" x2="30" y2="999999" stroke="black" stroke-width="7" />
  <polygon points="33,0 33,40 63,20" style="fill:black" />
  <text x="75" y="25" style="font-weight: bold; font-size: 20px">
    {{itemsPerMonth.month}}
  </text>
</svg>
```

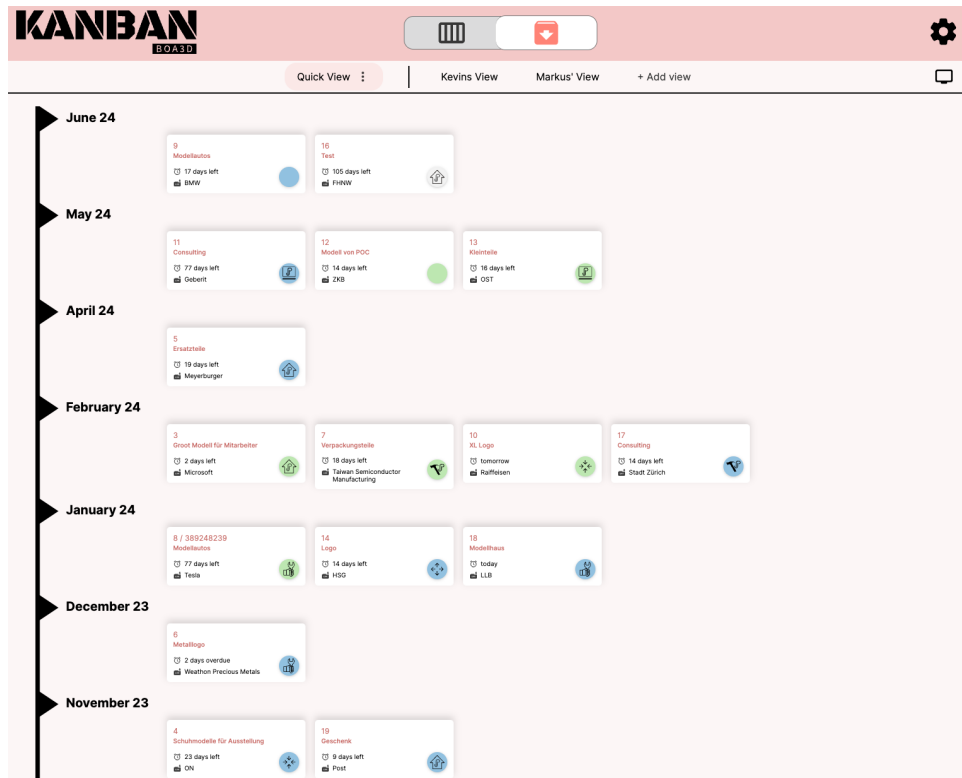


Figure 4.2: Archive Beam implementation

4.7 Integration of OneDrive

As covered in section 3.5 we have opted to integrate OneDrive. The following section covers challenges with authentication and rate limits, and how we solved them.

Authentication and Authorization

Integrating with OneDrive required implementing OAuth 2.0 for authentication and authorization. One of the main challenges was securely handling the access tokens, ensuring they were refreshed appropriately to maintain continuous access without manual intervention. Moreover, configuring the correct permissions (scopes) in Azure AD to allow our application to interact with OneDrive resources posed an initial challenge. To avoid any security leaks, we provide the access token through an environment variable.

```
fun getToken(): String? {
    val credential: IClientCredential = ClientCredentialFactory
        .createFromSecret(System.getenv("ONEDRIVE_AUTH"))

    val cca = ConfidentialClientApplication
        .builder(clientId, credential)
        .authority(tokenUrl)
        .build()

    val parameters = ClientCredentialParameters
        .builder(setOf("https://graph.microsoft.com/.default"))
```

```
        .build()

        return cca.acquireToken(parameters).join().accessToken()
    }
}
```

API Rate Limits and Performance

For each imported project from Bexio, we need to create a dedicated folder in OneDrive with multiple sub-folders. These folders serve as repositories for 3D models, contracts, designs, and more related to the projects.

Unfortunately, the OneDrive API consists of harsh rate limiting. On a first-time import, we need to create more than 1000 folders on OneDrive. This huge number combined with rate limiting posed a challenge to us. To solve this challenge, we have opted for an asynchronous approach which creates these folders over some time without weakening the rest of the system. To achieve an asynchronous approach, we leveraged the Spring Async feature, which makes it as easy as putting an `@Async` annotation on top of the method. By adding a fair timeout we were able to avoid any further rate limit conflicts.

4.8 3D Model Visualization Integration

CHANGE3D creates 3D models for each project, which are subsequently printed. To streamline the workflow and enhance the user experience, it is essential to integrate these 3D models into our system. This integration allows users to view all relevant models associated with each project, thereby providing a comprehensive overview and ensuring that the models are ready for printing.

4.8.1 Integration Process

To visualize the 3D models in our system, we employed *three.js*, a popular JavaScript library used for creating and displaying animated 3D graphics in a web browser. The integration involved several key steps:

- **Model Conversion and Storage:** The 3D models are converted into a base64 encoded string format, which is a method of representing binary data in an ASCII string format. This encoding is necessary for efficient data storage and transmission over the web.
- **Loading and Rendering:** Using *three.js*, we dynamically load the base64 encoded 3D models into the system. The *OBJLoader* is utilized to parse the OBJ file format, which is a common 3D model format. The models are then rendered in the browser using WebGL.
- **User Interaction:** To allow users to interact with the models (e.g., rotating, zooming), we integrated *OrbitControls* from *three.js*. This enables users to manipulate the 3D view for a better inspection of the models.

4.8.2 Challenges and Solutions

The integration of 3D model visualization presented several challenges, which we addressed through various solutions:

Challenge 1: Cross-Browser Compatibility

Ensuring that the 3D models render consistently across different web browsers was a significant challenge due to variations in WebGL implementations.

Solution: We performed extensive testing across various browsers and implemented fall-backs for browsers with limited WebGL support. Leveraging *three.js* also provided a layer of abstraction that mitigates some of the cross-browser inconsistencies.

Challenge 2: Lighting and Shadows

Properly lighting the 3D models to enhance visibility and realism posed a challenge.

Solution: We used a combination of *AmbientLight* and *PointLight* in *three.js* to ensure the models were well-lit from all angles. Adjusting the light intensity and position dynamically based on user interactions (like rotating the model) helped maintain consistent lighting.

Challenge 3: User Interaction

Providing a seamless and intuitive interaction experience for users manipulating the 3D models required careful consideration of control sensitivity and responsiveness.

Solution: By integrating *OrbitControls*, we enabled smooth rotation, zooming, and panning of the 3D models. Fine-tuning the control parameters, such as damping and sensitivity, ensured a responsive and user-friendly interaction.

Chapter 5

Results

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

5.1 Results

We will not only show the results but also compare the results to the initial requirements and go into any differences between the initial design and the finished product.

5.1.1 General

The general composition of the page is as it was defined in the designs. Our application starts with the bar shown in Figure 5.1.



Figure 5.1: Header bar

As defined, the bar serves the purpose of providing recognizability which we achieve through the logo at the top left. The page switcher serves as our form of navigation by both showing, where the user currently is, and where he could go. That leaves the settings icon which opens the overlay shown in Figure 5.2.

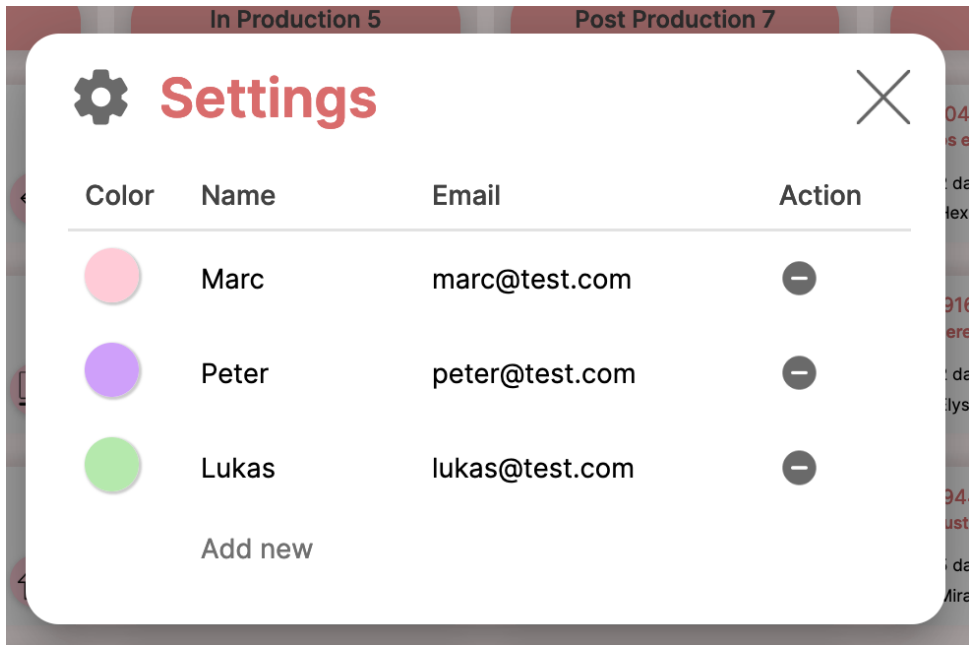


Figure 5.2: Settings

The settings currently are kept lean and allow the user to make changes to the users who use the tool by adding and removing users. As with all the overlays found in the tool, the rest of the page gets a lower opacity to help the user focus on the open panel. All the colors that the users can have can be selected through a color picker as shown by Figure 5.3

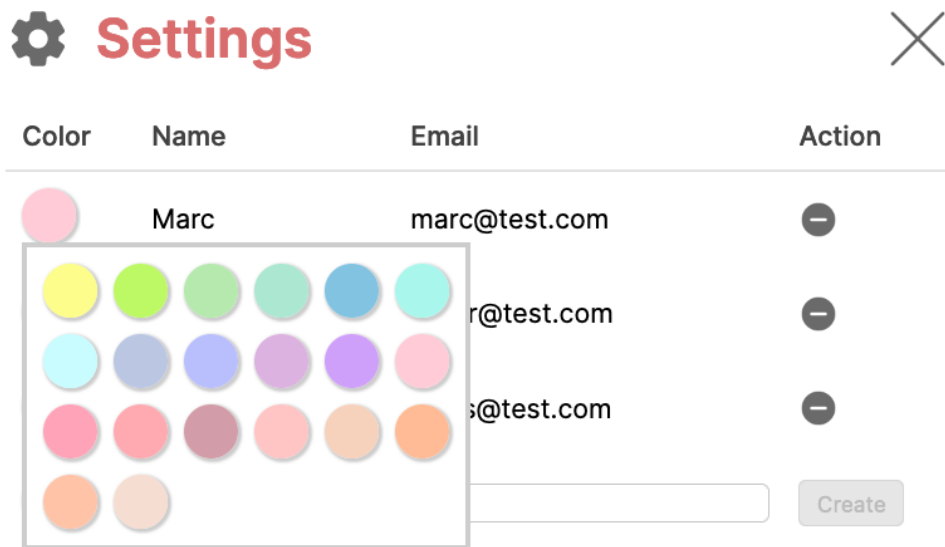


Figure 5.3: User Color Picker

The idea of the color picker is to provide the users with sufficient options but also prevent them from picking colors that would clash with the color scheme of the page or would have bad visibility.

5.1.2 Board

The core component of our application is the board. It is largely as it was designed from a composition point of view but has undergone a major restyling to achieve a more modern look. The finished board design is shown in Figure 5.4.

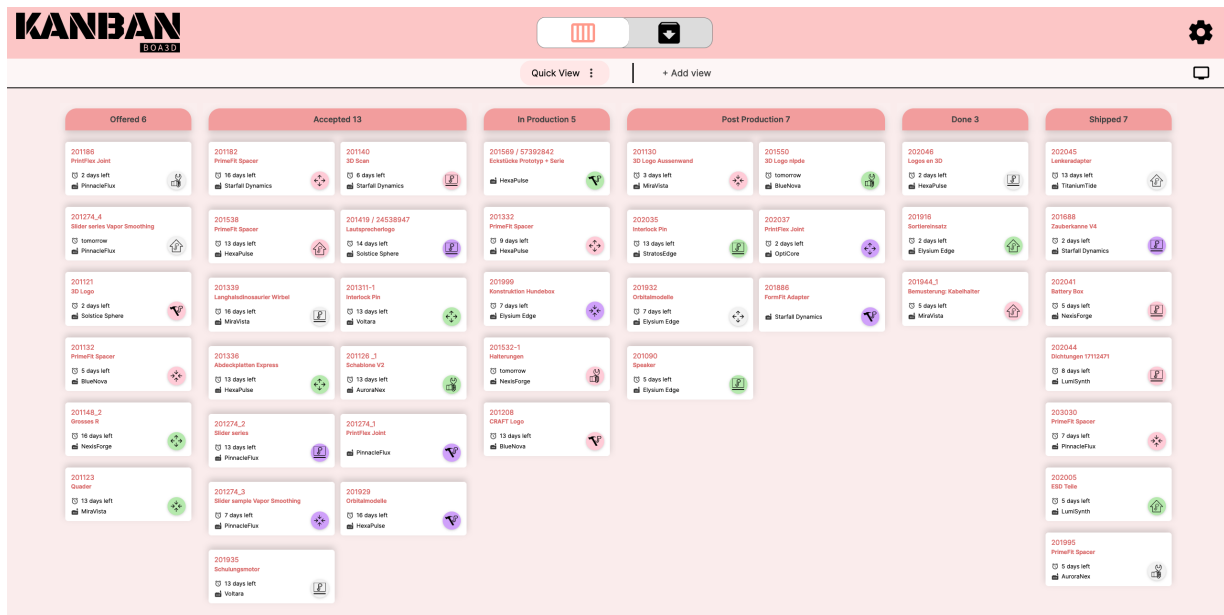


Figure 5.4: Board

The board allows the users to gain a quick overview of the state of affairs. Through the width of the columns and the number in the column title, the viewer can identify how many items are in each state. The board fulfills the requirement Kanban Board - Overview. All the active orders are displayed in this view and the assignee can be identified by the color of the sphere and the item type by the icon. This component also fulfills the requirement Kanban Board - Live Updating as well, as each change on a different client is reflected live on the board as well. The items can be moved between states through drag & drop thereby fulfilling Kanban Board - Drag & Drop. Additionally, when an item is moved to a new state, the corresponding action is automatically triggered, which fulfills Kanban Board - State Updates.

Most notably when compared to the original designs, the columns have no longer a background color giving the board a more modern feel. Here we wanted to follow the section "Don't Fear Blank Space" from [4] which states, "All good visual artists understand the importance of negative space, the empty area that draws attention to, and accents, the actual subject". Even though the columns are no longer separated as strictly as before, it is still easy to identify to which column an item belongs and it is even easier to focus on the items. This is based on the "Law of Proximity", as cited in [4], which states that images near to each other appear similar. This recognition is still maintained, even when there are too many items in one column to display them all as shown by Figure 5.5.

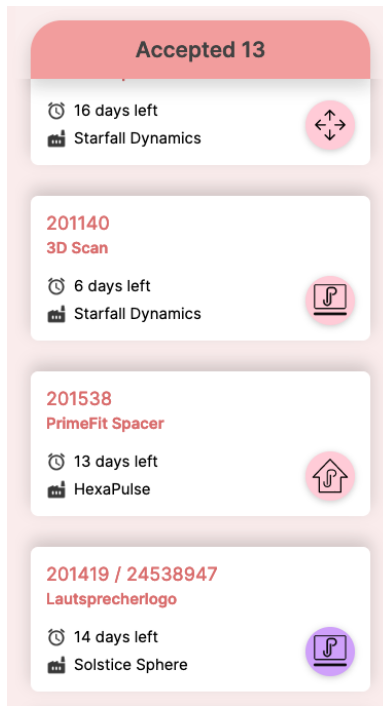


Figure 5.5: Scrolling Behaviour Board

When scrolling down, the column title is kept sticky at the top and remains visible at all times. As shown in the example, the items disappear under the title with the shadow of the title providing a feeling of depth.

Card

The cards do not have any major differences to the designs. Without any view settings, a card looks as shown by Figure 5.6.

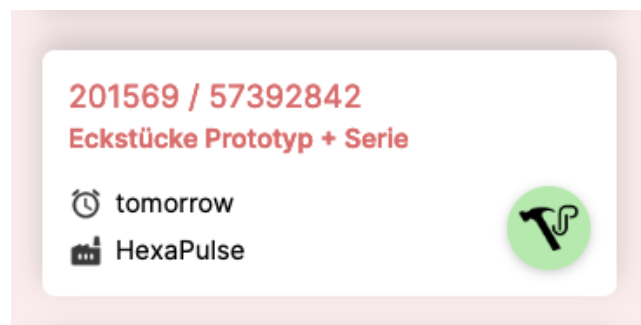


Figure 5.6: Card on Board

The card design focuses on providing the key information that the users are looking for without providing too much detail.

View options

One of the biggest improvements for our users, compared to their in-place solution was introduced through the view options. The view options are found in the bar between the header and the content area. The idea behind the views is that the team can define and share certain view configurations, which are useful for multiple members. An example of a view bar could look as shown by Figure 5.7.



Figure 5.7: View Bar

In this example, there are two views available that the user could select. Additionally, there is the option to choose the quick view, as not all view configurations need to be persistent, and sometimes are only relevant in a certain situation. To make changes to the view settings, they can be expanded, triggering a sliding animation. The expanded view settings are shown in Figure 5.8.

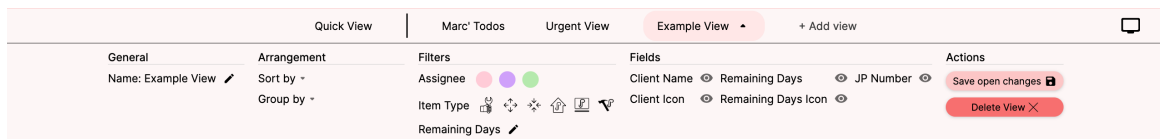


Figure 5.8: View Settings Expanded

The selected view is signaled by the pink background color of the view name in the view bar, here "Example View". The view bar is separated into the sections General, Arrangement, Filters, Fields, and Actions. In the General section, the name of the view can be set and edited. The Arrangement section addresses the order and the way the items are displayed on the board. Filters determine which items are displayed on the board and which ones are filtered out. In the Fields section, the user can determine which fields should be displayed on the cards, while the Actions section is responsible for saving and deleting views.

Arrangement

With the sort-by option, users can sort the items by assignee, due date, item type, and order number, both ascending and descending. The group-by-option allows for a clearer separation into groups and provides the possibility to group items either by the assignee or by item type. Figure 5.9 shows how the board looks when the items are grouped-by assignee.

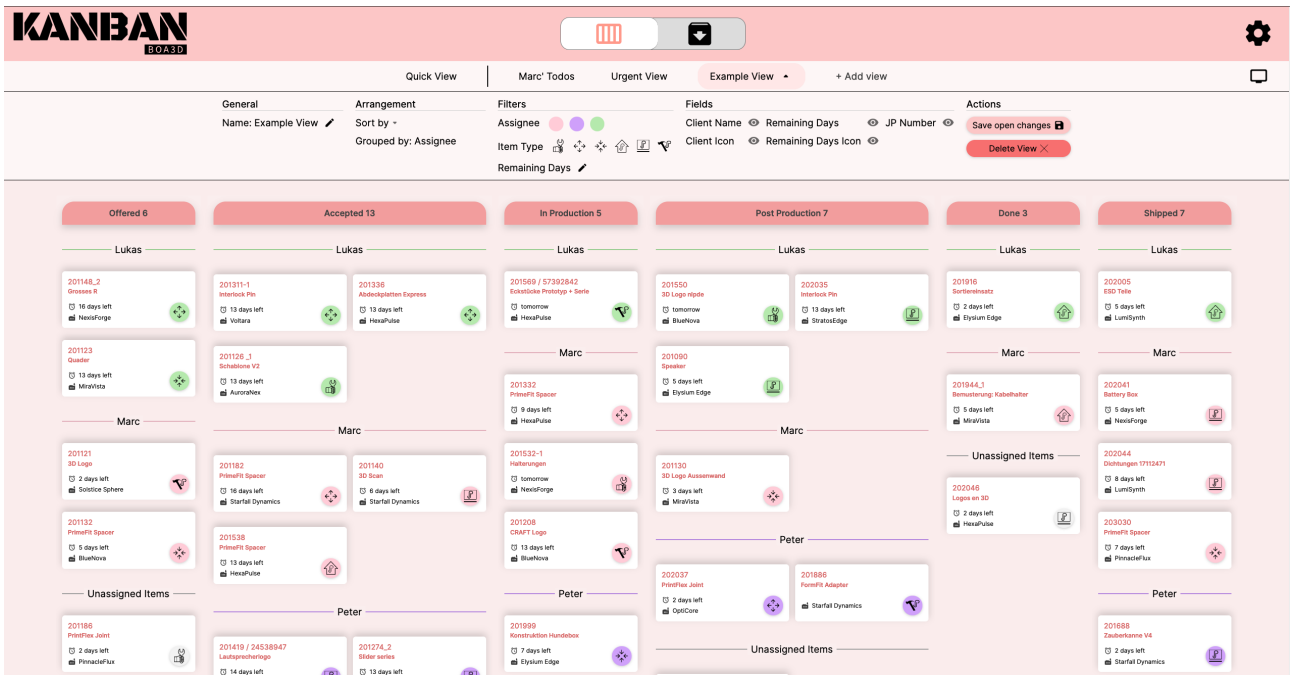


Figure 5.9: Board Grouped-By Assignee

When items are grouped by assignee, a separator is introduced consisting of the name of the assignee and a horizontal bar in the color of the user. Grouping by item type is shown in Figure 5.10.

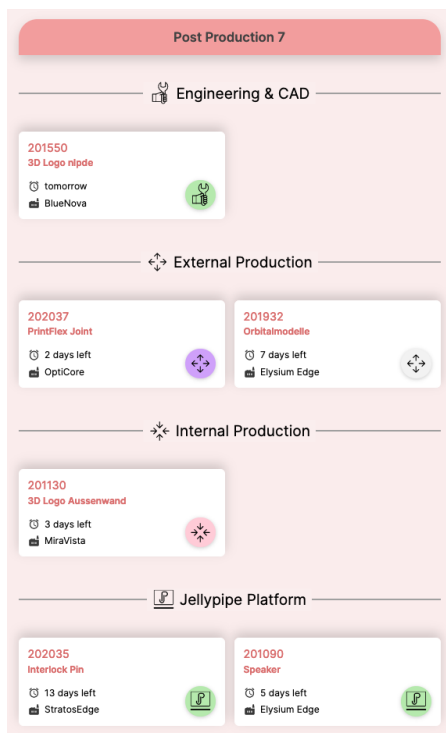


Figure 5.10: Board Grouped-By Item Type

When grouping by item type, the separator is not colored but displays the type icon as well as the name.

Filters

The filter options range from assignee and item type to the remaining days. This allows a user to only display items of specific users and certain item types that are due within a number of days. An example of filters is shown in Figure 5.11.

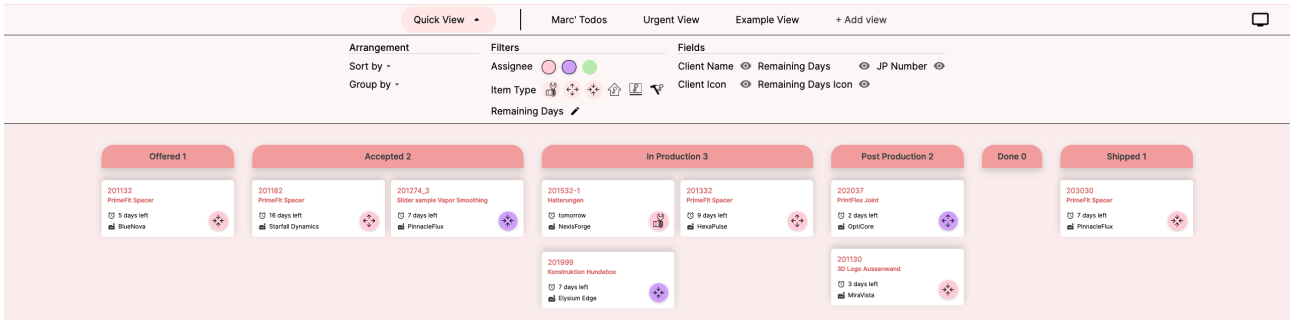


Figure 5.11: Filtered Board

This example shows the board filtered by assignees and item types. The selected assignees are signaled by the border around their color and are the pink and purple users. As for the item types, they are highlighted by the pink background color. The filtering functionality addresses and fulfills the requirement Kanban Board - Filtering - Assignee by allowing the user to filter by assignee.

Fields

The field options, allow the user to choose which pieces of information should be displayed on the board. By hiding fields that are not relevant in a certain situation, users can remove unnecessary clutter from the board which then also allows for more items to be displayed on a page as shown by Figure 5.12.

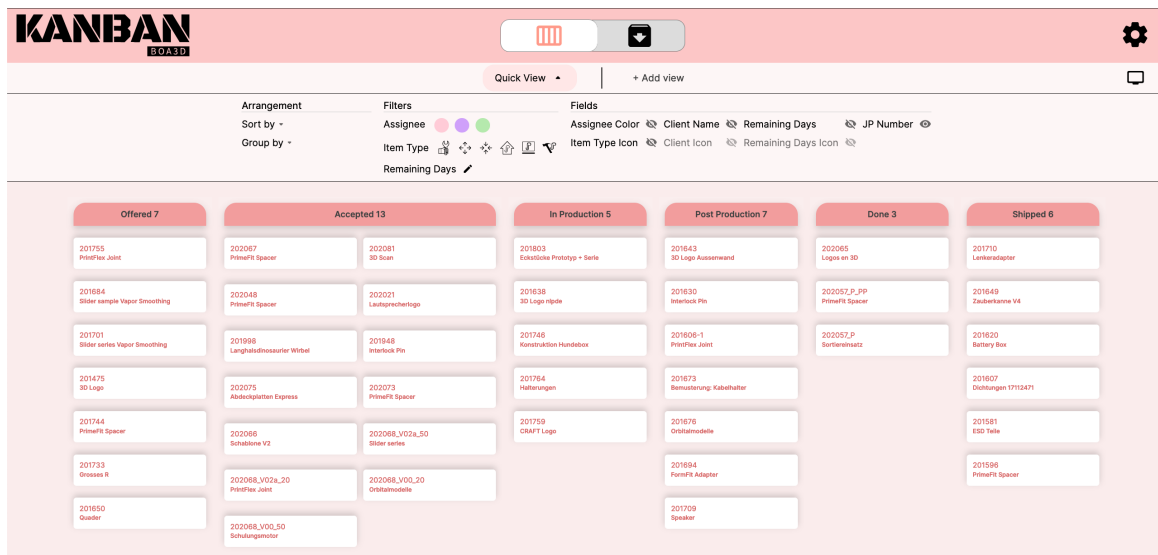


Figure 5.12: Board with Minimal Cards

This figure shows the board with everything but the item number and the title hidden. Here we implemented negative signifiers, by lowering the opacity of "Client Icon" and changing the cursor when "Client Name" is disabled.

View setting use cases

Where the view options become particularly helpful is when they are combined. One example can be having a set of settings to display the most urgent issues as shown by Figure 5.13.

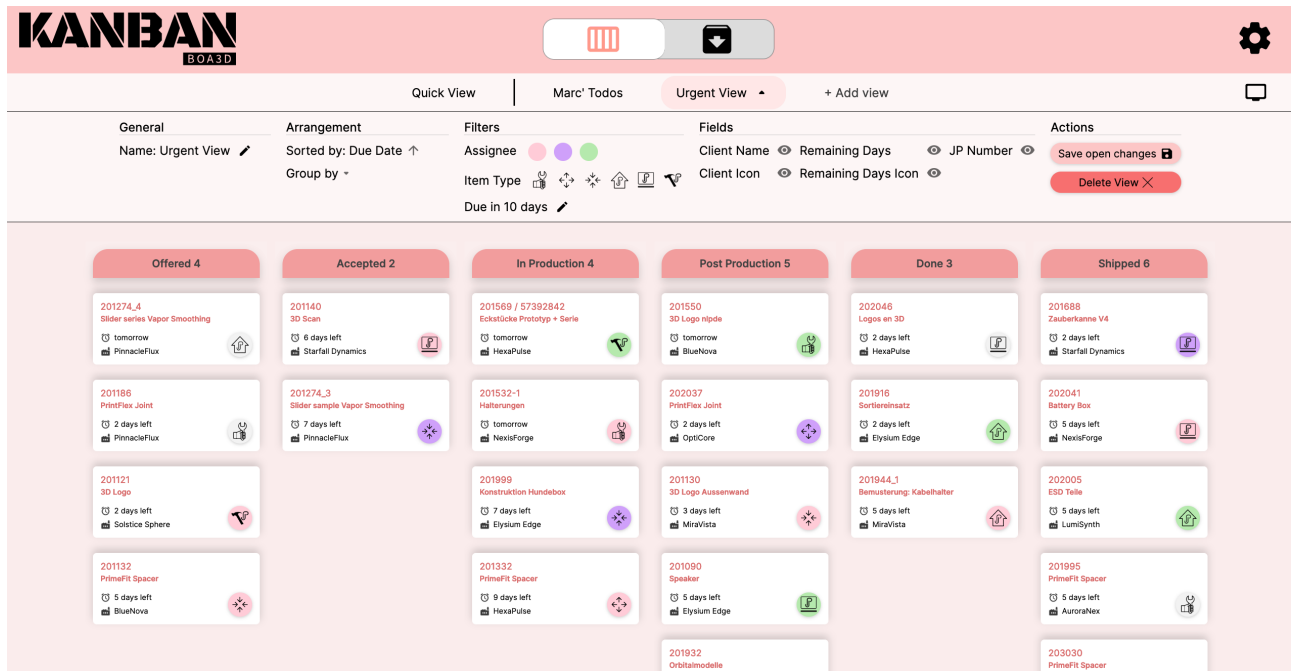


Figure 5.13: View Settings Expanded: Urgent View

In this example, the orders are sorted by the due date, as seen in the "Arrangement" section and filtered to only show items that are due in the next ten days. This addresses one of our partners' main pain points and helps them keep track of urgent issues which was also defined as the requirement Kanban Board - Filtering / Sorting - Due Date. Some other supported use cases could be:

- **Personal View:** By having a personal view that only shows items that are assigned to you, each member can keep track of his items.
- **Stand-Up View:** Grouping by assignee could be a great fit to discuss the state of affairs in a stand-up format as the team can go over the items of each user separately.
- **Production View:** The team member mainly responsible for the printing could filter the items which fit his profile, which helps him find new orders to work on.

5.1.3 Detail View

As the cards on the board only show the most important pieces of information about an item, we introduced a detailed view to provide space for more information. The detail view is opened by clicking on a card on the board and an example is shown in Figure 5.14.

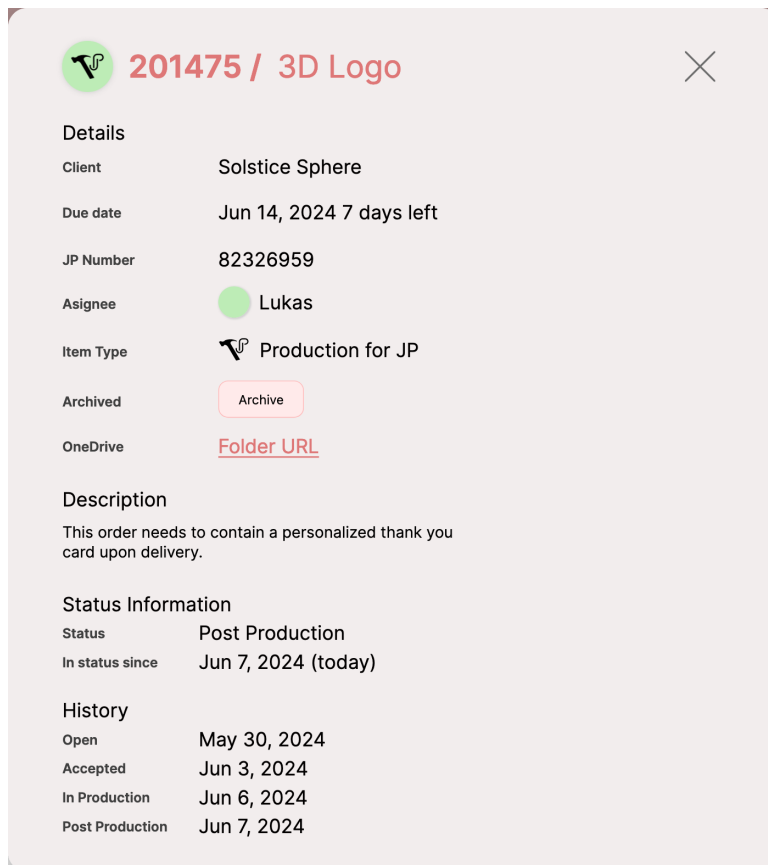


Figure 5.14: Detail View

As shown, the detail view largely corresponds to the designs we made. This component of the tool fulfills the following requirements we defined:

- **Card Details - Due Date:** The component provides the possibility to put additional relevant information about an item down, through the description field.
- **Card Details - JP Number:** The JP Number field allows the users to put down the JP Number of an order.
- **Card Details - Assignee & Card Details – Reassign:** Through the assignee field, we enabled the users to set and change the assignee of an item.
- **Card Details - Description:** The component provides the possibility to put additional relevant information about an item down, through the description field.
- **Card Details - History:** The history section, shows the viewer in which states the item previously was and when they were moved.

Additionally, this component allows users to set the type of the item. Another feature we implemented is the link to the One Drive folder corresponding to each order which provides another shortcut to the users. With the One Drive Integration, we were also able to embed a 3D render of the part that needs to be printed as shown by Figure 5.15.

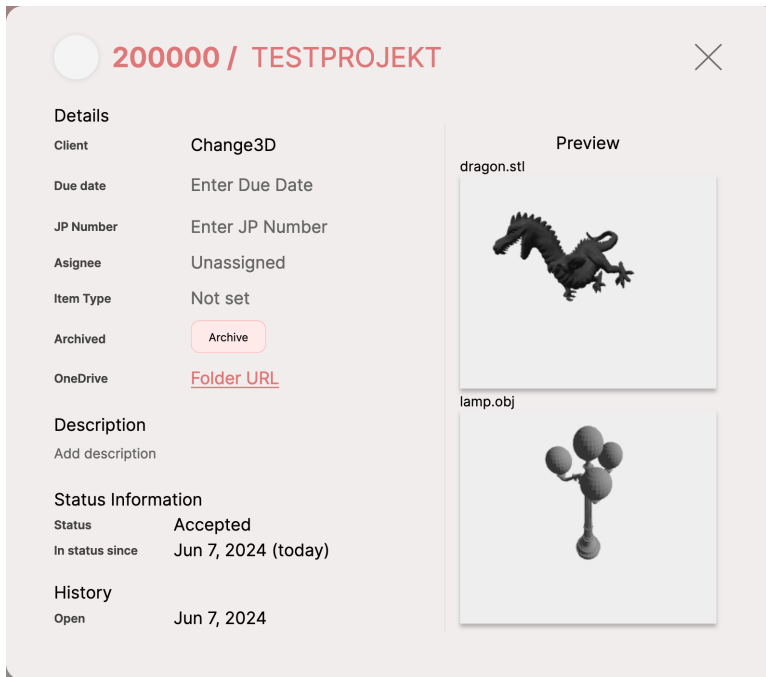


Figure 5.15: Detail View with Preview

The renders shown in this example of the dragon and the lamp can dynamically be moved and viewed from each perspective and help provide more of an association between the order and the object that should be printed. Displaying the object, retrieved from One Drive, addressing requirement OneDrive / Card Details - 3D Models.

5.1.4 Archive

The archive page addresses the requirement Archive - Overview by displaying all the items that have been archived. An example of how the archive could look is shown in Figure 5.16.

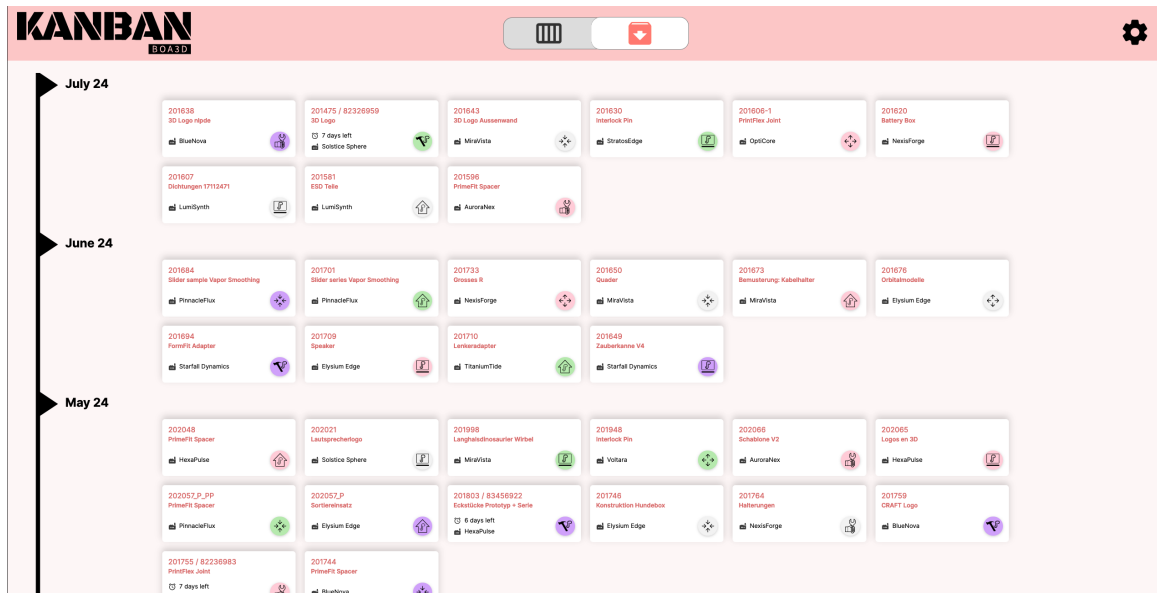


Figure 5.16: Archive

The archive displays the items sorted by when they were archived. Additionally, we split the items up based on the month they were completed to provide a better sense of when which set of items were archived. This also provides the viewer with a sense of how many items were completed in each month which could provide the team with a sense of joy at the end of a hard month completing loads of issues. We also see this format as fitting for a retrospective, as the team sees which issues have been completed in each month.

5.1.5 Full Screen

The full-screen mode is accessible through the full-screen icon on the view bar and displays only the board itself. It aims to provide a fitting view without any clutter for the team to look at the board together and discuss items and address requirement Kanban Board – Full Screen. The full-screen applied view is shown in Figure 5.17.

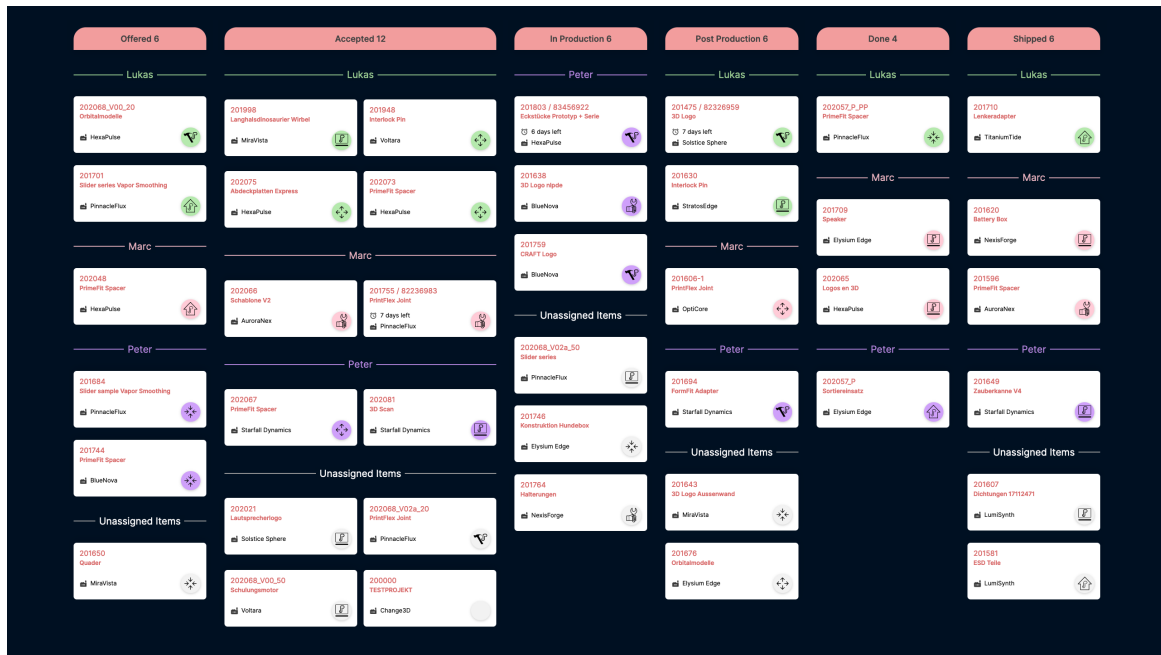


Figure 5.17: Full Screen

In this example the items are grouped by assignee to resemble how the board could look like during a stand-up meeting of the team. The finished version of the full-screen view looks significantly different from the initial design as it was restyled during the course of the project. We decided to use a dark blue background instead of black to prevent viewers from getting eye strain when looking at the board for an extended period of time.

5.1.6 Interactions

A special emphasis in our project was on the interaction design. Something we implemented through the application was giving visual confirmation to the user after a state transition. One example can be the shining animation we trigger when an item is moved to a new state. Upon a state transition, we also show the moved item as the first item in the column to always have it in the user focus. This change was implemented after we identified it during the exploitative user test with the client. This animation is also triggered when someone changes an item or moves an item on a different client, so the user is visually notified about changes.

Additionally, we implemented shortcuts that make certain interactions easier for the user. When an item is in state done or shipped, we show an archive button when the user hovers over it as shown in Figure 5.18.

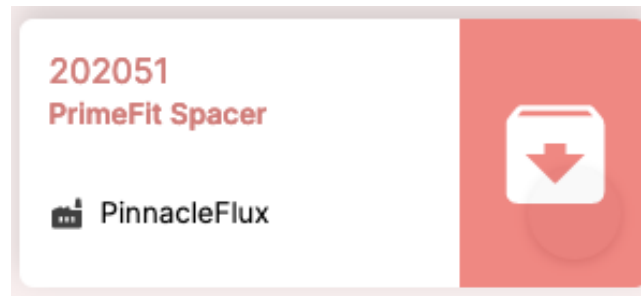


Figure 5.18: Archive Shortcut

The archive button which is now visible allows the user to move an item to the archive without having to expand the detail view first. As moving an item to the archive is most relevant when it is already in the state done or shipped, we only enabled the functionality for those states.

5.1.7 OneDrive Integration

Another part of our result is the OneDrive integration, which is only partly visible in the user interface and mainly in the backend. When an order is first created, the folder structure on OneDrive is created, eliminating manual steps and addressing requirement OneDrive - Folders. Additionally, the project PDFs which previously had to be manually downloaded from Bexio and stored in the One Drive are now automatically fetched and stored, fulfilling requirement OneDrive - Pdf.

The visible part of the OneDrive solution can be seen through the link to the corresponding OneDrive folder shown in Figure 5.14. Additionally, the display of the 3D model is shown in Figure 5.15.

5.1.8 Differences to Design and Requirements

In this section, we want to discuss the differences between the final result compared to the design and requirements.

Design: One of the main differences between the designs and the finished result is the overall design. During the course of the project, we realized, that the look and feel of our page are okay but also rather stale. Therefore we decided to invest resources into a general overhaul to transform the page into a more modern-looking application to provide the users with more satisfaction when using the tool. This led to other features being moved down in the priority list, which was worth it to us, as it was something we wanted to do, as we previously only built functional user interfaces and none, we liked the look of.

Search Bar: One of the pieces missing in the result is the search bar. During the exploitative user test, when the feature was not implemented yet, we could observe the subjects using the browser search function intuitively. Therefore we decided, that the search bar would not be implemented, to make place for other features. Especially because we intended to implement a rather complex search solution, this allowed us to save a lot of resources. The full-text search of the browser is already a suitable option, as it allows one to search for an item by title, customer, order number, and JP number. This means requirement Search is only somewhat fulfilled, as the search to the extent we intended is possible, just with browser functionality not through the dedicated functionality we implemented.

Mail Functionality: Another feature we thought about early on that did not come to fruition was the mail functionality. We could have integrated a mail gateway to send out reminder mails to the users, for example when an order is close to the deadline, but ultimately decided not to. We gave this feature a lower prioritization, as we have integrated mail gateways a fair few times in the past and

we did not see it as a particularly challenging task nor as a task with a significant learning effect for us. As we wanted to try out new things and our partners did not have it high on their priority list either, it ultimately did not make the scope. Additionally, with all the view options we already made strides to make it easier to identify urgent orders so we saw the demand as met and we did not want to integrate another component for the sake of having another system on our context diagram.

Card Details - Quantification: The intended solution we designed was undoubtedly a very useful feature which corresponded to the requirements Card Details - Quantification and Card Details Quantification Update. As it was quite a big feature that would have taken a lot of resources and had a lot of prerequisites like the Bexio integration as well as the One Drive integration it would have had to be implemented late on. This could have clashed with our plan to have a scope stop and a phase where we only introduce fixes and implement feedback from the user tests. Ultimately we gave the One Drive integration, which we gave a more extensive scope during the course of the project, higher priority as that feature is useful for every order and the quantification feature is only useful for a small subset of orders. With more resources, this feature would be implemented as one of the first.

Invoicing: In the initial design we had intended to introduce functionality to help keep track of invoicing as covered by requirement Kanban Board - Invoice. Ultimately this feature did not make the cut as it was neither prioritized by us nor the client. It would not be a big feature to implement and would be one of the first to be implemented if there were more resources.

5.2 Summary and Outlook

Summary

Overall, we are more than happy with how the project turned out. Our project definition stated the goal of implementing a Proof-of-Concept. From the start, we aimed higher and committed to our own goal of implementing and providing our partners with a running solution, which is an upgrade to their current solution. As our partners have already deployed our application and are using it successfully we exceeded the goal from the project definition and met ours. In the words of our partners CHANGE3D from the testimonial which can be found in German in Appendix B, our software significantly improved their efficiency and did not only improve their internal processes but even fundamentally changed their way of working.

The process we chose, had to goal to allow us to maximize the synergies between us and the partners, by involving them intensively. We hoped to gain a deeper understanding of their context by doing that. In our opinion we achieved those goals which tremendously helped us achieve a result we are proud of. Our partners share the same sentiment by saying, "The communication was outstanding from the first contact to the final implementation. They showed an understanding of our requirements and developed a tailor-made solution, which exceeded our expectations". One point that we are on the fence about, is that we could have planned when to introduce which feature more extensively. This would have been a violation of our agile way of working, which brought other benefits and we strictly did not want to violate and undermine our processes. Therefore if we had to start we would not make any amendments to the process.

From a technical point of view, in the Frontend we decided to limit the usage of libraries strictly and generally built all components from the ground up. We hoped by doing that, we would have fewer problems customizing our components which we encountered previously when we strongly depended on external components such as Material Angular and often ended up fighting against them. Generally, this worked out great and it was the right decision for us in hindsight. Where this came in negatively and where we should have done better, is with some input fields and forms. Especially in the detail view the inputs and the presentation of data are not up to the standard we wanted to achieve. Even though the User Interface can not be broken, in some places with absurdly long pieces of data, the presentation is not as expected. This was due to us copying some code from the *Figma* which looked great on *Figma* but did not perform as nicely in some edge cases. During the project, this problem was already identified and addressed, as we switched to always implementing the components without copying the code from *Figma*, other than some stylings.

We did not choose to implement the Frontend to support mobile devices, which was an optional goal of the project definition, as it was not high on neither our nor on the partners' priority list, as it is not a use-case for them. We also did not implement a search function, as we identified in the user tests, that the browser search function was intuitively used and is sufficient. To round off the design, this would still be a nice feature.

One thing we could have done earlier in the project was the deployment at the partners' office. This would have allowed our partner to test the software even more extensively. By deploying only a few weeks before the end, we could have run into problems, if there would have been major show-stoppers that were only identified after deployment. By involving the partner in the whole process we at least did not run the risk of the software turning out differently to their expectations, but technical problems could have had a negative impact.

Outlook

On top of the backlog in terms of features, would be the search and quantification functionality. Especially the quantification feature, which we already designed, would bring substantial improvements to their workflows when an order contains multiple parts or parts multiple times. Other than that, we could polish the Frontend and introduce things like tooltips in some places.

When thinking about commercializing our project as a product it could be generified, to make it applicable to more use cases. We could add support for more sources other than just Bexio and add support for other file storage options other than OneDrive. Building up on that, the workflow could be made configurable, to allow for different states and state transitions. As we emphasized loose coupling in our implementation, this feat could be achieved with without having to rewrite much of the existing code.

Bibliography

- [1] “What is user centered design (UCD)? — updated 2024,” <https://www.interaction-design.org/literature/topics/user-centered-design>, Jun. 2016, accessed: 2024-5-24.
- [2] J. Nielsen, “10 usability heuristics for user interface design,” <https://www.nngroup.com/articles/ten-usability-heuristics/>, Apr. 1994, accessed: 2024-5-24.
- [3] J. Yablonski, *Laws of UX using psychology to Design Better Products & Services*. O’Reilly Media, Inc., 2024.
- [4] J. Cao, *Interaction Design Best Practices*. UXPin, 2020.
- [5] 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.
- [6] 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.
- [7] L. Eder, “Java, sql and jooq.” Nov 2023. [Online]. Available: <https://blog.jooq.org/>
- [8] “Cloud computing patterns,” 2020. [Online]. Available: https://www.cloudcomputingpatterns.org/tenant_isolated_component/
- [9] R. C. Martin and J. O. Coplien, *Clean code: a handbook of agile software craftsmanship*. Upper Saddle River, NJ [etc.]: Prentice Hall, 2009.
- [10] R. C. Martin, *Clean Architecture: A Craftsman’s Guide to Software Structure and Design*, 1st ed. USA: Prentice Hall Press, 2017.
- [11] 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>
- [12] May 2021. [Online]. Available: <https://community.auth0.com/t/how-to-move-from-development-key-to-production-key-for-tenant/62860>
- [13] D. Syer and P. Webb, Nov 2023. [Online]. Available: <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>
- [14] M. Fowler, “Richardson maturity model,” Dec 2023. [Online]. Available: <https://martinfowler.com/articles/richardsonMaturityModel.html>
- [15] D. A. Norman, *The design of everyday things*. Basic Books, 2021.

Appendix A

Project Definition

Order Management Tool für 3D-Druck Dienstleistungen

Beteiligte Personen

Diese Arbeit wird verfasst von

Joel Sauvain, joel.sauvain@ost.ch

Noah Stalder, noah.stalder@ost.ch

Betreuer dieser Arbeit

Prof. Dr.-Ing. Frieder Loch; frieder.loch@ost.ch

Themensteller

Markus Grimm, markus.grimm@change3d.ch

Experte

Dr.-Ing. Felix Ocker

Problembeschreibung

Change3D ist ein Dienstleister für verschiedene Services rund um den 3D Druck (z.B. Beratung, Druck, Konstruktion). Die Firma expandiert und möchte das bestehende analoge System zur Auftragsverwaltung digitalisieren. Es soll ein Webfrontend (Tablet, PC, evtl. Smartphone) entstehen, dass mit dem Buchhaltungssystem verknüpft werden kann. Die Arbeit soll die folgenden Teilaspekte behandeln:

- Anbindung an Buchhaltungssoftware Bexio
- Generierung von Aufträgen, wenn ein Angebot akzeptiert wurde
- Kanban Board, bei dem die Karten per Drag&Drop verschoben und parametrisiert werden können
- Technologieauswahl und Implementierung eines Proof-of-Concept

Formulierung eines konkreten Auftrags

Die Arbeit soll die folgenden Aspekte adressieren. Nach Absprache kann im Verlauf der Arbeit vom beschriebenen Auftrag abgewichen werden.

- **Anforderungsanalyse.** Gemeinsam mit dem Themensteller findet eine Anforderungsanalyse statt. Dabei werden technische Rahmenbedingungen und Prozesse erhoben.
- **Recherche und Abgrenzung.** Es werden bestehende Anwendungen mit einer vergleichbaren Funktionalität betrachtet, um diese vom Ziel der Arbeit abzugrenzen. Relevante Quellen aus der wissenschaftlichen Literatur sind zu analysieren und bei der Lösungsfindung einzubeziehen.
- **Technologieauswahl.** Es werden Anforderungen formuliert, auf deren Basis eine nachvollziehbare Technologieauswahl durchgeführt wird. Grundsätzlich sollen freie und quelloffene Technologien bevorzugt werden.
- **Menschzentrierte Analyse.** Bei der Gestaltung aller Komponenten der Benutzerschnittstelle wird ein menschzentrierter Ansatz verfolgt. Dies beinhaltet z.B. die Entwicklung und Evaluierung von Konzepten mit realen Nutzern.
- **Prototypische Implementierung und Evaluation.** Der gewählte Ansatz wird prototypisch beim Auftraggeber implementiert. Die Softwarearchitektur und das

Ergebnis werden ausführlich dokumentiert. Es soll eine Implementierung mit potentiellen Usern in einer realistischen Umgebung getestet werden.

- **Kritische Reflektion.** Das gewählte Vorgehen wird kritisch reflektiert. Es werden konkrete Verbesserungsvorschläge am Vorgehen und am Ergebnis diskutiert.

Darüber hinaus ist eine geeignete Projektmanagementmethode auszuwählen und zu beschreiben. Die Beschreibung von Arbeitspaketen und einer angemessenen Anzahl von Meilensteinen ist obligatorisch. Die Planung ist zu dokumentieren und regelmässig im Projekt fortzuschreiben. Die Arbeitszeiten werden auf der Ebene der Arbeitspakete erfasst.

Umfang und Form der erwarteten Resultate

Die Ergebnisse der Arbeit (Quellcode der Software inkl. Dokumentation, sowie der schriftliche Projektbericht) werden den Projektbeteiligten zur weiteren Nutzung zur Verfügung gestellt.

Bei der Erstellung des Berichts werden die üblichen Qualitätsstandards für die Erstellung wissenschaftlicher Arbeiten angewendet. So müssen z.B. alle Quellen (z.B. für Definitionen) konkret und spezifisch nachgewiesen werden. Dies gilt auch für alle Abbildungen, sofern sie nicht selbst erstellt wurden. Quellennachweise erfolgen im Literaturverzeichnis im APA-Stil.

Bei der Erstellung des Berichts sollen die [Empfehlungen des Bundes zur geschlechtergerechten Sprache](#) einbezogen.

Anfangs- und Abgabetermin

- Start der Bearbeitung: **Montag, 19. Februar 2024**
- Abgabe: **Freitag, 14. Juni 2024 (17:00 Uhr)**

Zulässige Hilfsmittel und weitere Betreuung

Alle verwendeten Hilfsmittel werden in der Arbeit aufgeführt. Die Betreuung erfolgt durch die genannte Betreuungsperson. Es werden wöchentliche Beratungstermine vereinbart und von den Studierenden protokolliert.

Appendix B

Testimonial

This testimonial was written by Kevin Seliner from CHANGE3D:

Wir sind ein kleines Ingenieurbüro aus Siebnen und haben unser Order-Management bisher mithilfe von Bexio und einer Magnetwand organisiert. Früher mussten wir mühsam Auftragskärtchen schreiben und diese manuell in verschiedene Produktionsstadien (Offerte, Auftrag, In Produktion, Nachbearbeitung, Versand) setzen. Dies war nicht nur zeitaufwändig, sondern auch fehleranfällig.

Dank der sehr guten Zusammenarbeit mit Joel und Noah konnten wir diesen Prozess revolutionieren. Das engagierte Team hat eine Software entwickelt, die automatisch nach Eingabe eines Angebots im Buchhaltungsprogramm eine elektronische Auftragskarte erstellt. Diese Auftragskärtchen lassen sich in verschiedene Stadien per Drag&Drop verschieben, es können Mitarbeiter zugewiesen werden und es werden automatisch Auftragsbestätigungen und Rechnungen generiert.

Von der ersten Kontaktaufnahme bis zur finalen Implementierung lief die Kommunikation mit den Studenten hervorragend. Sie zeigten Verständnis für unserer Anforderungen und entwickelten eine massgeschneiderte Lösung, die unsere Erwartungen übertroffen hat.

Die neue Software hat unsere Effizienz erheblich gesteigert und viele unserer Produktionsabläufe automatisiert. Die mühselige manuelle Arbeit gehört der Vergangenheit an und wir können uns nun auf wichtigere Aufgaben konzentrieren. Joel und Noah arbeiteten äusserst professionell und innovativ, ihre Lösung hat sich als unverzichtbar für unser tägliches Geschäft erwiesen.

Wir sind den beiden Studenten sehr dankbar für ihre sehr gute Arbeit. Die Entwicklung hat nicht nur unsere internen Prozesse verbessert, sondern auch unserer Arbeitsweise insgesamt verändert. Vielen Dank für eure Unterstützung und eure tolle Leistung!

Appendix C

Slides First Design Iteration

Kanban Design Varianten

Noah Stalder & Joel Sauvain

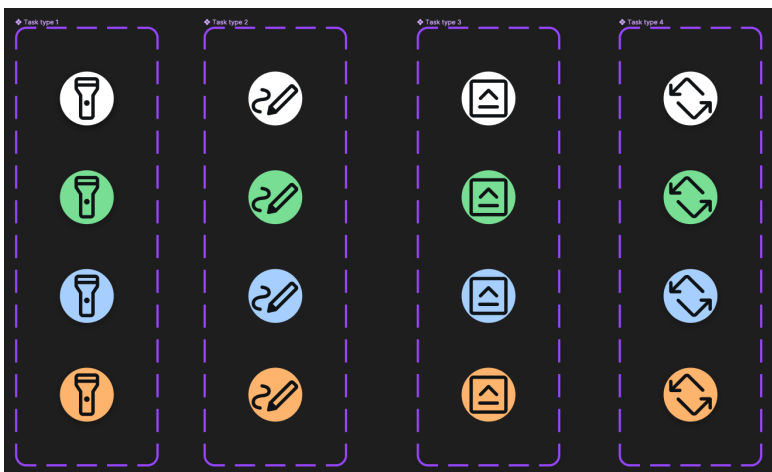
20.3.24



Overview

- Colored Icons
- Anordnung Kanban Board
- Cards (orders)

Colord Icons




Compact eclipse bottom right

201494 /
Auftragsname sehr lang Beispiel
















Remaining days / Due Date
3 days / 18.4.2025

Client name
Globe Solutions AG

JP Number
187420



Anordnungen Kanban Board – Columns

Accepted 7	In production 8
201494 / Auftragsname sehr lang Beispiel Remaining days / Due Date 3 days / 18.4.2025 	201494 / Auftragsname sehr lang Beispiel Remaining days / Due Date 3 days / 18.4.2025 
201494 / Auftragsname sehr lang Beispiel Remaining days / Due Date 3 days / 18.4.2025 	201494 / Auftragsname sehr lang Beispiel Remaining days / Due Date 3 days / 18.4.2025 
201494 / Auftragsname sehr lang Beispiel Remaining days / Due Date 3 days / 18.4.2025 	201494 / Auftragsname sehr lang Beispiel Remaining days / Due Date 3 days / 18.4.2025 
201494 / Auftragsname sehr lang Beispiel Remaining days / Due Date 3 days / 18.4.2025 	201494 / Auftragsname sehr lang Beispiel Remaining days / Due Date 3 days / 18.4.2025 
201494 / Auftragsname sehr lang Beispiel Remaining days / Due Date 3 days / 18.4.2025 	201494 / Auftragsname sehr lang Beispiel Remaining days / Due Date 3 days / 18.4.2025 
201494 / Auftragsname sehr lang Beispiel Remaining days / Due Date 3 days / 18.4.2025 	201494 / Auftragsname sehr lang Beispiel Remaining days / Due Date 3 days / 18.4.2025 
201494 / Auftragsname sehr lang Beispiel Remaining days / Due Date 3 days / 18.4.2025 	201494 / Auftragsname sehr lang Beispiel Remaining days / Due Date 3 days / 18.4.2025 
	201494 / Auftragsname sehr lang Beispiel Remaining days / Due Date 3 days / 18.4.2025 

Anordnung Kanban Board - Rows

Row Label	Card 1	Card 2	Card 3	Card 4
Offered 3	<p>201494 / Auftragsname sehr lang Beispiel</p> <p>Remaining days / Due Date 3 days / 18.4.2025</p> <p>Client name Globe Solutions AG</p> <p>JP Number 187420</p> 	<p>201494 / Auftragsname sehr lang Beispiel</p> <p>Remaining days / Due Date 3 days / 18.4.2025</p> <p>Client name Globe Solutions AG</p> <p>JP Number 187420</p> 	<p>201494 / Auftragsname sehr lang Beispiel</p> <p>Remaining days / Due Date 3 days / 18.4.2025</p> <p>Client name Globe Solutions AG</p> <p>JP Number 187420</p> 	
Accepted 7	<p>201494 / Auftragsname sehr lang Beispiel</p> <p>Remaining days / Due Date 3 days / 18.4.2025</p> <p>Client name Globe Solutions AG</p> <p>JP Number 187420</p> 	<p>201494 / Auftragsname sehr lang Beispiel</p> <p>Remaining days / Due Date 3 days / 18.4.2025</p> <p>Client name Globe Solutions AG</p> <p>JP Number 187420</p> 	<p>201494 / Auftragsname sehr lang Beispiel</p> <p>Remaining days / Due Date 3 days / 18.4.2025</p> <p>Client name Globe Solutions AG</p> <p>JP Number 187420</p> 	<p>201494 / Auftragsname sehr lang Beispiel</p> <p>Remaining days / Due Date 3 days / 18.4.2025</p> <p>Client name Globe Solutions AG</p> <p>JP Number 187420</p> 

Cards


Compact eclipse bottom right

201494 /
Auftragsname sehr lang Beispiel

Remaining days / Due Date
3 days / 18.4.2025

Client name
Globe Solutions AG

JP Number
187420



Minimal Card

201494 /
Auftragsname sehr lang Beispiel

Remaining days / Due Date
3 days / 18.4.2025

