

Group 25 – Transport Company

Managing the supply chain

Business Process Integration Lab

David Galati, Emma Cañavate, Lars Klunder Miguel De La Cruz

4147 words

December 2023

Table of Contents

1. Introduction	2
1.1. Case description	2
2. Approach	2
2.1. Methodologies and tools	3
3. Assumptions	4
4. Requirements	6
4.1. Data protocol with control towers	6
4.2. Data protocol between transport companies	6
5. Design decisions and modelled processes	8
5.1. Microflows	8
5.1.1. Generate a random Order ID	8
5.1.2. POST orders to shared Firebase	8
5.1.3. GET orders from shared Fire Base	9
5.1.4. Create Bid	9
5.1.5. POST bid	10
5.1.6. Published REST API's	10
5.1.7. <i>Accepted Bid</i>	11
5.1.8. Denied Bid	11
6. Set up instructions & Testing	12
7. Reflection and evaluation	18
8. Conclusion	19
References:	20
Appendix A	21
Appendix B	22

1. Introduction

As a case study, the group has studied the operations of Uniprocterlevergamble Ltd., a major producer of Fast Moving Consumer Goods. The company's supply chain involves interactions with transport companies, distribution centers, and a control tower, where each of them play a distinct role in ensuring the smooth flow of goods. For this project, an examination of Uniprocterlevergamble Ltd.'s supply chain has been done focusing on the role of **Transport Companies (TC)**.

1.1. Case description

Uniprocterlevergamble Ltd., a distributor of goods to supermarkets, relies on transport companies like ours to facilitate the movement of their products from distribution centers to the designated supermarkets. In our role as a transport company, we play a crucial part in the supply chain, ensuring timely and cost-effective delivery of goods.

When Uniprocterlevergamble receives new orders, they initiate a request to various transport companies, including ours, where each company can submit their best price for the transportation of those orders. The selection process is based on the pricing proposals received from different transport companies where the chosen one is always the lowest-priced option. When a truck is less than a truck load (LTL), communication between transport companies can take place: If another company offers to carry the additional load at a lower cost than the negotiated price with Uniprocterlevergamble, that company carries both goods in order to make a full truckload (FTL). In contrast, if the offered pricing exceeds our internal costs, we take on the transport responsibilities ourselves.

It is important to note that the decision-making process hinges on a precise alignment of costs and pricing offers where any deviation from this results in a mismatch. This is a result so that Uniprocterlevergamble maximizes efficiency with the best cost possible cost during the distribution process. As a transport company, the challenge and benefit come from creating competitive pricing while maintaining an operational process that secures transport contracts and take part in the seamless flow of goods within the supply chain.

However, a different approach has been created in order to fulfill the necessities of the supply chain: Some changes such as information exchange and communication between actors have been done.

2. Approach

Collaboration is a core activity. Working with the other groups and other stakeholders like the control towers is essential for the well-being of the process. This collaboration is driven by the need for information exchange in this system and the penetration of this system in daily business processes.

Among transport companies there was pooled interdependency (Kumar & van Dissel, 1996). This meant that any participating actor can be plucked out without it interfering with the functionality of the system. In this system communication by standardization is appropriate according to Kumar and van Dissel (1996).

By studying the theory from (Barrett & Konsynski, 1982), we could subtract some benefits of the typology of IOS that are useful during the project. The first benefit of this typology is cost reductions: This comes from reducing personnel, the automation of transmission of orders and checks, and the implementation of standards. Moreover, because of the reduction in error rates, cost displacement and better resource allocation and utilization, it is possible to increase productivity. To finalize, it creates a competitive advantage and gain market leadership or share, what improves product/market strategy. Because of its benefits, this approach has been considered during the realization of the project.

To optimize our role in Uniprocterlevergamble Ltd.'s supply chain, our approach was grounded in robust collaboration and continuous communication with all relevant stakeholders, including other transport groups and control towers. Recognizing the centrality of collaboration in our business strategy, we outlined key assumptions that guided our operations and strategies (detailed in Appendix I).

At the heart of our approach was the emphasis on effective communication as a driver of supply chain efficiency. By establishing clear and standardized data protocols, we streamlined the exchange of information, which is crucial for resolving complex logistical challenges. To this end, we adopted SOAP (Simple Object Access Protocol) and REST (Representational State Transfer) as our primary data exchange formats. These formats were chosen for their widespread usage.

2.1. Methodologies and tools

In order to get a better understanding of the whole process and analyze our main goal, it is crucial to analyze the High-level Business Processes and Primary Processes (See Figure 1) for our actor.

To do so, the whole process was modelled using ArchiMate. Once Uniprocterlevergamble receives an order from the Supermarket, it contacts and selects the transport company to transport the order. Here is where the truck company's order fulfillment process starts. Ideally, the process was initiated when the transport companies received an order. However, to be able to recreate the scenario, the truck company creates a **transport request** from Uniprocterlevergamble. This request encompasses details such as the type and quantity of goods, name of the TC, DC name and supermarket name. In addition, the value of the order, number of pallets and delivery deadline it's added as well. If the cargo is equal to FTL then the truck company will transport the goods itself. However, if the cargo is less than the FTL, a **collaborative process** between other transport companies begins to match the FTL. Depending on the Uniprocterlevergamble and other truck companies' fees, we decide to either pick up both orders or let the other company pick up our order. Once everything is decided, we communicate **with Control Tower** and the order is sent to them. The control tower receives an XML file sent over API with all necessary data. By this said, as a transport companies, there's no contact with the different Distribution Centers, since Control Tower is the responsible for it.

Upon arrival at the Distribution Center, the **loading process** starts. With the goods loaded, we **depart to the destination** (Supermarket). Upon reaching the supermarket, we get **notified by the Central Tower of the truck's arrival**. After unloading, the supermarket notifies the Control Tower of the received order details. We **received confirmation** from the Central Tower that the transport process has been fulfilled or if some goods are missing (See assumptions) (then we might get extra information about how to proceed in those situations). The business processes model diagram is also available in Appendix B.

The tools being used for this project are ArchiMate and Mendix.

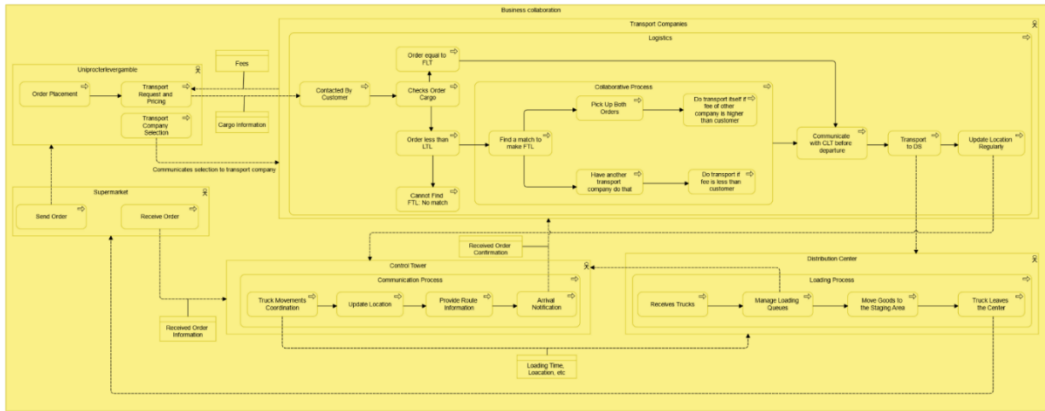


Figure 1 High-level business model

3. Assumptions

Supply chain requires communication between the different parts and actors that are involved during the whole process. To ensure a smooth flow of goods, the different companies have accorded different assumptions, so the cooperation is possible:

1. We receive requests for quotations from Unilever. These requests are always one order for driving a load of goods from one DC to one supermarket. The goods and locations are fixed and will not change.
2. We will send a quotation back to Unilever, with the price for which we will perform the transport. They will choose the cheapest one and accept that one.

Note: As there are no students taking the role of the Uniproctorleverageable group, we can 'skip' the steps above. Therefore, just create a few fictitious agreements with each transport company and Unilever, some which are full truck loads, and some are not full truckloads so we can simulate the integration between the different transport companies.

3. If the request is not a full truckload, we communicate with other transport companies, and see whether they also have non-full truckloads for the same DC to the same supermarket. If this is the case, one of the transport companies will take over the order for the other, if and only if the company willing to take the extra transport does it for less money than the other company received from Unilever. Once this has been agreed, the company that takes over the transport will need to communicate with the control tower.
4. When the company is ready for transport, we will send a request to the control tower with the following information:
 - Truck.
 - Driver.
 - Quantity and type of goods.

- What DC.
 - What supermarket.
 - Price of goods.
 - We pay the CT fixed amount per trip.
- Data structures have also been discussed.

Extra notes:

- *There do not need to be any contracts between the transport companies for taking each other's loads, these are out of the scope.*
- *There is no need for us to communicate with the DC's. Communication is done through CT's who take care of that.*

4. Requirements

As state during the chapter above, some requirements related with data structures have been discussed and stablished, so a common methodology is used between actors and between transport companies.

4.1. Data protocol with control towers

A data protocol with control towers has been created since when a company is ready for transport, this one will need to resend a request to the control tower with the information required.

The SOAP XML request shall be transmitted through the API request to the Control Tower.

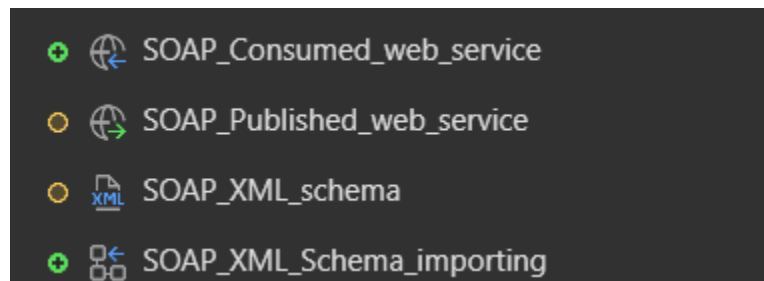


Figure 2 SOAP XML schema and web service objects

The Control Tower should have provided us with a WSDL schema for the SOAP endpoint, but due to unexpected circumstances, this happened very close to the deadline, making it impossible for us to finish the SOAP communication with the CT.

4.2. Data protocol between transport companies

During different meetings between transport companies, a data protocol was stablished. The implementation of a standardized data protocol between transport companies is essential when it comes to improve efficiency and coordination within the supply chain. By establishing a common language for information exchange, the protocol ensures interoperability among the different companies. This facilitates decision-making, minimizes delays, and optimizes delivery routes. Moreover, it is crucial especially when combining shipments for cost savings. In summary, a standardized data protocol not only promotes efficiency and collaboration but also addresses crucial aspects of accuracy and optimization in the transportation companies.

Transport companies created:

1. **Central Hub (Google Firestore)** where all LTLs will be placed. These are all connected to the hub companied with our different messages.
2. **Private communication (API endpoints)** where bids will be done in other companies LTLs. This is done through a REST API endpoint.

Transport Companies can publish their LTL shipment to a centralized hub. Subsequently, a private bidding process is facilitated through an Application Programming Interface (API), enabling transport companies to submit bids on each other's LTLs. Following the bidding phase, the company that originally posted the LTL has the decision to evaluate and accept or reject bids based on pricing considerations. This structured approach ensures efficiency in the LTL allocation process within the transport network.

The uploading LTL to the central hub is as follows:

```
JSON
{
    'order_number': (string, random at least 16 characters),
    'tc_name', (the name of your transport company, string)
    'dc_name', (name of the distribution center, string)
    'supermarket_name', (name of the supermarket, string)
    'amount_of_pallets', (amount of pallets that the transport is,
integer),
        'good_type' , (A, B or C)
        'good_amount': (amount of given good type)
        'total_value', (total value of the goods that you are
delivering, double/decimal)
    'delivery_deadline', (latest time when the goods have to be
delivered to the supermarket, dd-MM-yyyy HH-mm-ss)
    'time_of_upload', (time which you upload this message, dd-MM-
yyyy HH-mm-ss)
    'time_end_of_bidding', (end time which indicates the latest
time that you accept bidding, dd-MM-yyyy HH-mm-ss)
}
```

In addition, all transport companies require of the following 3 API endpoints for private communication:

```
/tc/{tc_name}/order/{order_number}/bid
- With fields:
    'bidding_tc_name' => string (this is the name of
your own transport company, when you bid on the other order)
    'bid_value' => double
    This endpoint sends back a 201 CREATED with a bid_id,
so the company can keep track of the bids it has done.
/tc/{tc_name}/order/{order_number}/bid/{bid_id}/accept
- Use this endpoint to accept a bid. Therefore, the company who
placed the order on the hub will send this to the company that
sent a bid. For this reason, companies need to keep track of
their own bid_ids.
/tc/{tc_name}/order/{order_number}/bid/{bid_id}/deny
- Use this endpoint to deny a bid. Therefore, the company who
placed the order on the hub will send this to the company that
sent a bid. For this reason, companies need to keep track of
their own bid_ids.
```

This is decided by the considering the following assumptions:

- time_end_of_bidding is maximum 1 day before the delivery_time
- We assume that we always know what dc_name and supermarket_name belongs to what distribution center and supermarket.

- We assume to work with full pallets.
- Order numbers are always a **random** string of **at least 16** characters (otherwise applications will break).
- One order contains one type of good of one amount.
- This means goods need to be packaged up in a list when sending to CT.
- tc_name should be consistent throughout everywhere (so same in your app and in the public database)!

5. Design decisions and modelled processes

To set up the system there was a need for a communication medium, some file storing-hub and intra-communicative API's to have contact with other transport companies. The first design decision we needed to make was the implementation of a database. The database chosen was a Firebase. Firebase is a database from google and has 1GB of free data.

Mendix was used to develop the application. We made use of Mendix's microflows and nanoflows that enable us to connect to Firebase and POST and GET on firebase.

5.1. Microflows

5.1.1. Generate a random Order ID

Since we do not want to interfere with other TCs having the same Order ID as us, we created a microflow that generates a random number between 100,000 and 10,000,000, and then it converts it to a string, as the CTs request the order ID to be a string.

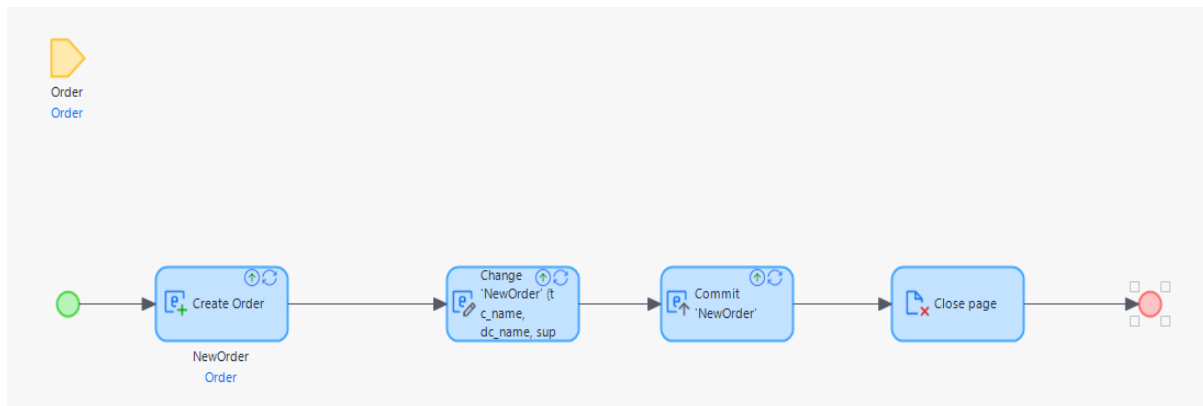


Figure 3 Create order Microflow

5.1.2. POST orders to shared Firebase

To POST an LTL order we send a JSON object to the shared FireBase endpoint so other companies can see its information. First, we create an object in the PostOrderToDatabase entity and fill it with the information of the order we want to POST. Then a POST request is created, and we do an export mapping of the PostOrderToDatabase entity. Instead of using a JSON structure we use Message Definitions in the export mapping.

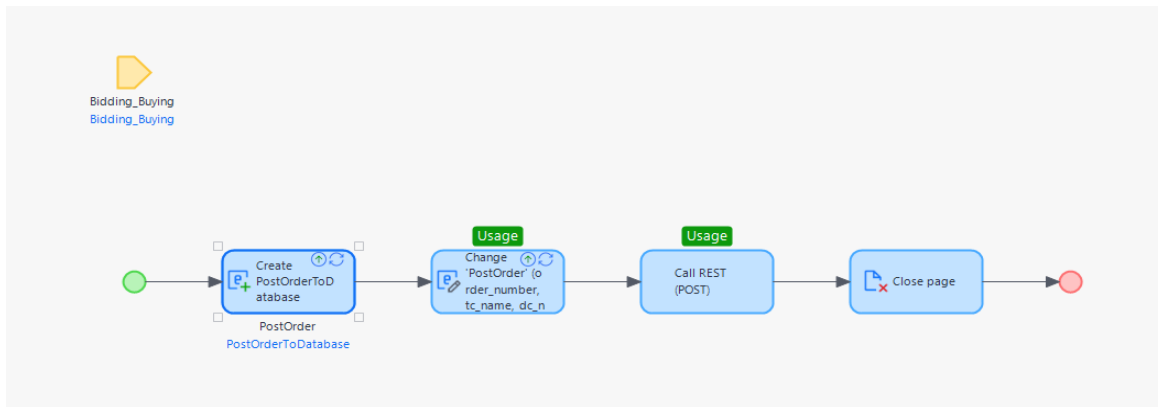


Figure 4 Orders POST Request API Call Microflow

5.1.3. GET orders from shared Fire Base

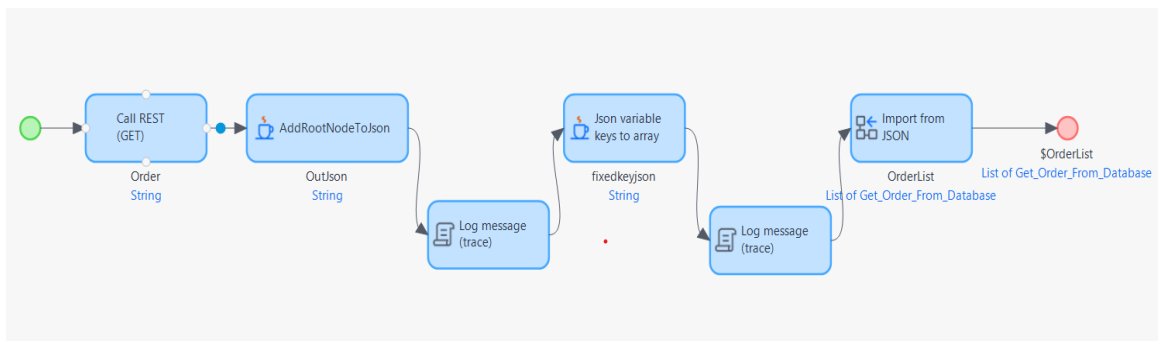


Figure 5 GET Request for orders from share Firebase

Getting the order list from the API endpoint seemed like a trivial thing at first, but it proved to be problematic. Since the backend and database is externally handled by Firebase services, the JSON format that was provided by the other teams used dynamic JSON keys as the identifiers of the requests, but Mendix does not support such mappings, so in order to overcome this limitation, we have found a Mendix marketplace extension called JSON variable key to array [citation needed], but that method needed a parent node as the root, which our JSON format didn't support, so we had to make another Java action called AddRootNodeToJson that just adds a root parent node that can be used In the marketplace Java Action.

5.1.4. Create Bid

This microflow creates a Bid Object with the order_number of the TC that adds their order in the shared database. In our data grid we have a button to bid on that order, so the microflow basically links that bid the

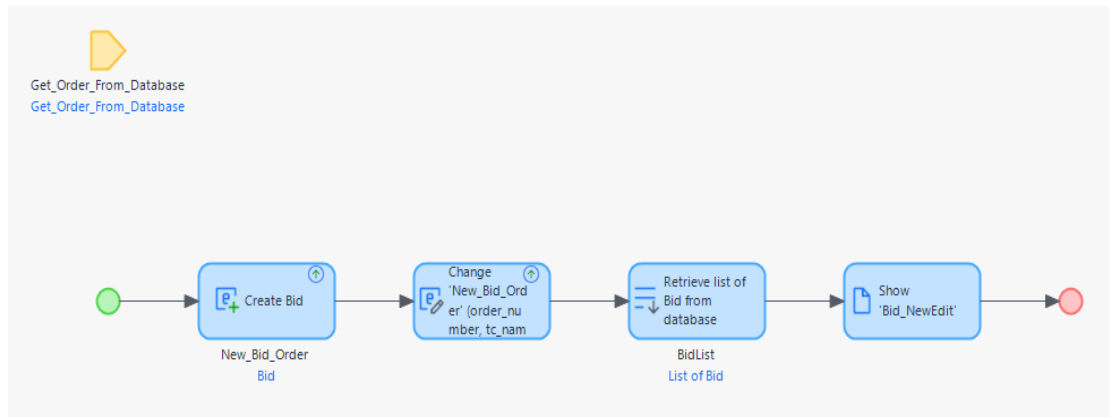


Figure 6 Create Bid Microflow

5.1.5. POST bid

To send a bid to another TC we create a bid object and add it to our Bid entity containing the order_number, tc_name (our TC name), and the bid amount). Then a REST POST request is sent to the receiving TC's endpoint.

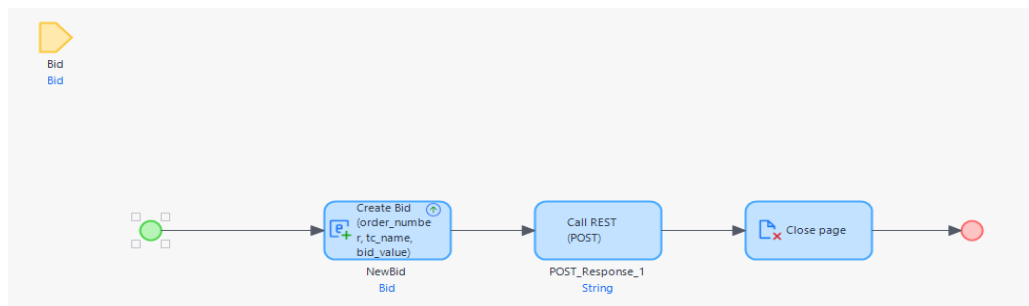


Figure 7 POST Bid Microflow

5.1.6. Published REST API's

Receive bids from other companies. This microflow creates a new object in our ReceivedBids entity when a TC bids on one of our orders in the shared database.

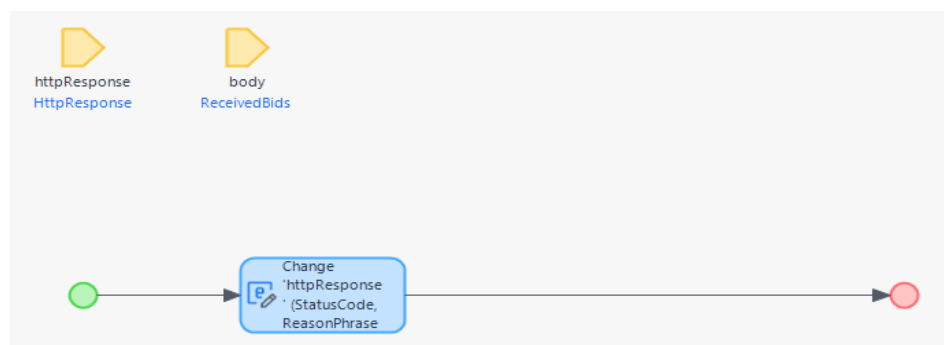


Figure 8 Received bids published API

5.1.7. Accepted Bid

This microflow changes the status of the bids we make on other TCs' orders. When we make a bid, our initial status is waiting. If the other TCs accept it, then they make a REST POST to our endpoint and the object in our Bid database changes the status to accepted.

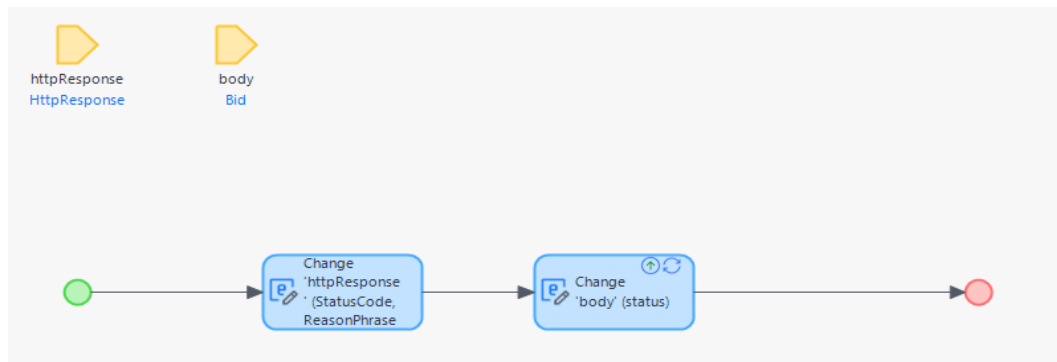


Figure 9 Accept Bid API Endpoint

5.1.8. Denied Bid

This microflow changes the status of the bids we make on other TCs' orders. It has the same logic as the accepted microflow but it changes the status of the Bid to denied when the other TCs call this endpoint.

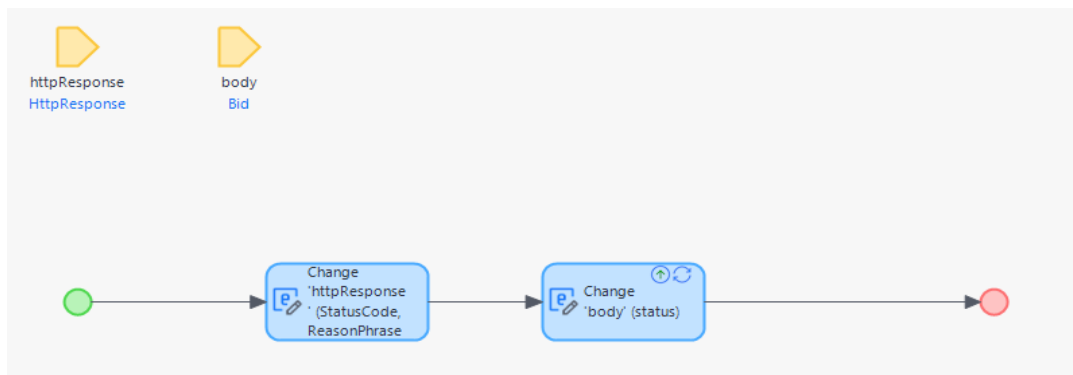


Figure 10 Denied Bid API endpoint

6. Set up instructions & Testing.

The app that has been created, its composed by a main *Home page* that contains the four main buttons: Orders, Bid Market, Drivers and Send Orders. This contains the four main pages that are needed to navigate through the application.

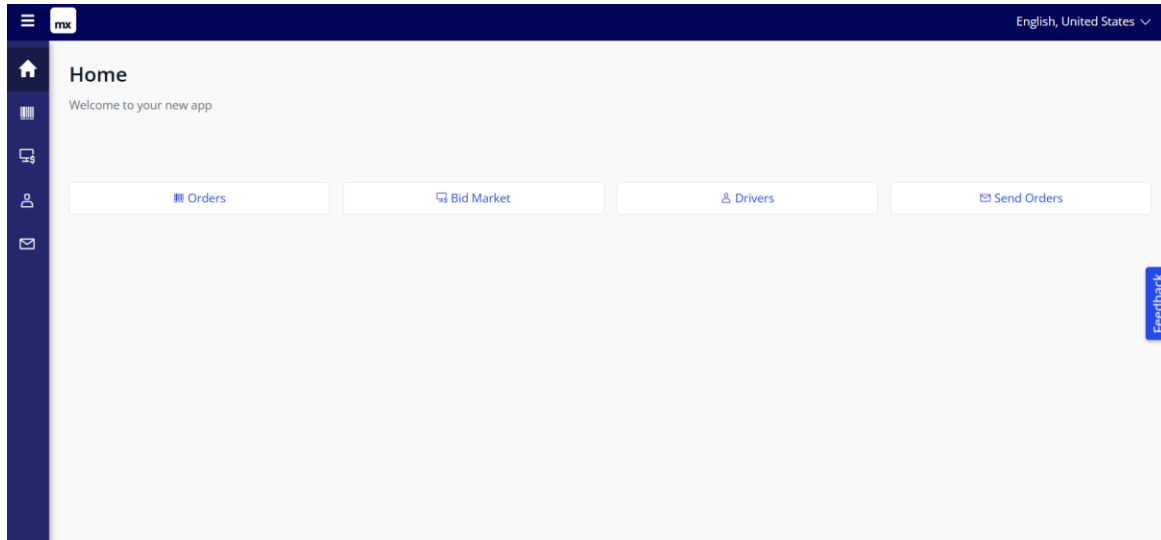


Figure 11 Main App Homepage

When navigating through the first page *Orders* we get the option to generate our orders: This is done by clicking into a button that can be found on the upper-right from the screen. By doing so, a pop-up window where the user can fill the different information about each of the orders. When generating an order all fields should be filled and the delivery date must be a future date, otherwise the user will receive an error message and the order would not be generated. When creating an order, a random order ID between 100,000 and 10,000,000 is generated and converted to a string as the Control Towers require string formats for the order ID's. Moreover, when an order has already been created, it can be modified by double clicking on the list view and deleted by clicking the trash icon.

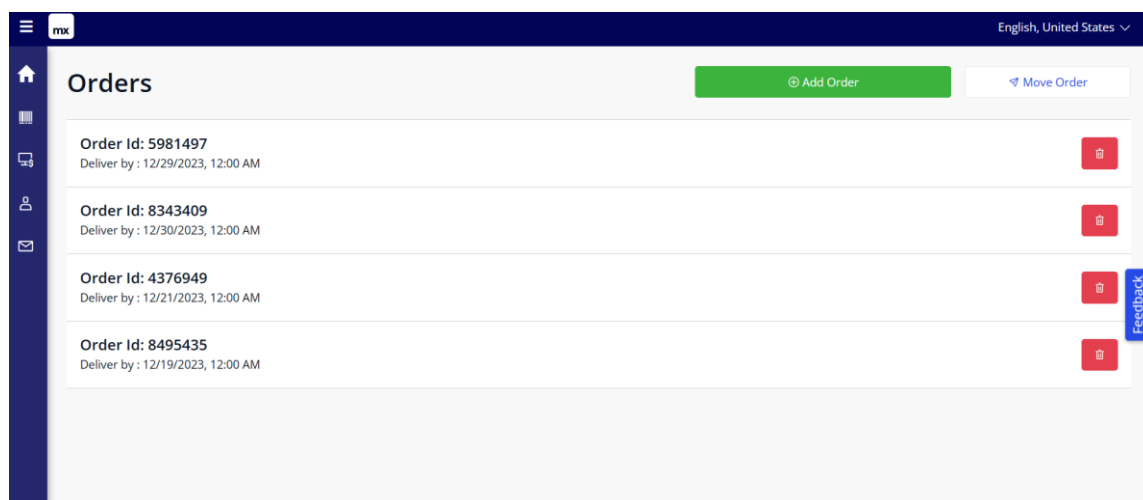
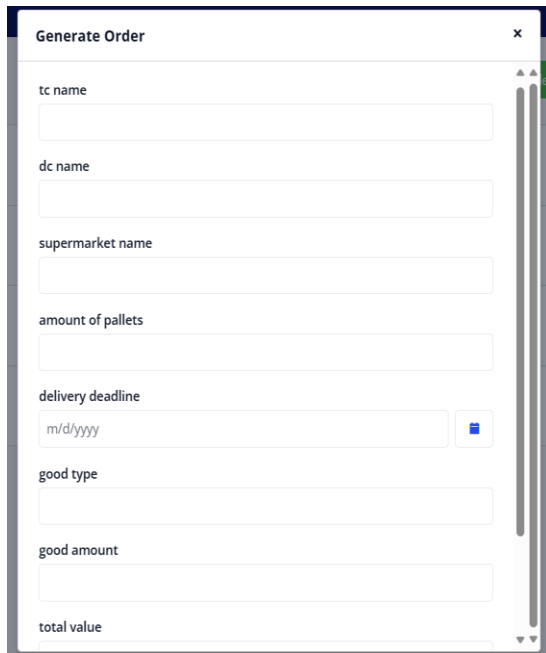


Figure 12 Orders page



Generate Order

tc name

dc name

supermarket name

amount of pallets

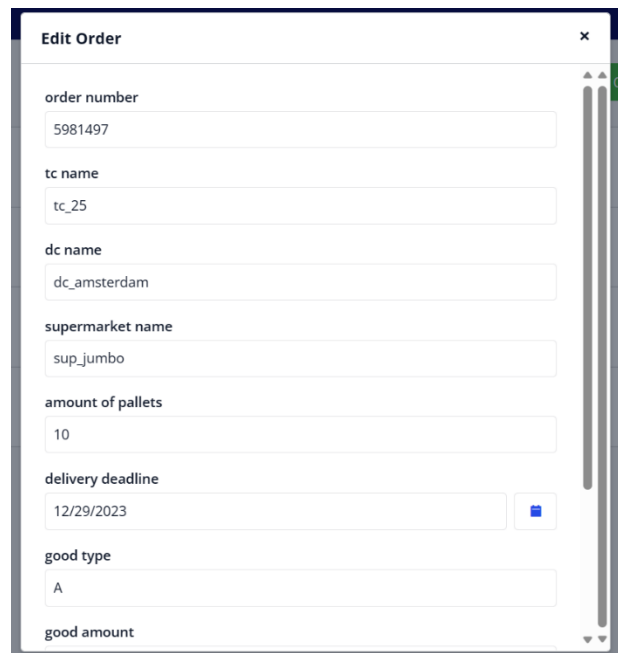
delivery deadline

good type

good amount

total value

Figure 14 Generate Order Page



Edit Order

order number

tc name

dc name

supermarket name

amount of pallets

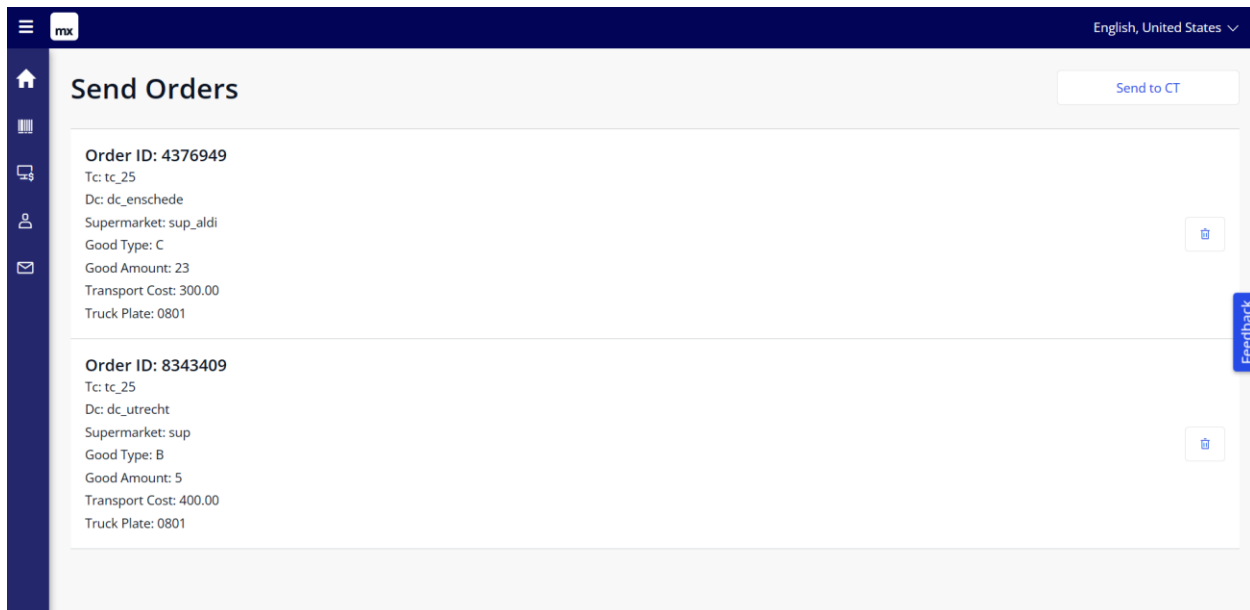
delivery deadline

good type

good amount

Figure 13 Edit Order Page

The page contains an extra button *Move Order*. Another pop-up window appears, where a driver and the transport cost are added. Then, the order is automatically sent to the *Send Orders* page. The page acts as a “shopping cart” from where we send a list of orders to the Control Tower by clicking on the *Send to CT* button.



Send Orders

Order ID: 4376949
Tc: tc_25
Dc: dc_enschede
Supermarket: sup_aldi
Good Type: C
Good Amount: 23
Transport Cost: 300.00
Truck Plate: 0801

Order ID: 8343409
Tc: tc_25
Dc: dc_utrecht
Supermarket: sup
Good Type: B
Good Amount: 5
Transport Cost: 400.00
Truck Plate: 0801

Send to CT

Feedback

Figure 15 Send Orders Page

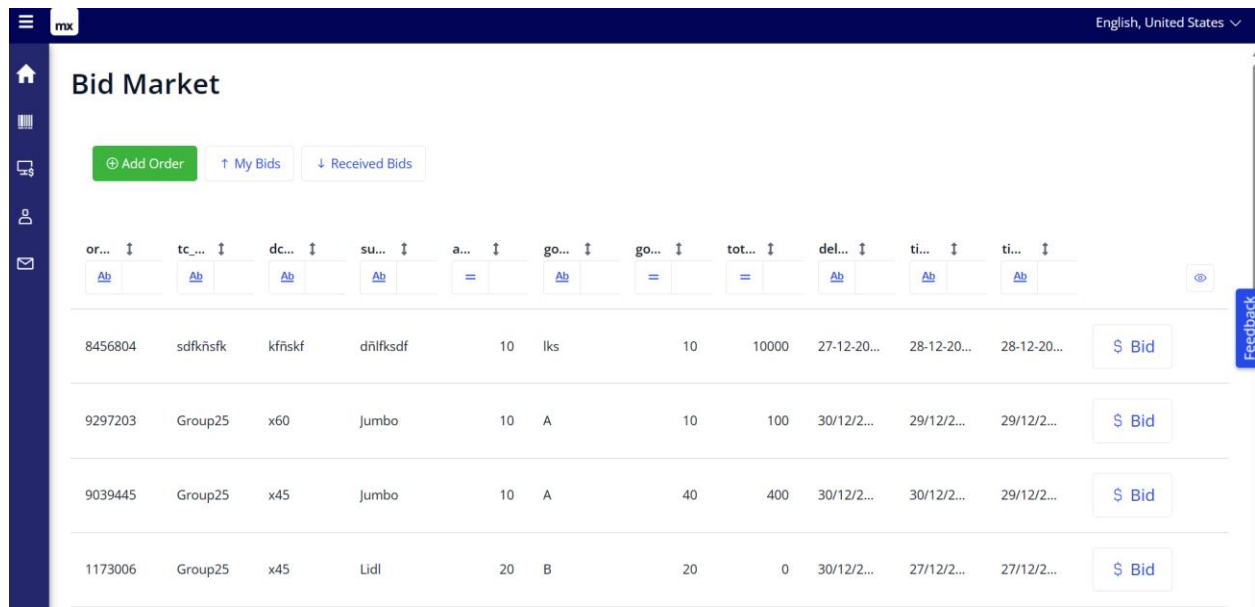
Another main page is the *Drivers*. Here,

different drivers and their information are displayed: Truck ID, the order ID that the truck is linked to, and the driver that will be doing the delivery. It is also possible to add new drivers as we are a growing company.



Figure 16 Drivers Page

The most interesting page of our app is the *Bid Market* page. This page displays the shared database with other truck companies (but in our case it only displays our Firebase database as we developed our own testing app to not have to rely on other groups). In this page we can add bids, post LTL orders, view the bids that we made on other companies' order, and view the bids that other companies have done in our orders. By clicking on the *Add Order* button we post the order to the shared database, where we add our order information plus the upload time and bidding time (time that the other companies must bid on this order). The *Bid* button places on the selected order, that is sent to *My Bids* page so we keep track of the orders we bided on. The *My Bids* navigates the user to the *My Bids* page. There is also a *Received Bids* button that takes the user to the *Received Bids* page where we receive bids from other companies on orders we post.



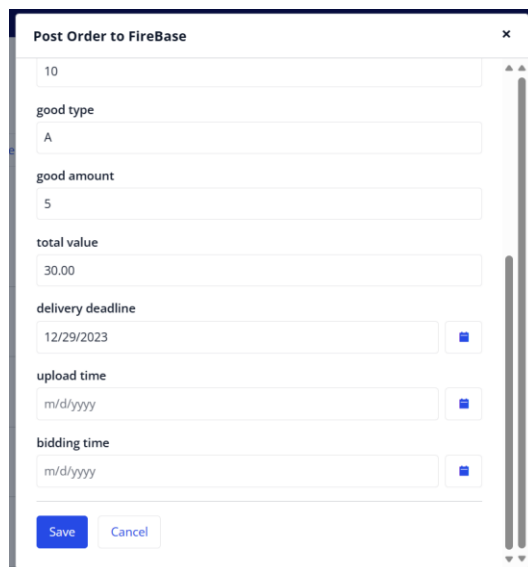
Bid Market

[Add Order](#)
[↑ My Bids](#)
[↓ Received Bids](#)

or...	tc...	dc...	su...	a...	go...	go...	tot...	del...	ti...	ti...	
8456804	sdfkñsfk	kfñskf	dñlfsdf	10	lks	10	10000	27-12-20...	28-12-20...	28-12-20...	\$ Bid
9297203	Group25	x60	Jumbo	10	A	10	100	30/12/2...	29/12/2...	29/12/2...	\$ Bid
9039445	Group25	x45	Jumbo	10	A	40	400	30/12/2...	30/12/2...	29/12/2...	\$ Bid
1173006	Group25	x45	Lidl	20	B	20	0	30/12/2...	27/12/2...	27/12/2...	\$ Bid

Feedback

Figure 17 Bid Market page



Post Order to Firebase

10

good type
A

good amount
5

total value
30.00

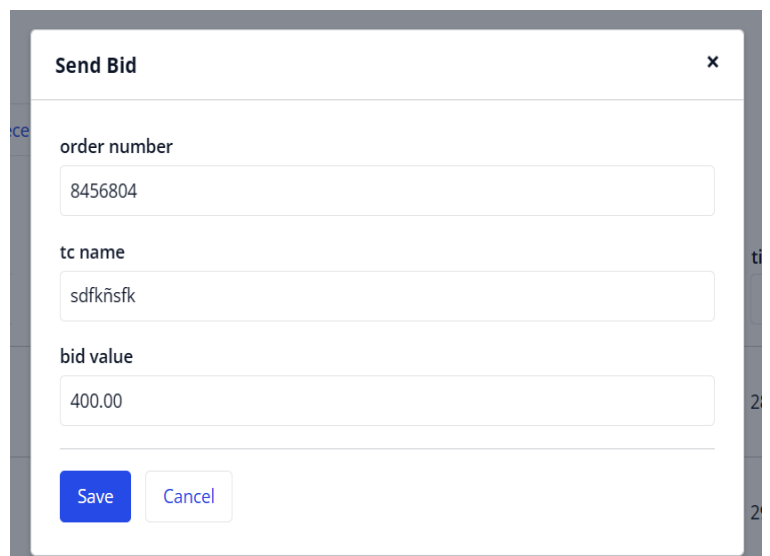
delivery deadline
12/29/2023

upload time
m/d/yyyy

bidding time
m/d/yyyy

[Save](#) [Cancel](#)

Figure 19 Post Order Page



Send Bid

order number
8456804

tc name
sdfkñsfk

bid value
400.00

[Save](#) [Cancel](#)

Figure 18 Send Bid Page



Figure 20 My Bids Page

When it comes to testing, we have created two apps, that can be differentiated by the two colors. During the next screenshots, we can see a testing on how the process of bidding acceptance.

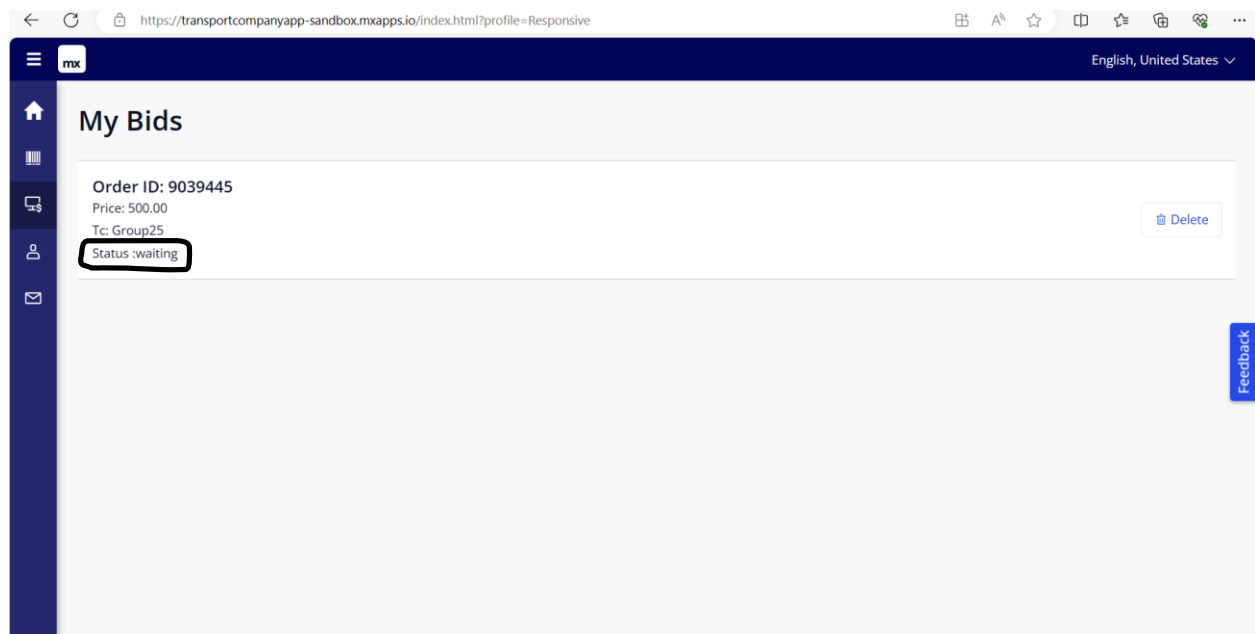


Figure 21 Bid waiting status (waiting/ accepted/ denied)

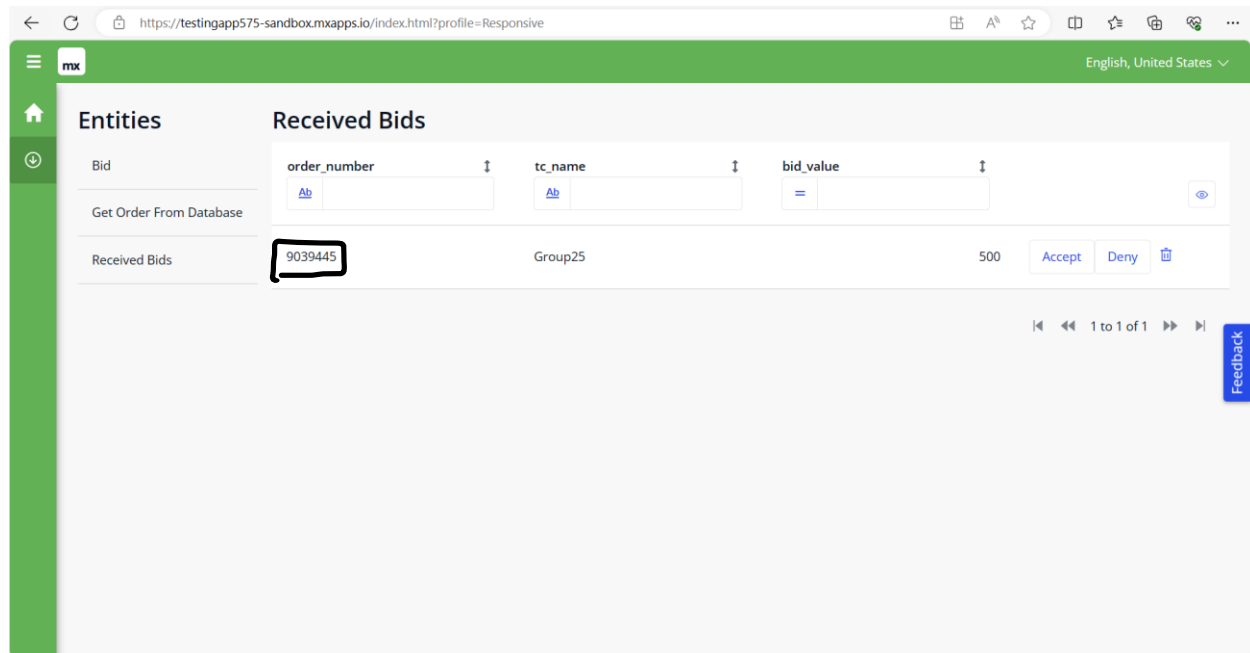


Figure 22 Testing App Received Bid Page

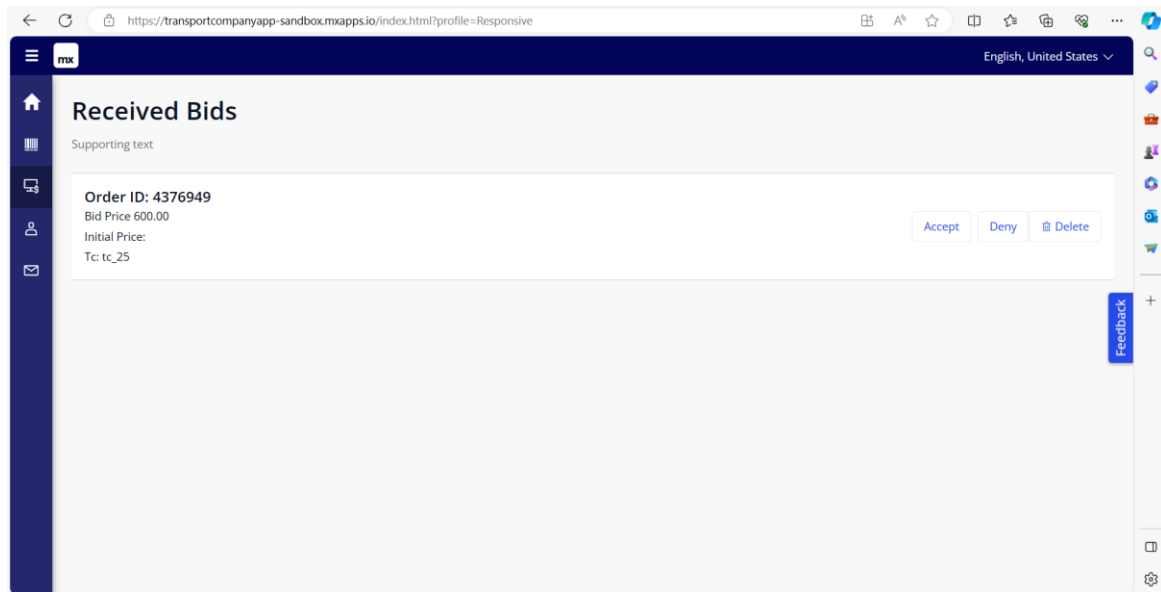


Figure 23 Received Bid Page

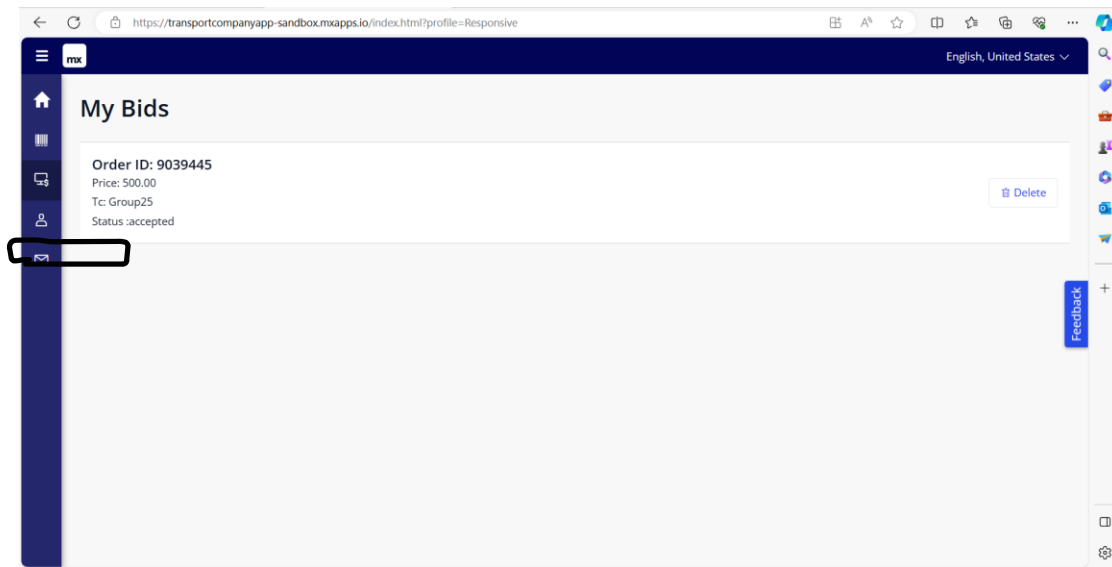


Figure 24 My Bids Page

7. Reflection and evaluation

This section aims to highlight the process of developing this application, to showcase the hurdles and challenges we faced in the development process, and how it changed our view on the process of implementing software that facilitate interorganizational systems. This project had a very short time span for such an endeavor, less than five weeks, which proved to increase the challenge even more.

The first two weeks of the project were used for the design phase. The main source of information for this project was the case description, which contained quite vague information that sometimes would even contradict itself. Empirically, this seems to be quite a common case in real life scenarios. Customers seem to have only a vague idea of what they want, and one shouldn't expect a proper software requirement list from them when started working on such a project. Instead, traditional software design techniques such as requirement elicitation, interviews, and lo-fi prototypes for validation should be used.

Furthermore, especially when multiple stakeholders are affected, no matter whether the development would be done by a single team, or multiple teams working on the different components, a traditional waterfall model might become costly in the long run. All the communication protocols for the different processes were all designed at the beginning of the project, which proved have serious flaws in the actual implementation. Because those were done in a rush by people with little to no experience in software development. We ended up changing the requirements and making assumptions in the absence of a real customer multiple times, which caused significant setbacks in the development process. Instead, a more flexible, cycle-based methodology such as Agile should be used. Clear user stories shall be derived from the case, which should be periodically discussed within the members of the team, and within the teams, so that the requirements are clear and aligned. Communication was not a strong point for us. We did schedule many meetings within our team, but to the other teams only when problems arose, which many of the times was too late to prevent major costly design changes. Our team did stick to the suggested protocols to facilitate interoperability, but it proved to be futile since towards the end of the project, changes have been made to the bidding protocol by other teams, to facilitate their interoperability, without us being

consulted. A similar situation happened with the control towers, due to the lack of proper communication and the misalignment of expectations, the whole SOAP API communication with the control towers became unfeasible to implement correctly, as the *WSDL* schema was received very late, and the CTs communicated with only some of the TC teams.

Additionally, the limitations of the used tools should be researched beforehand. None of the members in our team had prior knowledge in Mendix, so when designing the JSON format for receiving orders from the central firebase database, we did not have an informed opinion about it. The default JSON format from firebase utilized dynamic keys which were not supported in the Mendix import mappings, which proved to be way more difficult to overcome than expected.

Finally, this project gave us a good idea of how difficult achieving interoperability can be. In addition to adhering to a single customer's requirements, one must also constantly assess the other actors. If performed simultaneously, the likelihood of something going wrong increases drastically.

8. Conclusion

During this project we have studied the different aspects of supply chain management, specifically focusing on the operations of Uniprocterlevergamble Ltd. from the perspective of a transport company. By analyzing the business process modeling and execution, we have explored pricing, bidding, and decision-making within the context of transporting goods from distribution centers to supermarkets.

A key point during the whole project has been establishing effective communication between the business actors. To do so, designing standardized data transfer protocols among transport companies has been essential, showcasing their role in achieving interoperability, efficiency, and collaboration.

In addition, it was important to stand out from the different transportation companies and create competitive advantages. A crucial point in our implementation that we have used into our advantage, has been the way information is shared with regards to privacy. During the development, and the design process of the databases, different limitations have been identified: The first one and the most influential, has been that the transport companies create their own orders. This has led to information oversharing between transport companies. Because of all orders are shared in the same data base and these can be seen by all companies, the amount of information that is being shared is crucial. To differentiate from the different companies, we decided to not share the full content from an order unless this one is accepted on a bid. By being the only ones doing this, and having full access to the other orders, we are able to manipulate orders and to adapt ourselves to the market creating competitive prices. In reality, this would optimize and reduce the costs of our company.

To conclude, by working on this project we were able to learn how to use different methods and tools to study and comprehend the Uniprocterlevergamble Ltd.'s supply chain process. Moreover, we also realized the importance of data protocols and standardized processes in order to achieve smooth communication and an operational final result.

References:

- Barrett, S., & Konsynski, B. (1982). Inter-Organization Information Sharing Systems. *MIS Quarterly*, 6, 93-105. <https://doi.org/10.2307/248993>
- Kumar, K., & van Dissel, H. G. (1996). Sustainable Collaboration: Managing Conflict and Cooperation in Interorganizational Systems. *MIS Quarterly*, 20(3), 279-300. <https://doi.org/10.2307/249657>

Appendix A

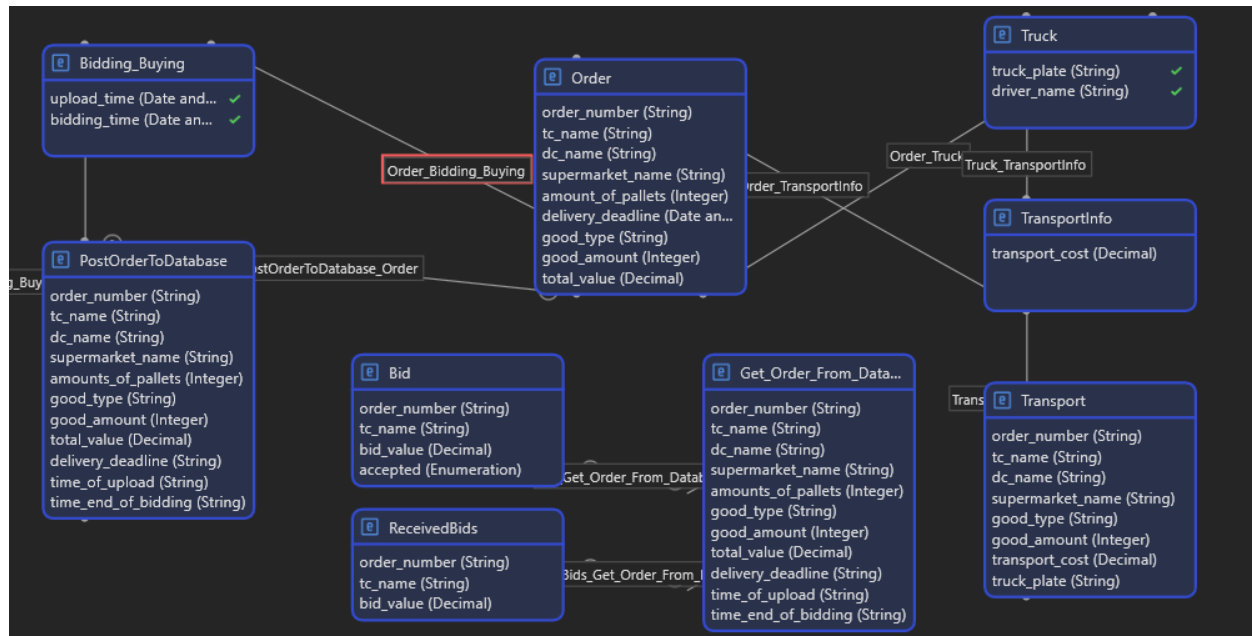


Figure 25 Main Application Domain Model

Appendix B

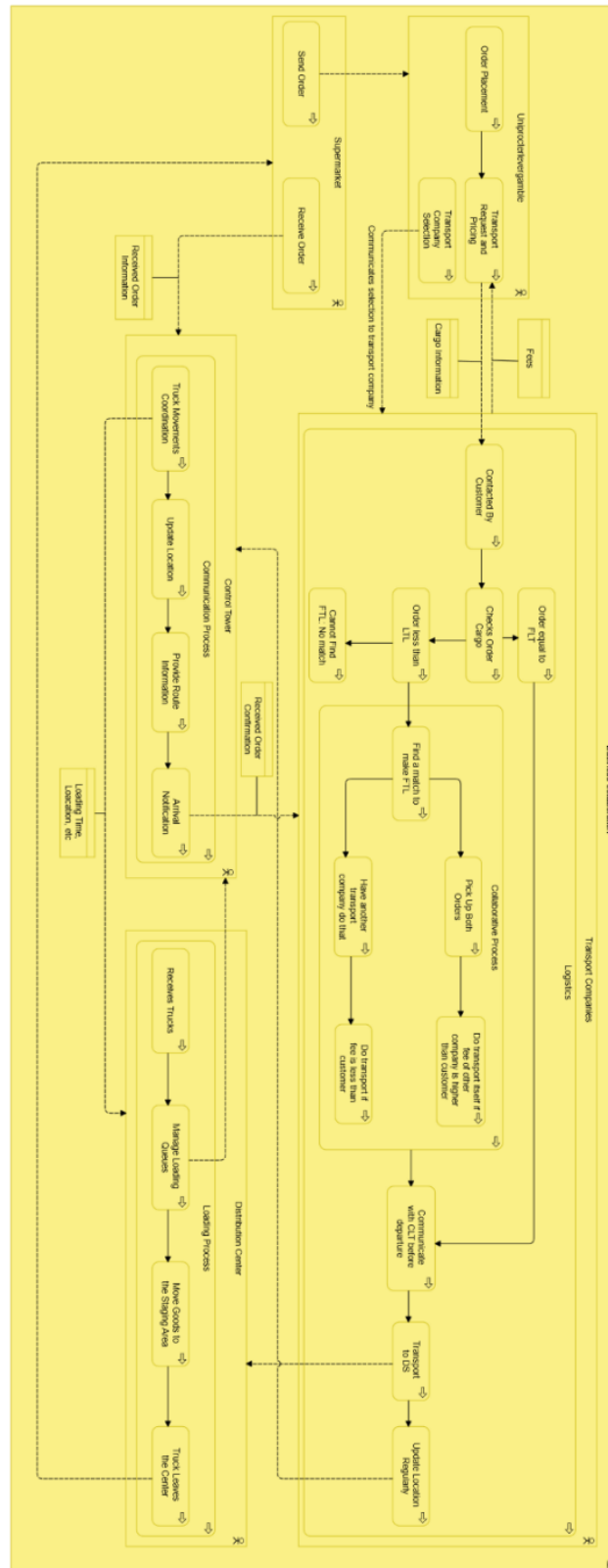


Figure 26 High-Level Business Process