

TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Andres Vassiljuk 220683IABB, Ilja Kissel'ov 213642IABB

**Implementation of shopping cart and product
functionality on the example of VKM Trading
OÜ**

Bachelor's thesis

Supervisor: Viljam Puusep
MSc

Tallinn 2024

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Andres Vassiljuk 220683IABB, Ilja Kisseljov 213642IABB

Ostukorvi ja toodete funktsionaalsuse realiseerimine VKM Trading OÜ näitel

Bakalaureusetöö

Juhendaja: Viljam Puusep
MSc

Tallinn 2024

Author's declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Andres Vassiljuk, Ilja Kisseljov

20.05.2024

Abstract

This thesis explores the development and implementation of a shopping cart and product functionality for VKM Trading OÜ, a B2B company. The shift to an online platform aims to enhance operational efficiency and customer service. The primary goal is to create a robust B2B e-commerce website to streamline purchasing for existing clients and attract new customers with a user-friendly interface. The platform also equips administrators with essential tools for database interaction and management, improving internal processes. The project employs modern web development technologies, focusing on product listing, shopping cart management, order processing, and user authentication. Despite significant progress, the project is ongoing. This thesis documents the development process, challenges encountered, and the impact on VKM Trading OÜ's operations, aiming to drive efficiency, enhance customer satisfaction, and support future growth.

This thesis is written in English and is 50 pages long, including 6 chapters, 39 figures and 2 tables.

Annotatsioon

[Thesis title in Estonian]

See bakalaureusetöö käsitleb ostukorvi ja toodete funktsionaalsuse arendamist ning juurutamist VKM Trading OÜ näitel. VKM Trading OÜ on B2B sektoris tegutsev ettevõtte, mis pakub laia valikut tooteid erinevatele ettevõtetele. Traditsiooniliste ärimudelite digitaliseerimine ja veebipõhiste platvormide kasutuselevõtt on muutunud ettevõtetele strateegiliselt oluliseks, et suurendada operatiivset efektiivsust ja parandada klienditeenindust.

Projekti peamine eesmärk on arendada välja tugev B2B e-kaubanduse veebisait, mis lihtsustab olemasolevate klientide ostuprotsessi ning meelitab uusi kliente kasutajasõbraliku ja tõhusa veebipõhise ostukogemuse kaudu. Lisaks sellele varustab platvorm administraatoreid oluliste tööriistadega, mis hõlbustavad andmebaasi haldust ja platvormi juhtimist, parandades seeläbi sisemisi protsesse ja üldist ärijuhtimist.

Projekti viivad läbi kaks tudengit, kes kasutavad kaasaegseid veebiarendustehnoloogiasid, et luua VKM Trading OÜ spetsiifilistele vajadustele kohandatud terviklik e-kaubanduse lahendus. Peamisteks funktsionaalsusteks on toodete nimekirja koostamine, ostukorvi haldamine, tellimuste töötlemine ja kasutajate autentimine. Arendusprotsess hõlmab üksikasjalikku planeerimist, disaini ja teostuse etappe, millest igaüks on suunatud konkreetsete äri vajaduste rahuldamisele ja kasutajakogemuse parandamisele.

Praegune projekti staatus hõlmab kogu protsessi põhjalikku dokumenteerimist, alates esialgsest nõuete kogumisest ja süsteemi disainist kuni juurutamise ja testimise etappideni. Kuigi on tehtud märkimisväärseid edusamme, on projekt endiselt pooleli ja vajab lõpuleviimist. See lõputöö annab üksikasjaliku ülevaate tekkivatest väljakutsetest, lahendustest ja üldisest mõjust VKM Trading OÜ äritegevusele.

Oodatav tulemus on transformatiivne B2B e-kaubanduse platvorm, mis suurendab efektiivsust, parandab kliendirahulolu ja positioneerib VKM Trading OÜ tulevaseks kasvuks konkurentsitihedas turul. Projekti jätkamise käigus keskendutakse ülejäänud

väljakutsete lahendamisele, kavandatud funktsioonide rakendamisele ja sujuva toimimise tagamisele realses keskkonnas. Lõputöö dokumenteerib saavutatud tulemused ja vajaliku edasise töö, pakkudes selget teekaarti projekti edukaks lõpuleviimiseks.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 50 leheküljel, 6 peatükki, 39 joonist, 2 tabelit.

List of abbreviations and terms

CSS	Cascading Style Sheets
HTML	Hypertext Markup Language
JWT	Json Web Token
SQL	Structured Query Language
ORM	Object–relational mapping
HTTP	Hypertext Transfer Protocol
DBMS	Database Management System
API	Application Programming Interface
UI	User Interface
UX	User Experience
VS Code	Visual Studio Code
SME	Small and medium-sized enterprises
SPA	Single-page application
CI/CD	Continuous integration and deployment
ACID	Atomicity, Consistency, Isolation, and Durability
B2B	Business to business

Table of contents

1 Introduction	13
1.1 The solvable problem	14
1.2 The structure of the work.....	16
2 Methodology.....	17
2.1 Description of the work process	17
2.1.1 Literature analysis	17
2.1.2 Coordination of selected technologies.....	21
2.1.3 Creating a graphic prototype	23
2.1.4 Gathering feedback on the prototype	25
2.2 Development.....	25
2.2.1 Frontend application development	25
2.2.2 Backend development	27
2.2.3 Connecting front and back application.....	29
2.2.4 Testing	30
2.2.5 Release plan.....	30
2.3 Description of tools	32
2.3.1 Why does this combination work better than others?	33
3 Why B2B	37
3.1 Elements	37
3.1.1 User Registration and Authentication:	37
3.1.2 Product Management:.....	37
3.1.3 Shopping Cart and Order Processing:	37
3.1.4 Reporting and Analytics:.....	37
3.1.5 Security and Compliance:.....	38
4 Work results.....	39
4.1 Functional requirements	39
4.2 Non-functional requirements	40
4.3 Application design.....	41
5 Analysis and conclusions	43

5.1 Comparison with Existing Software.....	43
5.1.1 User Interface Advantages Over Existing Software.....	43
5.1.2 User Interface Shortcomings Compared to Existing Software.....	43
5.1.3 Functionality Advantages Over Existing Software	43
5.1.4 Functionality Deficiencies Compared to Existing Software	43
5.2 Reflection on the work done.....	43
5.2.1 Things That Went Well	44
5.2.2 Things That Went Wrong.....	44
5.2.3 Usefulness of the Work Done.....	45
5.2.4 Things to Do Differently	46
5.3 Further Development.....	46
6 Summary.....	47
References	49
Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis	51
Appendix 2 Desktop version Figma model	52
Appendix 3 Phone version Figma model	62
Appendix 4 Endpoints table	63
Appendix 5 Table relations	67
Appendix 6 Postman.....	68
Appendix 7 Backend	70
Appendix 8 Test Frontend	72
Appendix 9 Frontend external design.....	75

List of figures

Figure 1 Admin page prototype in Figma	52
Figure 2 Home Page prototype in Figma	53
Figure 3 Registration, Project start & Order checkout prototype in Figma	54
Figure 4 Products page prototype in Figma	55
Figure 5 Unit product page prototype in Figma	56
Figure 6 News page prototype in Figma	57
Figure 7 Other news page prototype in Figma	58
Figure 8 Contacts page prototype in Figma	59
Figure 9 About us & Certificates pages prototype in Figma.....	60
Figure 10 Error 404(Not found) & Log-in pages prototype in Figma.....	61
Figure 11 Shopping cart page prototype in Figma	61
Figure 12 Phone version prototype in Figma	62
Figure 13 Table relations.....	67
Figure 14 Postman collection structure and a test sequence example.....	68
Figure 15 Example of a Postman test (POST api/company).....	69
Figure 16 Backend file structure	70
Figure 17 Example of a request.....	70
Figure 18 Example of a router.....	71
Figure 19 Example of a middleware.....	71
Figure 20 Example of a table in admin panel in test frontend.....	72
Figure 21 Items fetched from the backend displayed in the test frontend.....	72
Figure 22 Item detailed view and item count calculator in test frontend	73
Figure 23 Example of an edit form in admin panel in test frontend.....	73
Figure 24 Basket in test frontend.....	74
Figure 25 Home page in frontend external design	75
Figure 26 Product page in frontend external design.....	75
Figure 27 News page in frontend external design	76
Figure 28 About us page in frontend external design.....	76
Figure 29 Certificates page in frontend external design.....	77

Figure 30 Contact page in frontend external design.....	77
Figure 31 General finder in frontend external design.....	78
Figure 32 Login in frontend external design	78
Figure 33 Login page in frontend external design.....	79
Figure 34 Registration in frontend external design	79
Figure 35 Error 404 (Not found) page in frontend external design.....	80
Figure 36 Admin panel Products page in frontend external design	80
Figure 37 Admin panel New User page in frontend external design	81
Figure 38 Unit product (item) page in frontend external design	81
Figure 39 Footer in frontend external design	82

List of tables

Table 1 Description of pages	24
Table 2 List of endpoints	63

1 Introduction

In the digital age, the transformation of traditional business models into online platforms is crucial for companies aiming to remain competitive and expand their market reach. E-commerce solutions are at the heart of this transformation, facilitating seamless transactions and interactions between businesses and their customers. This thesis examines the development and implementation of shopping cart and product functionality for VKM Trading OÜ, a company operating in the B2B sector.

VKM Trading OÜ specialises in distributing a wide range of products to various businesses. To enhance operational efficiency and customer service, transitioning to an online platform becomes a strategic necessity. The goal is to create a robust B2B e-commerce website that simplifies the purchasing process for existing clients while attracting potential new customers through a user-friendly and efficient online shopping experience. Additionally, the platform will equip administrators with essential tools for database interaction and platform management, streamlining internal processes and improving overall business operations.

This project is undertaken by two students, leveraging contemporary web development technologies to design and implement a comprehensive e-commerce solution tailored to the specific needs of VKM Trading OÜ. Key functionalities include product listing, shopping cart management, order processing, and user authentication. The collaborative effort encompasses detailed planning, design, and execution phases, each aimed at addressing specific business needs and enhancing user experience.

The current status of the project involves thorough documentation of the process from initial requirements gathering and system design to implementation and testing phases. This thesis provides a detailed account of the challenges encountered, solutions devised, and the overall impact on VKM Trading OÜ's business operations. While significant progress has been made, it is important to note that the project is still ongoing and not yet

complete. This work includes an analysis of what has been achieved so far and outlines the remaining tasks necessary for full implementation.

By examining the steps taken to develop this e-commerce platform, this thesis contributes to the academic understanding of e-commerce implementations in the B2B sector and offers practical insights for businesses seeking to adopt similar solutions. The anticipated outcome is that the B2B e-commerce platform will transform VKM Trading OÜ's operations, driving efficiency, enhancing customer satisfaction, and positioning the company for future growth in a competitive marketplace.

1.1 The solvable problem

In the competitive B2B landscape, VKM Trading OÜ faces several inefficiencies due to its traditional, manual processes for order management, product listings, and customer interactions [19]. These challenges hinder the company's ability to operate efficiently, scale effectively, and provide timely customer service [21][23][34].

Identified Challenges:

1. **Manual Order Processing:**
Labor-intensive and error-prone order management leads to delays and inaccuracies.
2. **Inefficient Product Management:**
Updating product information and inventory levels without a digital catalogue is cumbersome.
3. **Limited Customer Interaction:**
Clients face inconvenience with current methods of communication, which are slow and restricted to business hours.
4. **Scalability Issues:**
The existing processes are not scalable, limiting VKM Trading OÜ's growth potential.

Proposed Solution:

To address these challenges, the project proposes developing a B2B e-commerce platform with the following key features:

1. **Online Shopping Cart:**
Simplifies the order placement process for clients.
2. **Product Management System:**
Enables real-time updates of product information and inventory levels.
3. **User Accounts and Authentication:**
Provides secure access for clients to manage orders and view history.
4. **Order Processing Automation:**
Reduces manual intervention and errors, speeding up fulfilment.
5. **Admin Panel:**
A user-friendly administrative panel allows company representatives to manage products and news without programmer assistance. They can easily add, delete, or update product information and maintain a news feed.
6. **Registration Control:**
New company registrations must be approved by an admin, ensuring controlled and secure access to the platform.

By implementing this e-commerce platform, VKM Trading OÜ will improve operational efficiency, enhance customer service, and position itself for scalable growth in the B2B market. This solution aims to streamline processes, reduce errors, and provide a better user experience, ultimately driving business success.

1.2 The structure of the work

The work is divided into several main stages, starting with the introduction to the project and its objectives [19]. This is followed by a detailed description of the methodologies used for system analysis, design, and development [25]. The next stage involves a comprehensive review of B2B e-commerce literature and technologies [19][32]. The development of the platform is documented, highlighting the implementation of key features such as the shopping cart, product management system, user authentication, admin panel, and customer support integration [23][27][29]. The final stage involves a thorough analysis and comparison with existing systems [21][22]. In the analysis and conclusions section, the project's execution process, and recommendations for future development are evaluated [24][26]. The work concludes with a summary of the main findings and outcomes [28][29].

2 Methodology

The development process for this thesis employed an inductive methodology. This approach is ideal for creating new software, allowing us to build a functional e-commerce platform tailored specifically for VKM Trading OÜ [25]. Inductive reasoning facilitated the identification of general patterns and principles from specific examples, critical in creating a product from scratch. This method also ensured a deep understanding of the target audience's needs and preferences, resulting in a more effective solution [26]. Additionally, the inductive approach allowed for the analysis and comparison of existing solutions to identify their strengths and weaknesses, vital for developing a product that meets the end-users' needs [21][24].

The development was iterative, where we organised a meeting every week and shared progress. However, as development progressed, difficulties began to arise more and more often and meetings were postponed or completely cancelled. Due to the general lack of time, the work schedule was free - we worked when there was time, which also contributed to the general desynchronization [25].

We made all important decisions together. We organised a small meeting and exchanged arguments in favour of one proposal or another. To solve less significant problems, we had freedom of action within our areas of responsibility. If necessary, meetings were organised with the VKM Trading OÜ representative to clarify various details of the project [25].

2.1 Description of the work process

The development process began with a thorough analysis of the existing systems to identify their shortcomings and determine the requirements for the new platform. The identified requirements were divided into functional and non-functional categories to ensure a comprehensive understanding of the system's needs [21][22].

We began the project by receiving input from our supervisors, including the system metadata database, which already contained classifier values and test tasks. We also received a description of the task verification logic. Our task was to implement a

functional, user-friendly interface intended for everyday business use, not just a prototype but a working software solution [25]. We started by familiarising ourselves with similar systems. Responsibilities were distributed: one person at the frontend, one person at the backend. The backend person also worked on a simple minimalistic test frontend prototype for frontend-backend connection testing [25].

The initial phase involved creating a prototype of the user interface using Figma. Based on the feedback received, we made several enhancements to the user interface before proceeding to the system implementation [26]. Additionally, we engaged with the client (VKM Trading OÜ) to ensure the developed solution met their specific needs and expectations [26].

On the server side, we utilised Node.js with PostgreSQL as the database. PostgreSQL was chosen for its robustness, scalability, and support for complex queries and transactions [27]. We used Sequelize as the ORM tool to interact with PostgreSQL, which simplifies database operations and ensures efficient querying and data manipulation. Additionally, the *pg-hstore* library was employed to serialize and deserialize the *hstore* data type in PostgreSQL, allowing us to handle key-value pairs efficiently.

For the development of the user interface, we chose React and TypeScript due to their flexibility and robustness. React allowed us to build a dynamic and responsive interface, while TypeScript provided strong typing and error checking, which enhanced the code quality and maintainability [29]. SCSS was used for styling, allowing us to write modular and reusable CSS with improved readability and maintainability [29]. Additionally, HTML was used to structure the web pages [30].

The development was carried out using Git for version control, which facilitated collaboration and code management [31]. Given the tight deadlines, we had to quickly learn new technologies while simultaneously working on the development. This included familiarising ourselves with React, TypeScript, SCSS, HTML, and the implementation of JWT for secure sensitive data transmission [32]. We also used the PostgreSQL client for Node.js (*pg*) and the *pg-hstore* library for handling key-value pairs [28].

Throughout the project, we faced challenges in communication due to different schedules. Balancing rapid learning of new technologies with the development process, all within a compressed timeline, was demanding [25]. However, we have successfully developed the

most crucial structures and functionalities supporting the further development of the B2B e-commerce platform adapted to the needs of VKM Trading OÜ [19].

It is important to note that the project is not yet completed, and its finalisation will be carried out at a later stage. We plan to continue refining and enhancing the platform based on further feedback and testing to ensure it fully meets the requirements and expectations of VKM Trading OÜ [24].

2.1.1 Literature analysis

The development and implementation of our e-commerce platform are guided by several key insights drawn from relevant academic literature and industry best practices.

The adoption of e-commerce applications by small and medium-sized enterprises (SMEs) is influenced by factors such as perceived relative advantage, compatibility with existing systems, and cost considerations. SMEs are more likely to adopt e-commerce solutions that offer clear benefits, integrate seamlessly with their current practices, and are economically viable [19]. This underscores the importance of demonstrating the tangible benefits of our platform and ensuring it aligns well with VKM Trading OÜ's existing workflows.

Secure client authentication and session management are critical components of our platform. The use of JSON Web Tokens (JWT) for secure authentication ensures that our user sessions are managed securely and efficiently [20].

Performance optimization is another crucial aspect. The impact of Object-Relational Mapping (ORM) frameworks on query performance emphasizes the need for careful ORM usage and the potential incorporation of custom SQL queries for performance-critical operations. This is essential for maintaining efficient database interactions and ensuring the platform's responsiveness [21].

The choice of a single-page application (SPA) framework significantly affects the user experience. It has been demonstrated that React generally outperforms other frameworks like AngularJS and Angular 2 in terms of performance and maintainability [1][22].

Node.js offers scalability and efficiency for handling heavy network loads, which is critical for our backend performance. However, the learning curve associated with its asynchronous nature requires focused training for our developers to ensure effective implementation [23].

Differential testing of ORM systems can significantly enhance the robustness and correctness of our ORM system. By comparing query results across different implementations, we can detect and address bugs, ensuring reliable database interactions [24].

Implementing a continuous integration and deployment (CI/CD) pipeline will be crucial for maintaining high availability and scalability of our platform. While we have initialized Docker, future implementation of CI/CD frameworks will streamline our deployment processes and ensure consistent updates [26].

Effective state management is essential for a smooth user experience in our React applications. The use of observables and actions for efficient and reactive state management in *mobx* will simplify state management and improve performance, making our codebase more maintainable [1][27].

User interface design principles are fundamental to creating an intuitive and user-friendly platform. Clear entry and exit points, consistent logic, and immediate feedback for user actions will enhance the usability and overall user experience of our e-commerce platform [28].

Writing clean, maintainable code is crucial for the long-term success of our platform. Using descriptive names and ensuring each function has a single responsibility will make our codebase easier to understand and maintain, facilitating better collaboration and more efficient debugging [29].

The architectural principles of DBMS [31] support our choice of PostgreSQL for our e-commerce platform. PostgreSQL's efficient resource management, robust query optimization, and adherence to ACID properties align with these principles, ensuring

reliable and efficient database operations. This alignment guarantees that our platform can handle high transactional demands while maintaining data integrity and performance.

The article on JSON Web Tokens (JWT) provides a standard for securely transmitting claims between parties. JWTs are compact, URL-safe, and can be signed or encrypted to ensure integrity and confidentiality. Implementing JWTs in our project will enhance the security and efficiency of client authentication and session management [32].

The article highlights TypeScript's role as a statically typed superset of JavaScript, providing a smooth transition for JavaScript developers without major rewrites. It emphasizes TypeScript's advanced type system, which helps catch errors at compile time and enhances code reliability. Implementing TypeScript in our project will improve code quality and maintenance while offering enhanced tooling and IDE support for a better development experience [33].

The reference provides standardized data models for core business functions like product management, ordering, and invoicing. These templates ensure robust and scalable database designs, streamlining development and reducing errors in our data architecture. Utilizing these models will enhance the efficiency and reliability of our e-commerce platform's database system [35].

2.1.2 Coordination of selected technologies

PostgreSQL - DBMS for data storage.

The backend uses PostgreSQL for managing and storing application data [5].

Express.js - Web framework for Node.js.

Handles HTTP requests, routing, middleware, and API endpoints in the backend [4].

React - Library for building user interfaces.

The frontend uses React to create the user interface [1].

Node.js - JavaScript runtime environment for server-side execution.

Runs the backend server and handles server-side logic [3].

TypeScript - Statically typed programming language that extends JavaScript.

Used as the primary language for both client and server to enhance reliability and ease of development [2].

Tools and Development Environment:

VSCode - Source code editor.

Used for writing, editing, and debugging code. It supports many extensions that make it easier to work with various programming languages and technologies.

WinRAR - File archiving program.

Used for compressing and decompressing files and archives.

Pgadmin - Graphical interface for managing PostgreSQL.

Used for administering PostgreSQL databases and executing queries.

Git - Version control system.

Used for tracking changes in code and collaborating on projects.

Drawio - Tool for creating diagrams.

Used for creating diagrams like class diagrams and flowcharts.

Figma - Tool for interface design and prototyping.

Used for developing user interface mockups and interactive prototypes.

Postman - Tool for testing APIs.

Used for developing, testing, and debugging APIs.

Windows 11 - Operating system.

Used as the primary operating system for development. It provides the necessary environment to work with the various tools and technologies used in the project.

2.1.3 Creating a graphic prototype

Following the selection of technologies, detailed discussions on the web application's functionality were conducted, and the minimum number of necessary pages was determined. Figma was chosen as the application for creating the prototype because of its extensive functionality for creating detailed prototypes of web applications and its

convenient feature for demonstrating the prototype to third parties. As a result, the following page concepts were developed:

Table 1 Description of pages

<p>Admin - Products - NewUsers - Users - NewCompanies - Companies</p>	<p>The Admin Interface serves as the central hub for managing various aspects of the VKM Trading OÜ platform. Administrators can handle product listings, user registrations, and company profiles. The Products page allows adding, updating, and removing products to keep the catalogue current. The New Users section displays pending registrations for approval, and the Users page manages all user accounts, including password resets and updates. Similarly, the New Companies page lists company registrations awaiting approval, while the Companies page maintains and updates company profiles, ensuring streamlined and effective platform management.</p>
<p>Home</p>	<p>The main landing page provides general information about the platform, login functionality, and language selection options.</p>
<p>Products</p>	<p>Displays a comprehensive list of products available on the platform, categorised for easy browsing. Includes detailed product descriptions, pricing, and availability.</p>
<p>News</p>	<p>A section for company news and updates. Admins can post news articles, announcements, and other relevant updates.</p>
<p>Contacts</p>	<p>Provides contact information for VKM Trading OÜ, including phone numbers, email addresses, and physical addresses.</p>
<p>About</p>	<p>Contains detailed information about VKM Trading OÜ, its mission, values, and history.</p>
<p>Certificates</p>	<p>Lists various certifications and accreditations that VKM Trading OÜ has earned, demonstrating the company's credibility and standards.</p>
<p>Not Found</p>	<p>A default page shown when users navigate to a nonexistent or broken link on the platform.</p>
<p>Login</p>	<p>The page where users enter their credentials to access the platform. It includes fields for username and password, and links for password recovery and new user registration.</p>

2.1.4 Gathering feedback on the prototype

Gathering feedback on the prototype was a crucial step in the development process. We presented the prototype to various stakeholders and gathered their input through informal, verbal feedback sessions [26]. This method allowed us to quickly capture reactions and suggestions, which we then incorporated into subsequent iterations of the design [26]. Additionally, feedback from the client, VKM Trading OÜ, was collected to ensure the platform met their specific requirements and expectations [26]. This feedback provided valuable insights into user experience and functionality, guiding us in refining the platform to better meet the needs and expectations of VKM Trading OÜ and its clients [26].

2.2 Development

2.2.1 Frontend application development

The frontend development for the B2B e-commerce platform of VKM Trading OÜ was meticulously planned and executed using modern technologies, including React, TypeScript, SCSS, and HTML [1][2].

Technology Choices:

React was chosen for its component-based architecture, enabling efficient handling of dynamic content and creation of reusable UI components, ensuring consistency and reduced development time. Despite being new to the team, TypeScript was selected for its static typing capabilities, which helped catch errors early in the development process, reducing runtime errors and improving code maintainability [1][2].

Development Workflow:

The development process began with creating wireframes and high-fidelity prototypes using Figma. These prototypes were instrumental in gathering feedback from stakeholders and refining the design. Once the design was approved, it was translated into functional React components [1].

Wireframing and Prototyping:

Initial wireframes were created to outline the UI structure, which were then transformed into detailed Figma prototypes. Feedback was collected from various stakeholders, including potential users and the client, to refine the design. This iterative process ensured that the design was both user-friendly and met business requirements.

Component Development:

Prototypes were converted into React components. This involved creating individual UI elements like buttons, forms, and navigation menus as reusable components [1]. These components adhered to the design specifications and were styled using SCSS for modular, maintainable, and reusable CSS. HTML was used to structure the web pages, ensuring semantic and accessible markup.

State Management:

To manage the application state, *mobx* was integrated. *Mobx* allowed for centralised state management, ensuring data consistency across different parts of the application. This was crucial for handling complex state transitions and maintaining a predictable state throughout the app. At the current stage it is only implemented in the test frontend.

API Integration:

Axios was used for API integration, facilitating communication between the frontend and backend services. This enabled dynamic data fetching and real-time updates, ensuring that the UI remained responsive and up-to-date with the latest information from the server. The integration process involved setting up *axios* instances, configuring request and response interceptors, and handling various HTTP methods for CRUD operations. At the current stage it is only implemented in the test frontend.

Styling:

SCSS was chosen for styling the components, utilising its features like variables, nesting, and *mixins* to create a clean and maintainable stylesheet structure. This approach allowed for easy modifications and ensured a consistent look and feel across the application. The use of SCSS also enabled better organisation of styles, making it easier to apply global themes and component-specific styles.

Challenges and Solutions:

- Learning TypeScript: Although TypeScript was mostly new to the team, its adoption significantly improved code quality. The initial learning curve was managed through dedicated study and practice, which paid off in the form of fewer runtime errors and more maintainable code.
- Translating Figma Designs: Ensuring that the final product matched the Figma designs required close attention to detail. This process involved iterative testing and adjustments to align the React components with the visual and functional specifications of the prototypes [1].
- State and Props Management: Managing state and props was a difficult task to accomplish without any experience. Especially promise handling in *mobx* took a lot of time to understand.

Project Status:

Despite significant progress, the project is not yet complete and has not been deployed for use. Future development plans include implementing additional features, enhancing security measures, and optimising performance. The foundation laid during this development phase has equipped the team with valuable experience and skills, positioning them well for completing and refining the platform to meet the evolving needs of VKM Trading OÜ and its clients.

2.2.2 Backend development

Data Collection, Analysis, and Technology Stack Selection

Before starting the project, it was essential to study available technologies, compare them, and form a stack suitable for the project's tasks. Information sources included official documentation, thematic forums, and discussions with professionals in web application development. Key factors in selecting the technology were the availability and volume of learning materials, ease of use, and simplicity.

Basic Backend Development

Once the stack was determined, the initial step involved initialising the project with Node.js and installing fundamental libraries such as Express, *pg*, *pg-hstore*, Sequelize,

cors, *dotenv*, and *nodemon*. Configuration and utility files with environment variables and helper functions were then created and adjusted. Subsequently, the foundation of the application was established by creating a PostgreSQL database and setting up necessary connections.

Database Design

With the basic backend elements in place, the next task was to design the database based on information and business logic provided by the client. During development, new requirements and clarifications were continuously incorporated, resulting in corresponding changes to the database.

Designing a flexible and understandable data architecture was a key requirement. Initial tables were created to store data about users, products, product characteristics, and auxiliary tables for product and characteristic typology, as well as for managing relationships between primary tables. Following discussions and revisions, a relational diagram was drawn up based on the chosen architecture.

All relationships and fields were defined in the application using Sequelize. Models were then used to create corresponding tables in the database, with Pgadmin utilised for database overview.

Backend Framework - Requests, Models, Routing, Error Handling, Middleware, Roles

With the database and its connection established, the development of the application's framework began. Initially, a routing file was created using Express to link all routes from different parts of the application. Separate route files were then developed for each section of the application, describing all primary routes for accessing tables and user operations.

To separate request handling logic, controllers were created for each table. Initial requests were defined to retrieve all data, fetch single objects, and add new objects. The UUID library was installed for generating string identifiers. As products included images, the *express-fileupload* library was installed to handle files from requests, and the path module

was used for moving files. Files received from requests were transferred to a designated folder.

For user operations, registration and authentication mechanisms were implemented using JWT. User data in the database is stored in encrypted form, with the *bcrypt* library installed for hashing.

Postman was employed to facilitate request testing, with environment variables and collections set up for convenience and efficiency. All requests were tested and verified in Postman.

Middleware for user role verification and error handling was also developed, with a separate file for defining different error types.

Docker Setup

Understanding Docker principles, installation, and configuration were key steps when the backend foundation was ready and tested. Docker files necessary for application operation were created, including Docker images, containers, and required connections. PostgreSQL, Pgadmin, and the current backend version were installed in Docker. The correctness of Docker's operation was verified. At this stage, Docker preparation for backend deployment on a public domain was carried out, though it was not used in development to avoid slowing down the launch and debugging processes.

Refinement and Error Correction

The existing code was reviewed, with stability enhanced by adding validation in requests and additional error handling. Postman tests were refined, with missing tests added and redundant ones removed.

2.2.3 Connecting front and back application

The integration of the frontend and backend applications was essential for the B2B e-commerce platform for VKM Trading OÜ. We designed a RESTful API using Node.js and Express to handle client requests, with endpoints for user authentication, product

management, and order processing. JSON Web Tokens (JWT) ensured secure data transmission. On the frontend, API calls were made using *axios*, connecting React components to backend services for dynamic data fetching and state management. *Mobx* was used for state management. This seamless integration resulted in a secure, efficient, and maintainable platform. Although as 2 versions of the frontend were simultaneously in development, only the test frontend is linked to the frontend at the current stage.

2.2.4 Testing

To ensure the robustness and reliability of the backend, we utilised Postman due to our prior experience with it. Postman tests are written in JavaScript, which is convenient given our project's reliance on TypeScript and JavaScript [31][32].

We began by studying the structure and functionality of tests in Postman, familiarising ourselves with the syntax and basic principles [31].

After establishing the primary requests and integrating them into Postman, we developed tests to cover various scenarios, such as unauthorised requests, requests without appropriate permissions, invalid requests, and duplicate data entries [32]. These tests also evaluated response times, status codes, response structures, and other critical aspects [32]. All tests were organised into collections for ease of navigation, with a master collection allowing for the sequential execution of all tests, significantly improving efficiency and usability [31].

The tests were structured as usage scenarios, ensuring comprehensive coverage. For example, a scenario involving the *feature* and *featureType* tables included steps for adding, modifying, and subsequently deleting test data, avoiding persistent changes in the database [32]. Environmental variables were employed throughout the tests, supporting the addition of new tests and enhancing current ones [31].

Frontend testing has not been performed at this stage. Further development and testing of the frontend will be necessary to ensure full integration and functionality [29].

2.2.5 Release plan

This section outlines the release plan for the project, based on the current development status and future goals.

Finalising Frontend Development

Properly linking the production frontend to the backend.

Enhancing interfaces for all key functions (product management, cart, order placement, user registration, and authentication).

Integrating additional libraries and tools to improve user experience

Integrating Additional Features

Implementing a news and newsletter system.

Integrating external APIs to expand functionality (e.g., chat, notifications).

Testing and Debugging

Conducting a full cycle of functional and user testing.

Fixing identified bugs and optimising performance.

Preparation for Deployment

Configuring the environment for deploying the application on a public hosting platform.

Ensuring data security and compliance with GDPR requirements.

Deployment on Public Hosting

Deploying the application using Docker on the chosen platform.

Setting up automated CI/CD to simplify subsequent releases and updates.

Post-Release Maintenance

Monitoring and supporting the application's functionality.

Collecting and analysing user feedback to further improve the system.

Future Development Plans

After the initial release, further development is planned to transform the project into a full-fledged online store with expanded features:

Payment System Integration

Connecting payment gateways for online payments.

User Management Enhancements

Including capabilities for administrators to manage roles and permissions.

Integration with External Systems

Connecting to ERP systems and other business management tools.

Analytics Expansion

Implementing tools for sales analysis and user activity monitoring.

Cybersecurity Enhancements

Continuously updating and improving security measures in response to new threats and requirements.

2.3 Description of tools

Node.js - a runtime environment for JavaScript that allows the execution of JS code on the server side [3].

Used for creating server-side applications, handling HTTP requests, interacting with databases, and other server tasks.

Node.js is well-suited for building scalable network applications, particularly when working with web sockets and handling multiple connections [27].

Express - a minimalist and flexible framework for Node.js applications that provides powerful features for web and mobile applications.

Used to create server-side web applications, manage routing, handle requests and responses.

Express simplifies the development of server-side code through middleware and routing, making the code cleaner and easier to understand [28].

TypeScript - a superset of JavaScript that adds strong typing and object-oriented concepts.

Used for writing code on both the client and server side, providing greater code safety and easier scalability.

Typing helps in large and complex projects to detect errors early in the development process, simplifies refactoring, and enhances code quality [2][29].

PostgreSQL - powerful, open-source relational database management system that supports SQL.

Used to store, process, and securely access data in the application.

PostgreSQL supports advanced features such as complex queries, transactions, and JSON support, making it ideal for enterprise-level applications [28].

Docker a platform for developing, shipping, and running applications in containers.

Used to package the application along with its dependencies into standardised units for development or deployment.

Docker ensures ease and convenience of deployment, as well as consistency between development and production environments [33].

2.3.1 Why does this combination work better than others?

The combination of these technologies offers a powerful full stack for developing modern web applications. Using JavaScript and TypeScript throughout the stack makes it easier for developers to move between the client and server sides. Docker makes it easy to deploy and scale applications, and PostgreSQL provides reliable data storage. This makes the stack powerful, flexible, and suitable for developing high-performance web applications [2].

Libraries:

Express - a minimalist and flexible web framework for Node.js that provides powerful features for creating web and mobile applications.

Used to create server-side applications, manage routing, handle HTTP requests and responses, and implement middleware.

Advantages:

- Simple to use and configure.
- Middleware support for adding various functionalities.
- Extensive ecosystem of plugins and extensions.
- High performance due to its minimalist architecture.

CORS - (Cross-Origin Resource Sharing) is middleware for Express that allows web applications to perform cross-domain requests.

Used to configure and allow cross-domain requests, ensuring secure interactions between the client and server located on different domains.

Advantages:

- Easy integration with Express applications.
- Flexible settings for managing permissions.
- Enhanced security through access control.

Dotenv - a module for loading environment variables from a *.env* file.

Used to manage application configuration by loading environment variables from a *.env* file, avoiding the need to store sensitive information in code.

Advantages:

- Simplifies configuration management.
- Enhances security by keeping sensitive information out of the codebase.
- Easy to change configuration without altering the code.

JSON Web Token (JWT) - a standard for creating access tokens used to transfer data between parties in a compact and secure format.

Used for user authentication, secure data transmission between client and server, and session management.

Advantages:

- High level of security.
- Easy to use and integrate.
- Lightweight and scalable.
- Independent of server session.

pg and pg-hstore

Pg is a PostgreSQL client for Node.js, and *pg-hstore* is a serializer for PostgreSQL's *hstore* data type.

Pg is used for interacting with the PostgreSQL database, executing SQL queries, and retrieving data. *Pg-hstore* is used for processing *hstore* data.

Advantages:

- Reliability and performance.
- Support for all PostgreSQL features.

- Easy to use and integrate with other libraries.

Sequelize - a powerful and popular ORM for Node.js that supports multiple SQL dialects, including PostgreSQL.

Sequelize simplifies working with databases by providing a high-level API for interacting with them. It allows developers to use an object-oriented approach for data manipulation, automatically managing SQL queries, table associations, and data model synchronisation.

Advantages:

- Accelerates development.
- Reduces the amount of code.
- Enhances code readability and maintainability.
- Supports associations, validations, and migrations.

Express-fileupload - middleware for file uploads in Express applications.

Used to handle file uploads on the server, saving and processing uploaded files.

Advantages:

- Simple integration and use.
- Supports various file types.
- Configurable upload parameters.

Bcrypt - a library for hashing passwords, ensuring their secure storage.

Used to hash passwords before storing them in the database and to verify passwords during user authentication.

Advantages:

- High level of security.
- Protection against brute force attacks.
- Easy to integrate and use.

Fs-extra - an extension of the standard fs module for working with the file system, adding additional methods.

Used for improved interaction with the file system, providing additional methods for working with files and directories.

Advantages:

- Extended capabilities compared to the standard fs module.
- Support for asynchronous and synchronous methods.

- Easy to use and integrate.

UUID - a library for generating unique identifiers (UUIDs).

Used to create unique keys for database records, session identifiers, and other objects requiring uniqueness.

Advantages:

- Guaranteed uniqueness of identifiers.
- Easy to use and integrate.
- Support for various versions of UUID.

Nodemon - utility that automatically restarts the server when project files are changed.

Used to speed up development by automatically restarting the application when changes are made to the code.

Advantages:

- Speeds up the development process.
- Reduces manual operations.
- Easy to integrate and use.

Mobx - a library for state management in React applications, based on reactive programming [1].

Used to simplify state management and automatically update the interface when the state changes.

Advantages:

- Reactive interface updates [1].
- Simple and flexible use.
- Support for complex data structures and dependencies.

3 Why B2B

The B2B functionality on the VKM Trading OÜ platform is designed to meet the specific needs of business clients, ensuring a seamless and efficient user experience. This section describes the main features and functionalities to be implemented to support B2B operations [19][34].

3.1 Elements

3.1.1 User Registration and Authentication:

The platform includes a robust user registration and authentication system tailored for B2B clients. Only businesses can register, ensuring that all users are legitimate business entities [32]. Registrations must be approved by an administrator to gain access to the platform [25]. Once registered, users can log in using secure authentication methods, including the use of JSON Web Tokens (JWT) for secure data transmission [32].

3.1.2 Product Management:

The product management functionality allows VKM Trading OÜ to efficiently manage their product catalogue. Administrators can add, update and remove products through an intuitive admin panel. The system supports detailed product descriptions, pricing, and categorization, ensuring that clients have access to accurate and up-to-date product information [30].

3.1.3 Shopping Cart and Order Processing:

The shopping cart feature enables business clients to select and manage products they wish to purchase. The system supports bulk ordering, multiple product variations. Once the selection is complete, clients can place orders [29].

3.1.4 Reporting and Analytics:

The platform plans to provide detailed reporting and analytics tools for both administrators and clients. Administrators will be able to generate reports on sales,

inventory, and customer activity, enabling them to make informed business decisions [27]. Clients will have access to their order history, track order statuses, and view personalised product recommendations based on their purchasing behaviour [30]. These features are currently in the planning stage and will be implemented as the project progresses, enhancing the platform's functionality and user experience [24].

3.1.5 Security and Compliance:

Ensuring the security and compliance of the platform is a top priority. Planned security measures include advanced encryption, secure data storage, and regular security audits [32]. We aim to comply with industry standards and regulations, such as GDPR, to protect client data and ensure the platform operates within legal requirements [32]. These measures are currently in the planning stage and will be implemented as the project progresses, ensuring that the platform remains secure and compliant as it continues to develop [24].

By incorporating these features, the B2B platform for VKM Trading OÜ provides a comprehensive solution tailored to the needs of business clients. The focus on secure, efficient, and user-friendly functionalities ensures that clients can manage their purchasing processes effectively, leading to improved satisfaction and business growth [19][23][30].

4 Work results

4.1 Functional requirements

Admin Panel

The admin panel serves as the central hub for managing the various aspects of the VKM Trading OÜ platform. It provides a comprehensive set of tools for administrators to manage products, user registrations, and company profiles. Key functionalities include:

Product Management: Administrators can add, update, and delete products.

User and Company Management: The admin panel allows for the review and approval of new user and company registrations. This controlled registration process ensures that only legitimate business entities gain access to the platform.

Discount Management: Administrators can assign discounts to specific products and assign the discounts to specific users, facilitating promotional activities and customer-specific pricing strategies.

Product Overview and Storefront

The product overview and storefront functionalities are crucial for providing users with a seamless browsing and shopping experience. These include:

Product Listings: A comprehensive list of products available on the platform, categorised for easy browsing. Users can view detailed product information, including specifications and pricing.

Basket

The basket functionality allows users to select and manage the products they wish to purchase. Key features include:

Quantity and Pricing Management: Users can select the desired quantity of each product, with real-time updates on pricing based on the selected quantity. A built-in calculator helps users determine the number of pallets and packages required.

Order Summary: The cart page provides quantities and prices of the selected products, enabling users to review their order before proceeding to checkout.

User and Company Registration and Authentication

A robust registration and authentication system is essential for ensuring the security and integrity of the platform. This includes:

User Registration: New users must register and provide necessary information. Registrations are reviewed and approved by administrators to ensure legitimacy.

Company Registration: Similar to user registration, new companies must also register and undergo an approval process. This ensures that all entities on the platform are verified businesses.

Authentication: Secure authentication mechanisms, including the use of JSON Web Tokens (JWT), ensure that user data is protected during login and subsequent interactions with the platform.

4.2 Non-functional requirements

Connecting the Final Frontend to the Backend

Currently, only a test frontend is connected to the backend for debugging purposes.

To do: Integrate the fully developed frontend with the backend to ensure seamless interaction and data flow.

Automation of Company Verification for Registration

Currently company registrations are manually verified by administrators.

To do: Develop an automated system to verify company registrations, reducing administrative workload and enhancing security.

Frontend Testing

Currently frontend testing has not yet been implemented.

To do: Implement comprehensive frontend testing to ensure reliability and functionality, using tools like Jest and React Testing Library [1].

Personal page

Currently not implemented

To do: Create a user dashboard where users can view active discounts, personal information, and order history, and manage their accounts.

Calculation of Final Order Price

Currently the final order price calculation is incomplete.

To do: Implement dynamic calculation of the final order price, including taxes, shipping costs, and other relevant factors.

Calculation of Discounted Prices

Currently discounts are not yet integrated into the price calculation.

To do: Develop functionality to apply discounts to product prices and display the discounted prices to users.

News and Newsletter System

Currently not implemented

To do: Create a system for posting news updates and sending newsletters to registered users.

4.3 Application design

The application is designed to be a B2B trading platform for VKM Trading OÜ. Each registration request from companies and users is individually reviewed by a VKM Trading OÜ employee [25].

The user can browse the product catalogue on the store page, where search tools are implemented for convenience. Each product can be viewed individually, displaying all its characteristics [30]. On this page, the user can also add the product to their cart by selecting the required quantity [29]. Orders are made in quantities corresponding to pallets and/or packages, and the built-in calculator assists in this process [30]. The user can then proceed to the cart page to review their order, where the selected quantity and prices of the products are displayed [29].

The administrator has the same capabilities as a regular user but additionally has access to the admin panel. The admin panel contains tools for database management [27]. Here, they can review registration requests, assign discounts to users, view, modify, delete, and add new entries to the database, such as products, discounts, and more [27].

The goal of this project is to establish a foundation for a comprehensive trading platform. At this stage, certain features such as payment processing, Smart ID and ID card authentication, public hosting, and other functionalities that require thorough preparation, cybersecurity, and strong integration with VKM Trading OÜ's business processes are not yet implemented [24].

5 Analysis and conclusions

5.1 Comparison with Existing Software

5.1.1 User Interface Advantages Over Existing Software

The new B2B platform offers several user interface improvements over the existing VKM Trading OÜ website. It provides a more interactive and user-friendly experience, including a shopping cart, real-time product updates, and an intuitive admin panel for easy product management [30]. These features significantly enhance the efficiency and convenience of the purchasing process for users [29].

5.1.2 User Interface Shortcomings Compared to Existing Software

Despite these improvements, the new interface may lack some advanced features found in leading e-commerce platforms, such as personalized user experiences and comprehensive analytics dashboards [24]. Users accustomed to simpler interfaces might find the new system more complex initially, presenting a potential learning curve [30].

5.1.3 Functionality Advantages Over Existing Software

The new platform introduces substantial functional enhancements, including automated order processing, real-time inventory management, and secure sensitive data transmission using JWT [32]. These features streamline business operations and improve overall transaction efficiency, providing significant advantages over the existing static informational site [19][21].

5.1.4 Functionality Deficiencies Compared to Existing Software

However, the platform may still lack certain advanced functionalities present in top-tier B2B solutions, such as integrated payment gateways and sophisticated search capabilities [24]. Initial versions might also lack robust integration with external ERP systems, which are critical for some businesses [24][26].

5.2 Reflection on the work done

The project greatly benefited from several courses taken during the academic program. The "Database" course provided essential knowledge for effective database design and

management, ensuring a robust foundation for the platform's data handling [28]. "ISAI" was invaluable for understanding REST API development and testing, which was crucial for building and verifying the backend services [27]. The "ISAI" course, which covered full-stack development and Vue.js, was particularly useful as Vue.js concepts were similar to React, facilitating a smoother transition and implementation of the frontend [1][29].

5.2.1 Things That Went Well

The database design was executed effectively, providing a robust foundation for data management [28]. Additionally, using Postman for API testing facilitated efficient development and troubleshooting, enabling rapid issue resolution [31]. We successfully transferred the visual designs from Figma to the frontend, maintaining the intended aesthetic and functionality [26]. Working with styles in SCSS was particularly interesting, allowing us to create a visually appealing and cohesive user interface [29].

5.2.2 Things That Went Wrong

We encountered several challenges during the project that contributed to its incomplete status. Communication issues due to differing schedules among team members led to significant delays, as coordinating tasks and meetings became difficult [25]. Additionally, parallel development efforts caused synchronization problems, necessitating frequent adjustments to the backend based on evolving client requirements [25].

Managing two different frontend versions and working separately on these versions created integration challenges, further complicating the development process. Specifically, ensuring consistency between the two versions required additional time and effort, leading to delays in the overall timeline [25].

Learning and implementing new technologies also required more time than anticipated. While courses like "Database," "ISAI," and "ISAI" provided a solid foundation, the practical application of these concepts in a real-world project presented unforeseen difficulties [28][29]. Troubleshooting and debugging new technologies such as React, Node.js, *mobx*, and TypeScript demanded extensive research and problem-solving, extending our development timeline [29][27][30].

Frequent changes in project requirements from the client added another layer of complexity. As new features and adjustments were requested, we had to continually revise our development plans and rework existing components, leading to further delays [24].

Moreover, the integration of new features often uncovered unforeseen technical challenges [24]. The initial lack of a structured development order and clear prioritization of tasks also contributed to inefficiencies. Without a well-defined roadmap, we often found ourselves juggling multiple tasks simultaneously, leading to fragmented focus and slower progress [25].

Overall, these challenges combined with the need to balance academic responsibilities and project deadlines prevented us from completing the project within the planned time frame. However, the experience gained has provided valuable insights that will guide future development efforts [24][25].

5.2.3 Usefulness of the Work Done

The project provided valuable experience in real-world development and project management, as well as proficiency in modern technologies like React, TypeScript, and Node.js [29][27]. These skills are highly applicable in future projects and contribute to professional growth [1][2][30].

Additionally, the project underscored the importance of effective communication and task distribution within the team. Regular meetings and calls were crucial for aligning on project goals, addressing issues promptly, and ensuring everyone was on the same page [25]. The experience highlighted how vital it is to have clear communication channels and well-defined roles to streamline development processes [25]. This project also emphasised the need for regular feedback loops and collaborative problem-solving, which are essential practices for any successful development team [25]. These lessons in teamwork and project coordination will be invaluable in any future professional endeavours [24].

5.2.4 Things to Do Differently

For future projects, improving communication strategies and Git workflows would enhance coordination [25]. Establishing a more structured development order could streamline the process and reduce synchronization issues [25]. Better planning and regular check-ins could mitigate many of the problems encountered [25].

5.3 Further Development

Further development of the platform could involve implementing a payment system to support online transactions, adding Smart ID authentication for enhanced security, and deploying the application using Docker on the client's domain to improve control and scalability [33]. Expanding database management tools could enhance data handling and analytics [28]. Enhancing cybersecurity measures is crucial to protect user data and system integrity [32]. Optimizing the platform for performance improvements and developing more advanced query capabilities to handle complex data requests would also be beneficial [24][27].

6 Summary

This project aimed to develop a comprehensive B2B e-commerce platform for VKM Trading OÜ, significantly enhancing the functionality compared to the existing informational site. Key improvements include an interactive user interface, real-time product management, automated order processing, and secure data transmission [19][30][32]. These advancements mark a substantial step forward in operational efficiency and customer satisfaction [29].

Despite these significant achievements, the project is still ongoing and has not yet been deployed for use. Several challenges have contributed to this delay. Communication issues due to differing schedules among team members led to significant delays in coordination and task execution [25]. Parallel development efforts caused synchronization problems, necessitating frequent adjustments to the backend based on evolving client requirements [25]. Additionally, managing two different frontend versions and working separately on these versions created integration challenges, further complicating the development process [25]. The learning curve associated with new technologies such as React, TypeScript, *mobx* and Node.js also required additional time for troubleshooting and debugging [29].

Future development plans include integrating additional features such as payment systems, advanced security measures, and improved database tools [24][32][28]. These enhancements are crucial for ensuring the platform meets the evolving needs of VKM Trading OÜ and its clients [24]. The anticipated outcome is that the B2B e-commerce platform will transform VKM Trading OÜ's operations, driving efficiency, enhancing customer satisfaction, and positioning the company for future growth in a competitive marketplace [19][21].

In summary, while the project has made considerable progress and introduced significant functional improvements, ongoing efforts are needed to complete the platform [25]. Continued work will focus on addressing the remaining challenges, implementing

planned features, and ensuring seamless operation in a real-world environment [25][27]. This thesis documents both the accomplishments to date and the work still required, providing a roadmap for the project's successful completion [24]. Through this comprehensive approach, VKM Trading OÜ is poised to leverage the full potential of a robust, user-friendly, and efficient B2B e-commerce platform [19][24].

References

- [1] React.js Documentation, „Learn - React”, [Online]. Available: <https://react.dev/learn> [Accessed 23 05 2024].
- [2] TypeScript Documentation, „Docs - TypeScript”, [Online]. Available: <https://www.typescriptlang.org/docs/> [Accessed 12 03 2024].
- [3] Node.js Documentation, „API Documentation - Node.js”, [Online]. Available: <https://nodejs.org/docs/latest/api/> [Accessed 11 05 2024].
- [4] Express.js Documentation, „Installing - Express”, [Online]. Available: <https://expressjs.com/en/starter/installing.html> [Accessed 18 04 2024].
- [5] PostgreSQL Documentation, „Docs - PostgreSQL”, [Online]. Available: <https://www.postgresql.org/docs/> [Accessed 25 05 2024].
- [6] Sequelize ORM Documentation, „Docs - Sequelize v6”, [Online]. Available: <https://sequelize.org/docs/v6/> [Accessed 16 03 2024].
- [7] JWT Documentation, „jsonwebtoken - npm”, [Online]. Available: <https://www.npmjs.com/package/jsonwebtoken> [Accessed 18 04 2024].
- [8] bcrypt Documentation, „bcrypt - npm”, [Online]. Available: <https://www.npmjs.com/package/bcrypt> [Accessed 04 03 2024].
- [9] Docker Documentation, „Docs - Docker”, [Online]. Available: <https://docs.docker.com> [Accessed 20 05 2024].
- [10] Visual Studio Code Documentation, „Docs - Visual Studio Code”, [Online]. Available: <https://code.visualstudio.com/docs> [Accessed 03 03 2024].
- [11] MobX Documentation, „README - MobX”, [Online]. Available: <https://mobx.js.org/README.html> [Accessed 15 05 2024].
- [12] Axios Documentation, „Intro - Axios”, [Online]. Available: <https://axios-http.com/docs/intro> [Accessed 14 05 2024].
- [13] Tailwind CSS Documentation, „Installation - Tailwind CSS”, [Online]. Available: <https://tailwindcss.com/docs/installation> [Accessed 05 05 2024].
- [14] Vite Documentation, „Guide - Vite”, [Online]. Available: <https://vitejs.dev/guide/> [Accessed 06 05 2024].
- [15] ESLint Documentation, „Getting Started - ESLint”, [Online]. Available: <https://eslint.org/docs/latest/use/getting-started> [Accessed 22 04 2024].
- [16] Postman Documentation, „Overview - Postman”, [Online]. Available: <https://learning.postman.com/docs/introduction/overview/> [Accessed 27 04 2024].
- [17] Bootstrap Documentation, „Introduction - Bootstrap v5.3”, [Online]. Available: <https://getbootstrap.com/docs/5.3/getting-started/introduction/> [Accessed 01 05 2024].
- [18] Figma Documentation, „Design Basics - Figma”, [Online]. Available: <https://www.figma.com/resource-library/design-basics/> [Accessed 15 02 2024].
- [19] Ghobakhloo, M., Arias-Aranda, D., & Benitez-Amado, J. (2011). Adoption of e-commerce applications in SMEs. *Industrial Management & Data Systems*, 111(8), 1238-1269.

- [20] Ethelbert, O., Moghaddam, F. F., Wieder, P., & Yahyapour, R. (2017, August). A JSON token-based authentication and access management schema for cloud SaaS applications. In *2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud)* (pp. 47-53). IEEE.
- [21] Colley, D., Stanier, C., & Asaduzzaman, M. (2018, August). The impact of object-relational mapping frameworks on relational query performance. In *2018 International Conference on Computing, Electronics & Communications Engineering (iCCECE)* (pp. 47-52). IEEE.
- [22] Molin, E. (2016). Comparison of single-page application frameworks. *A method of how to compare Single-Page Application frameworks written in JavaScript*.
- [23] Shah, H., & Soomro, T. R. (2017). Node.js challenges in implementation. *Global Journal of Computer Science and Technology*, 17(2), 73-83.
- [24] Sotiropoulos, T., Chaliasos, S., Atlidakis, V., Mitropoulos, D., & Spinellis, D. (2021, May). Data-oriented differential testing of object-relational mapping systems. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)* (pp. 1535-1547). IEEE.
- [25] Masse, M. (2011). *REST API design rulebook: designing consistent RESTful web service interfaces*. " O'Reilly Media, Inc."
- [26] Abhishek, M. K., Rao, D. R., & Subrahmanyam, K. (2022). Framework to deploy containers using kubernetes and ci/cd pipeline. *International Journal of Advanced Computer Science and Applications*, 13(4).
- [27] Podila, P., & Weststrate, M. (2018). *MobX Quick Start Guide: Supercharge the client state in your React apps with MobX*. Packt Publishing Ltd.
- [28] Blair-Early, A., & Zender, M. (2008). User interface design principles for interaction design. *Design Issues*, 24(3), 85-107.
- [29] Martin, R. C. (2009). *Clean code: a handbook of agile software craftsmanship*. Pearson Education.
- [30] Zhang, Y., Vasilescu, B., Wang, H., & Filkov, V. (2018, October). One size does not fit all: an empirical study of containerized continuous deployment workflows. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (pp. 295-306).
- [31] Hellerstein, J. M., Stonebraker, M., & Hamilton, J. (2007). Architecture of a database system. *Foundations and Trends® in Databases*, 1(2), 141-259.
- [32] Jones, M., Bradley, J., & Sakimura, N. (2015). *Json web token (jwt) (No. rfc7519)*.
- [33] Bierman, G., Abadi, M., & Torgersen, M. (2014). *Understanding typescript*. In *ECOOP 2014—Object-Oriented Programming: 28th European Conference, Uppsala, Sweden, July 28–August 1, 2014. Proceedings 28* (pp. 257-281). Springer Berlin Heidelberg.
- [34] Starplast, „Home - Starplast”. [Online]. Available: <https://starplast.ee> [Accessed 15 05 2024].
- [35] Silverston, L. (2011). *The data model resource book, Volume 1: A library of universal data models for all enterprises*. John Wiley & Sons.

Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis¹

We, Andres Vassiljuk & Ilja Kisseljov

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis “Implementation of shopping cart and product functionality on the example of VKM Trading OÜ”, supervised by Viljam Puusep
 - 1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
 - 1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

20.05.2024

¹ The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

Appendix 2 Desktop version Figma model

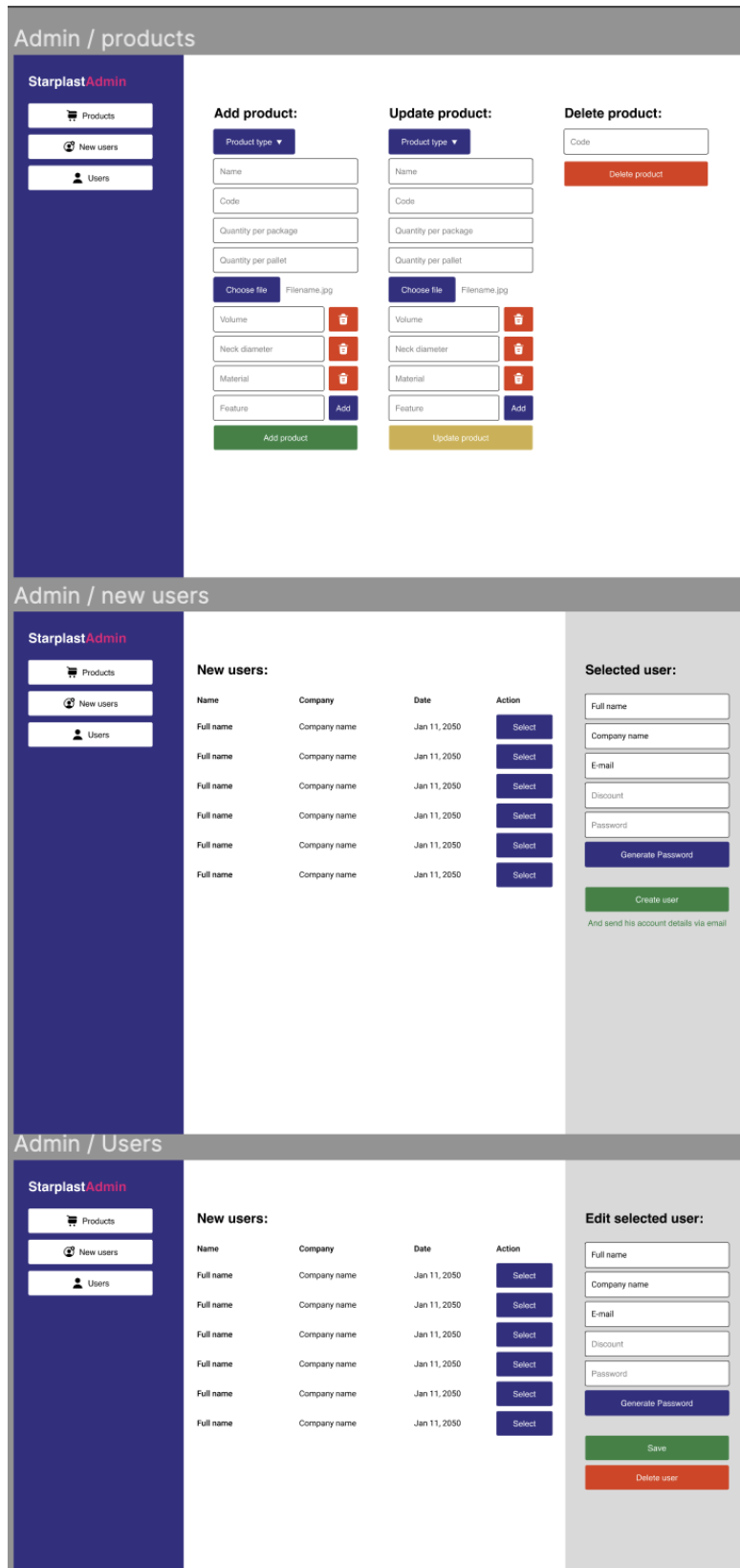


Figure 1 Admin page prototype in Figma

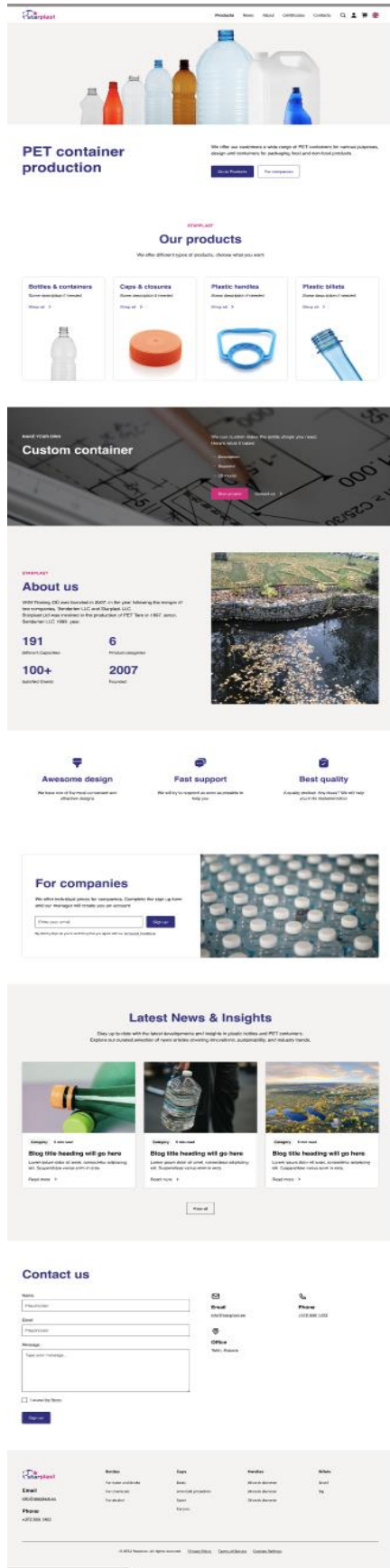


Figure 2 Home Page prototype in Figma

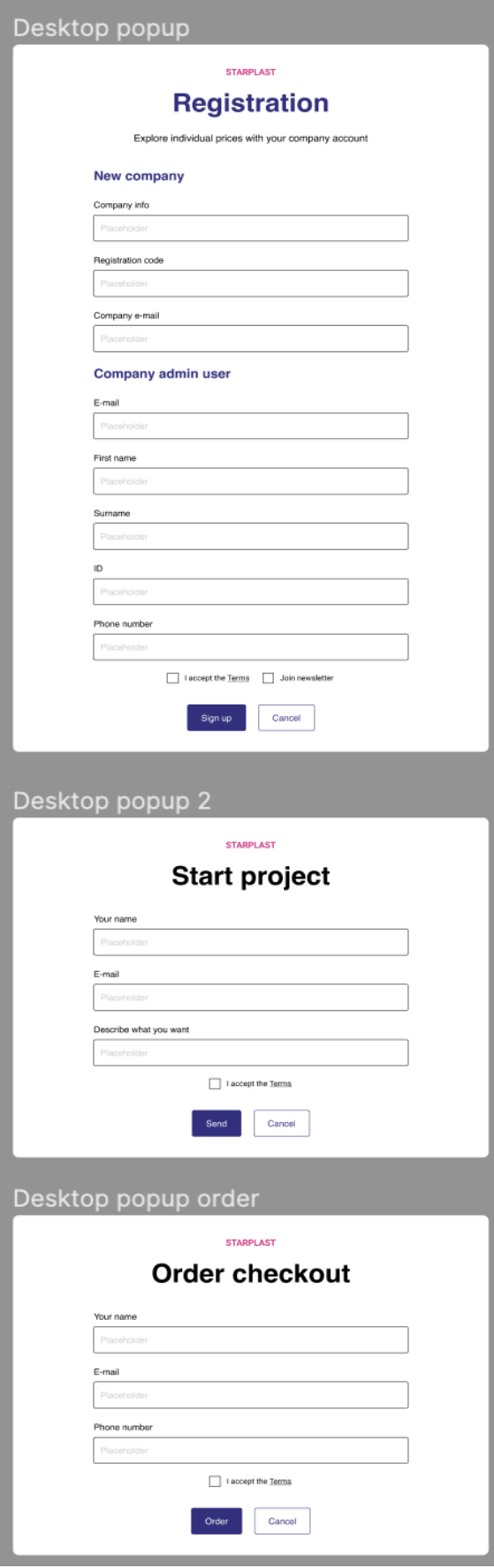


Figure 3 Registration, Project start & Order checkout prototype in Figma

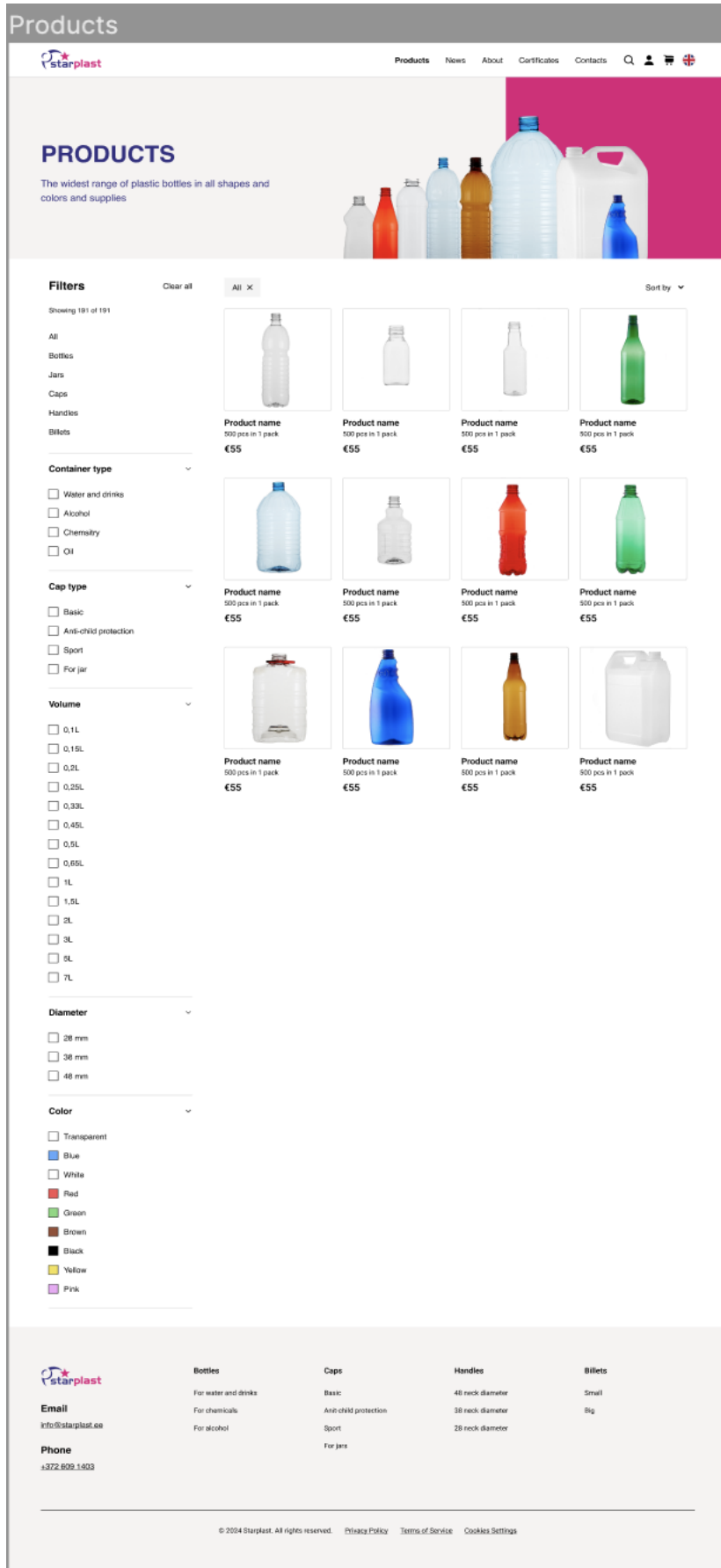


Figure 4 Products page prototype in Figma

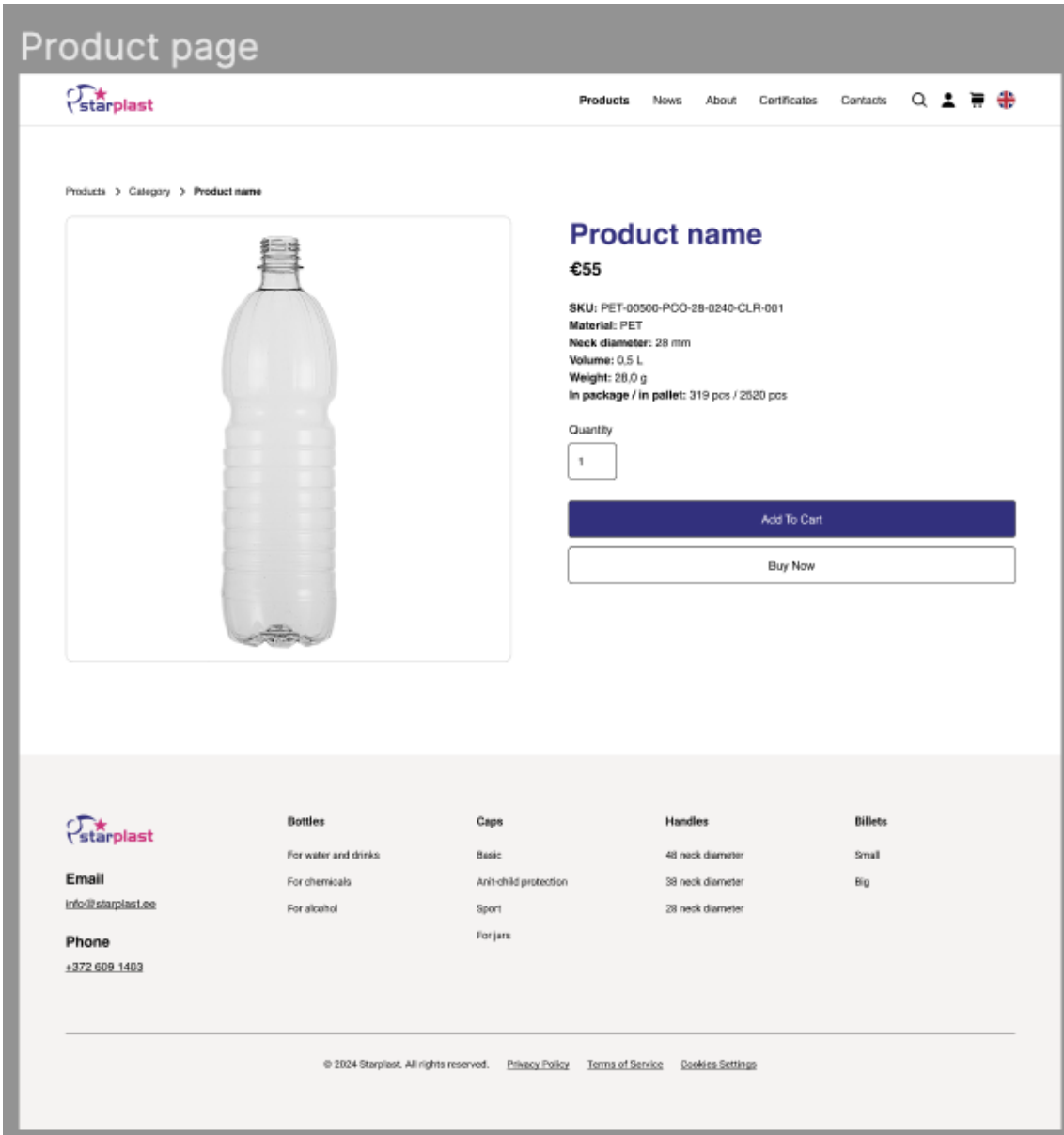


Figure 5 Unit product page prototype in Figma



Figure 6 News page prototype in Figma

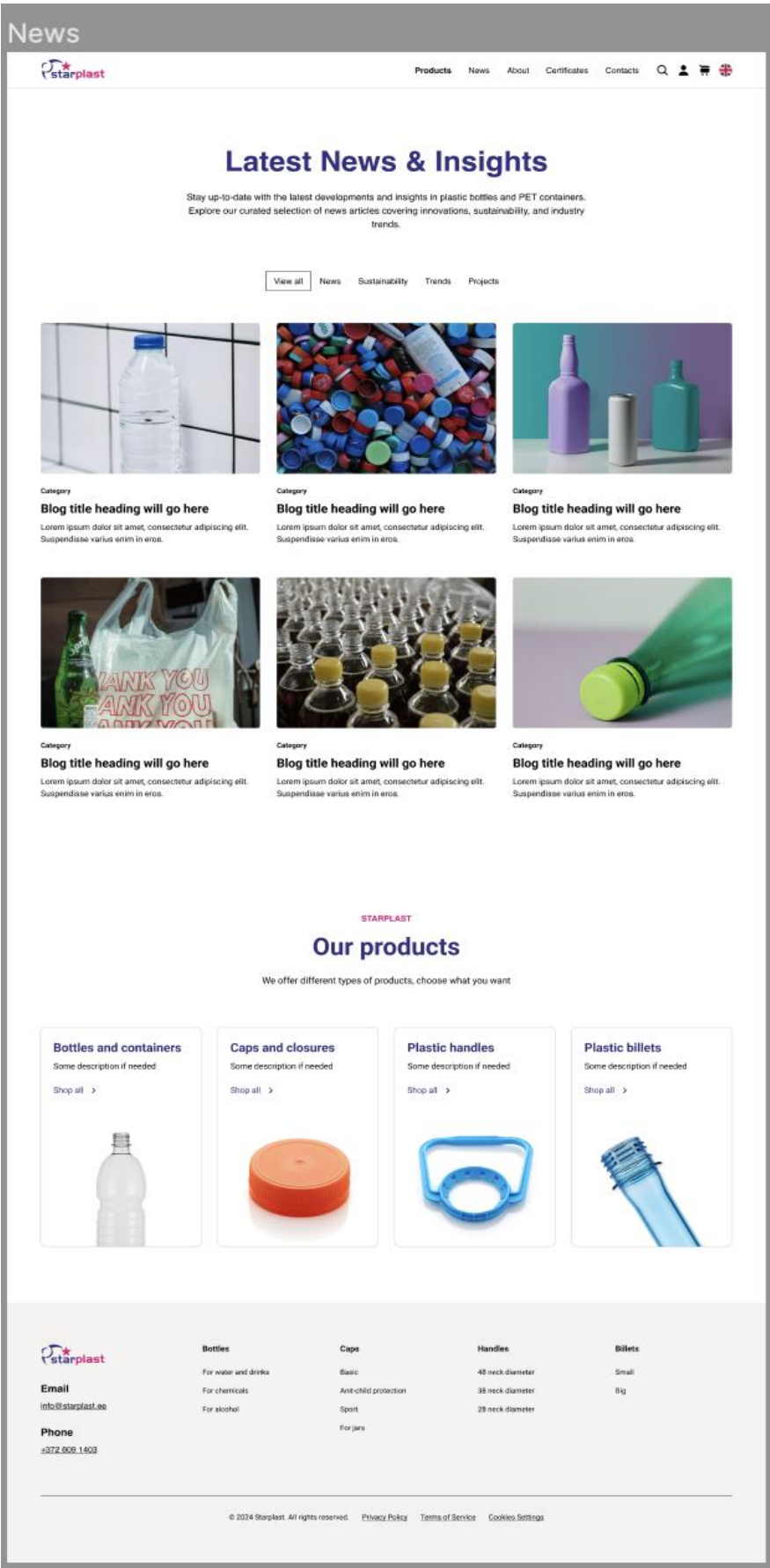


Figure 7 Other news page prototype in Figma

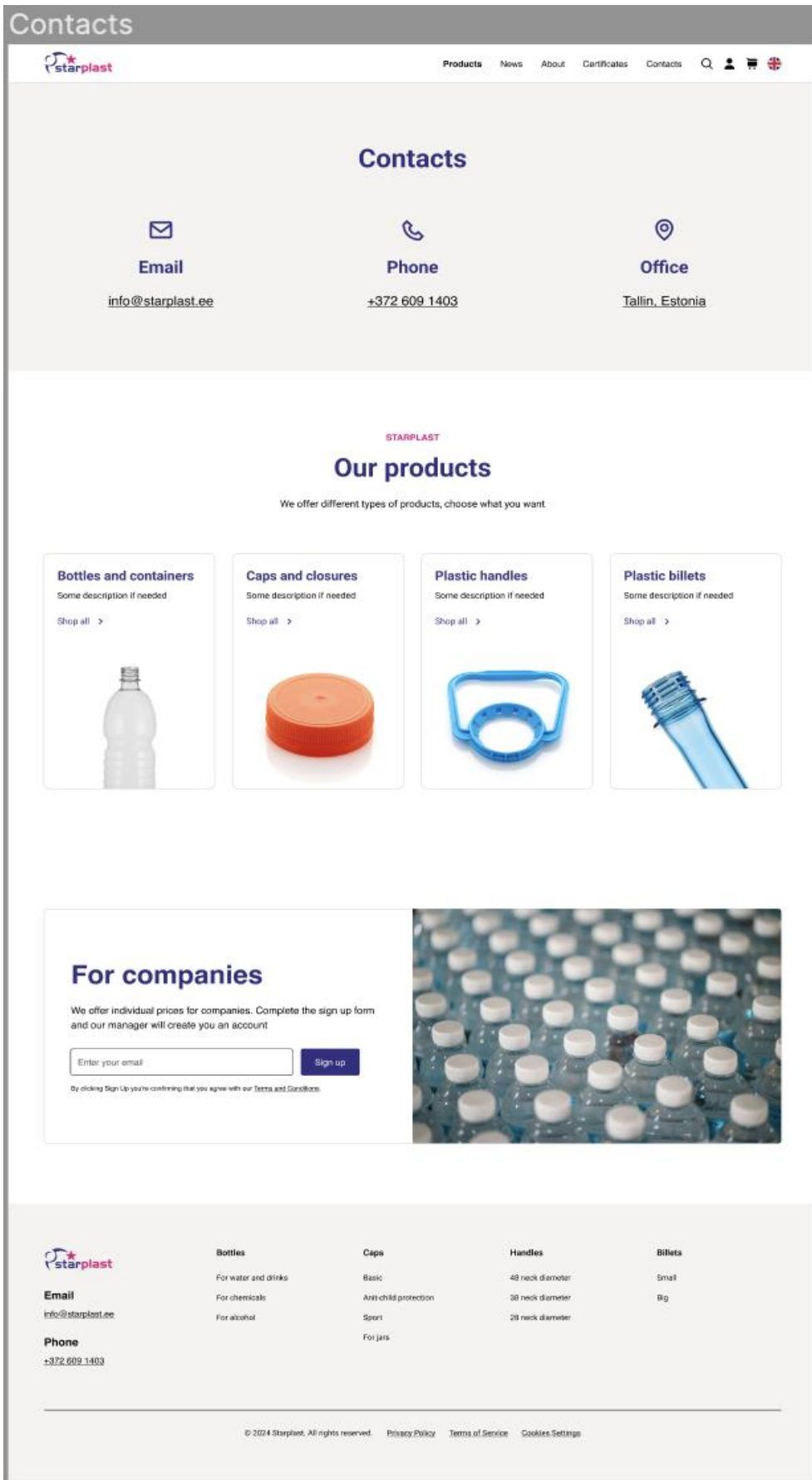


Figure 8 Contacts page prototype in Figma

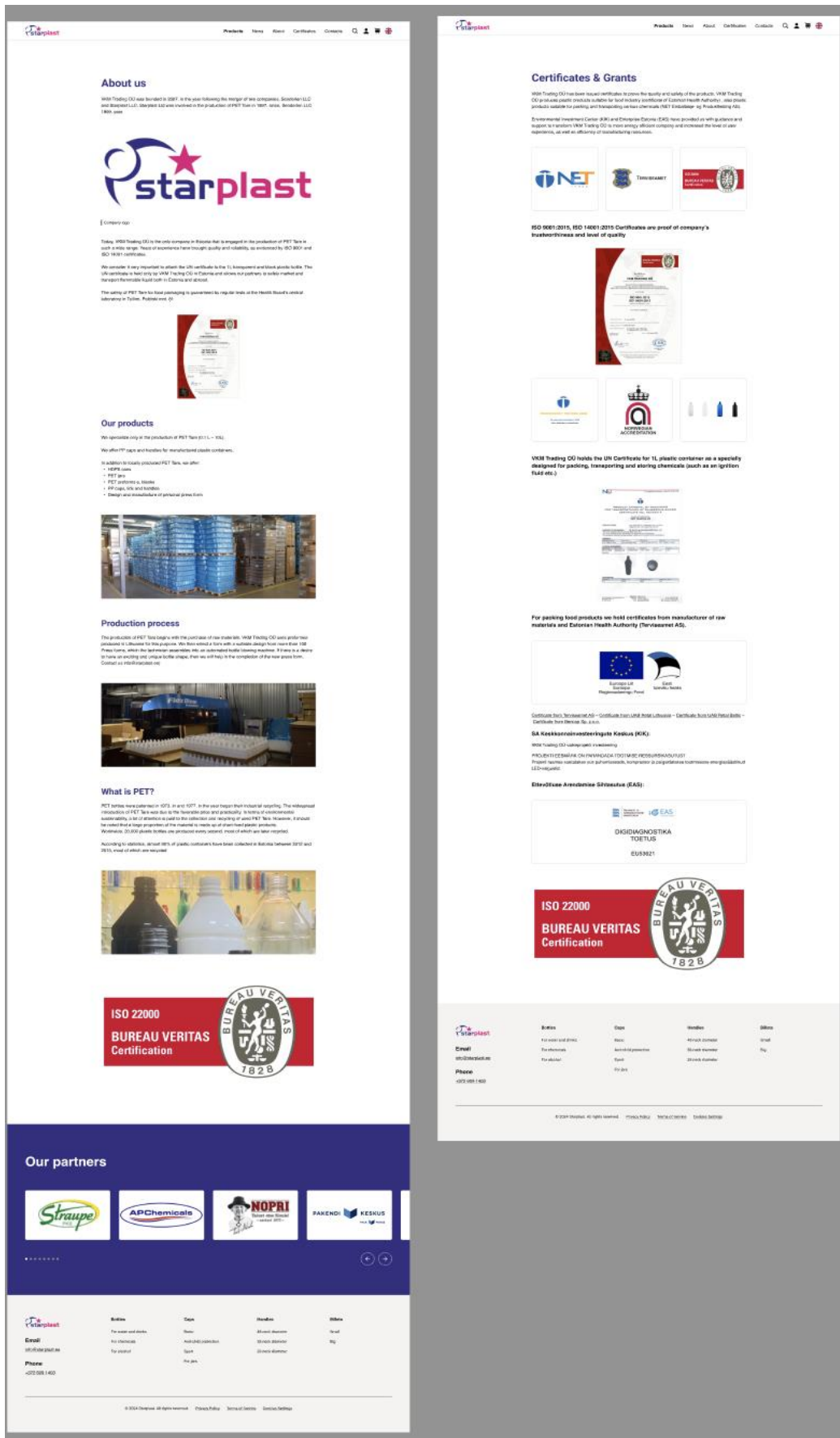


Figure 9 About us & Certificates pages prototype in Figma

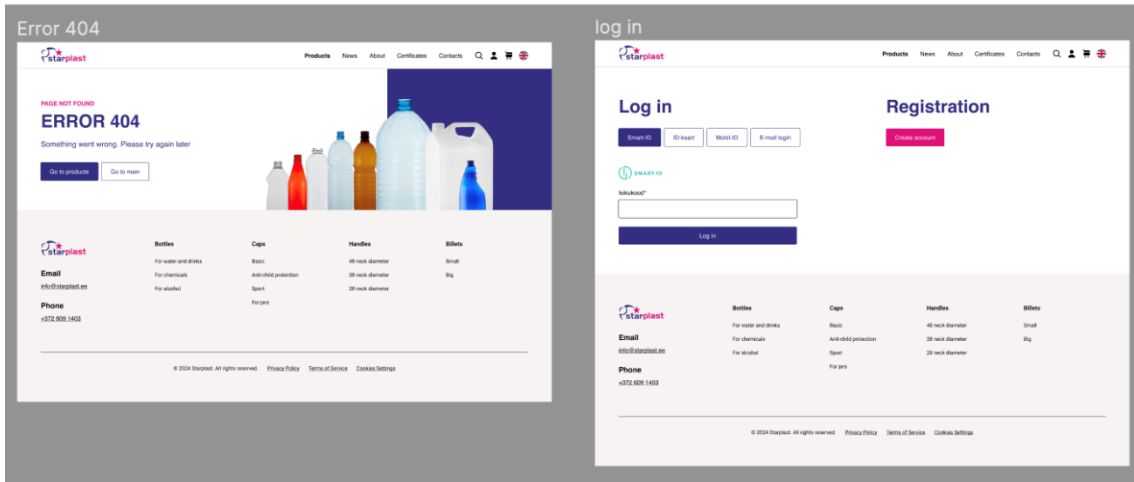


Figure 10 Error 404(Not found) & Log-in pages prototype in Figma

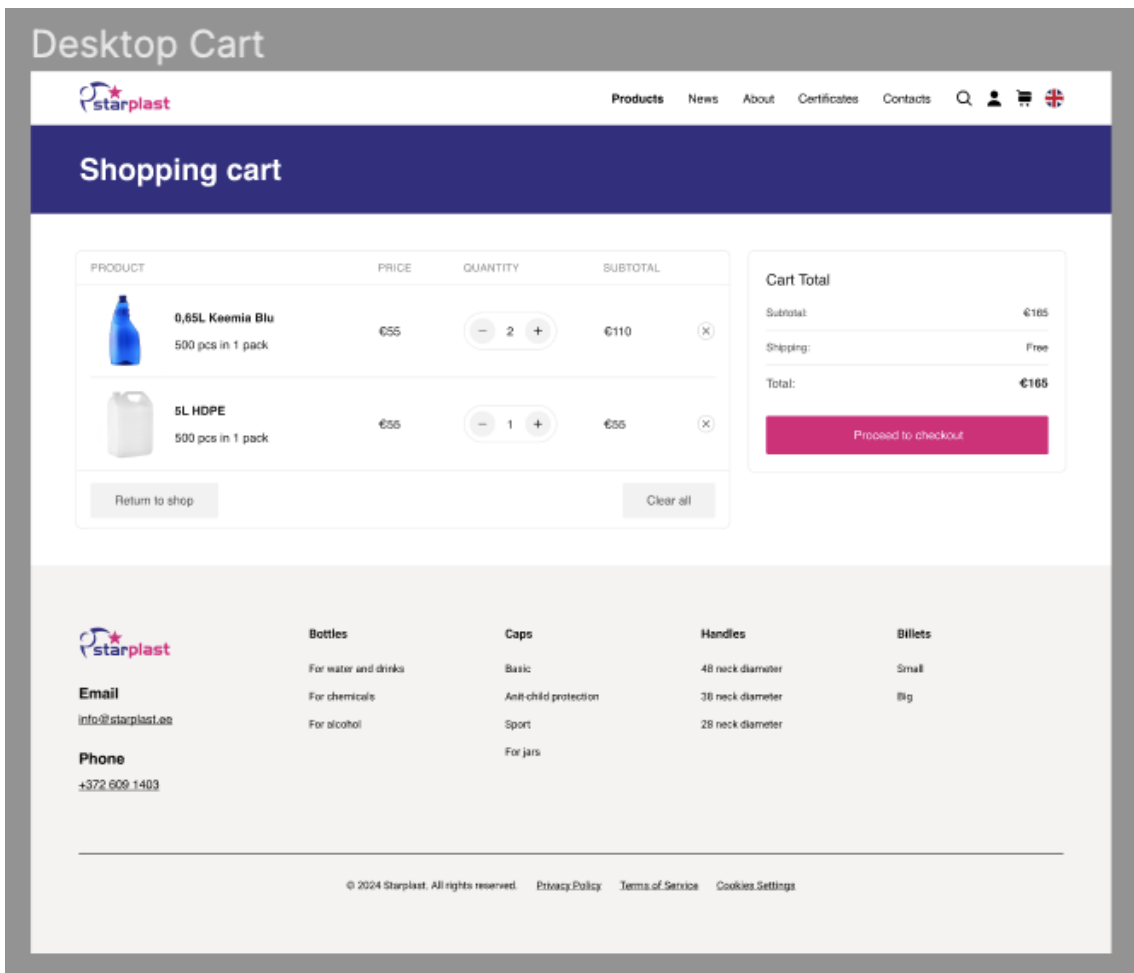


Figure 11 Shopping cart page prototype in Figma

Appendix 3 Phone version Figma model

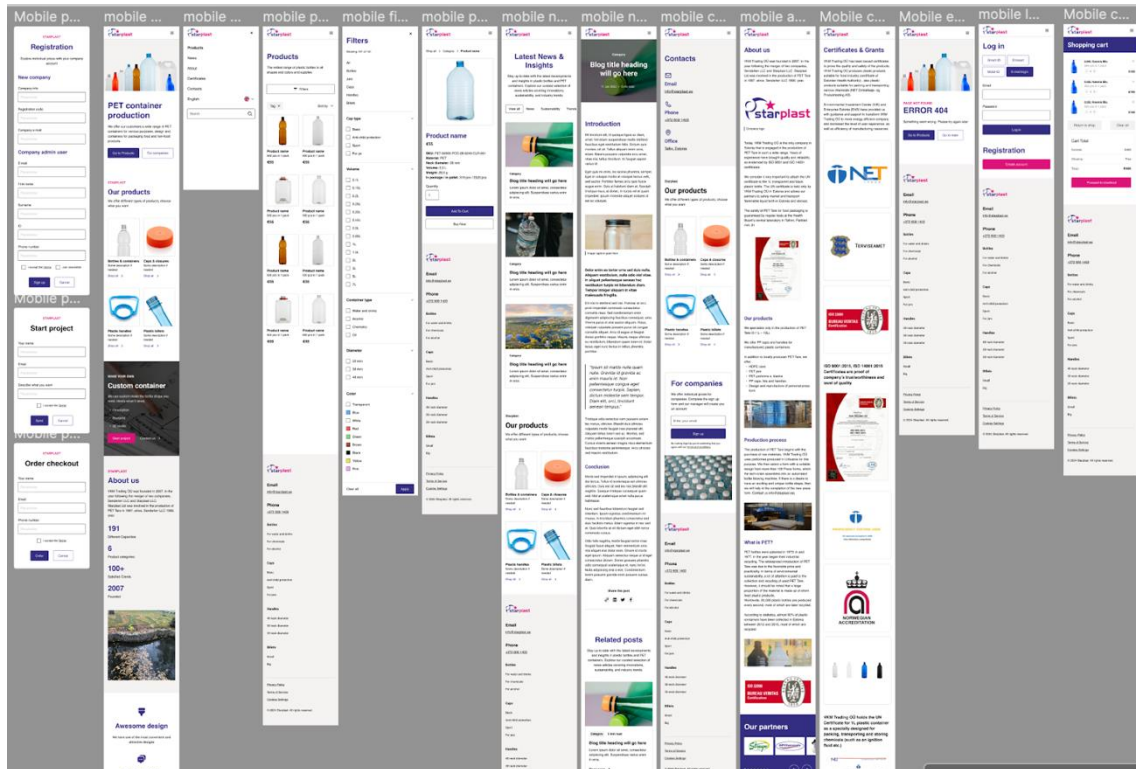


Figure 12 Phone version prototype in Figma

Appendix 4 Endpoints table

Table 2 List of endpoints

Method	Endpoint	Description
POST	api/item	Create new item (Admin)
GET	api/item	Get all items
GET	api/item/:id	Get a specific item
DELETE	api/item/:id	Delete a specific item (Admin)
PUT	api/item/:id	Update a specific item (Admin)
POST	api/itemType	Create new item type (Admin)
GET	api/itemType	Get all item types
GET	api/itemType/:id	Get a specific item type
DELETE	api/itemType/:id	Delete a specific item type (Admin)
PUT	api/itemType/:id	Update a specific item type (Admin)
POST	api/feature	Create new feature (Admin)
GET	api/feature	Get all features

GET	api/feature/:id	Get a specific feature
DELETE	api/feature/:id	Delete a specific feature (Admin)
POST	api/featureType	Create new feature type (Admin)
GET	api/featureType	Get all feature types
GET	api/featureType/:id	Get a specific feature type
DELETE	api/featureType/:id	Delete a specific feature type (Admin)
PUT	api/featureType/:id	Update a specific feature type (Admin)
POST	api/discount	Create new discount (Admin)
GET	api/discount	Get all discounts
GET	api/discount/:id	Get a specific discount
GET	api/discount/byItem/:itemId	Get discounts by item
DELETE	api/discount/:id	Delete a specific discount (Admin)
PUT	api/discount/:id	Update a specific discount (Admin)
POST	api/userDiscount	Create new user discount (Admin)

GET	api/userDiscount	Get all user discounts
GET	api/userDiscount/:id	Get a specific user discount
GET	api/userDiscount/byUser/:userId	Get user discounts by user
DELETE	api/userDiscount/:id	Delete a specific user discount (Admin)
PUT	api/userDiscount/:id	Update a specific user discount (Admin)
GET	api/basket	Get all baskets
GET	api/basket/:id	Get a specific basket
PUT	api/basket/:id	Update a specific basket
POST	api/basketItem	Create new basket item (Admin)
GET	api/basketItem	Get all basket items
GET	api/basketItem/:id	Get a specific basket item
GET	api/basketItem/byBasket/:basketId	Get basket items by basket
DELETE	api/basketItem/:id	Delete a specific basket item
PUT	api/basketItem/:id	Update a specific basket item
POST	api/company	Create new company

GET	api/company	Get all companies
GET	api/company/:id	Get a specific company
DELETE	api/company/:id	Delete a specific company (Admin)
PUT	api/company/:id	Update a specific company (Admin)
POST	api/user/registration	Register new user
POST	api/user/login	Login
GET	api/user/auth	Authorise current user
GET	api/user	Get all users (Admin)
GET	api/user/:id	Get a specific user (Admin)
DELETE	api/user/:id	Delete a specific user (Admin)
PUT	api/user/:id	Update a specific user (Admin)

Appendix 5 Table relations

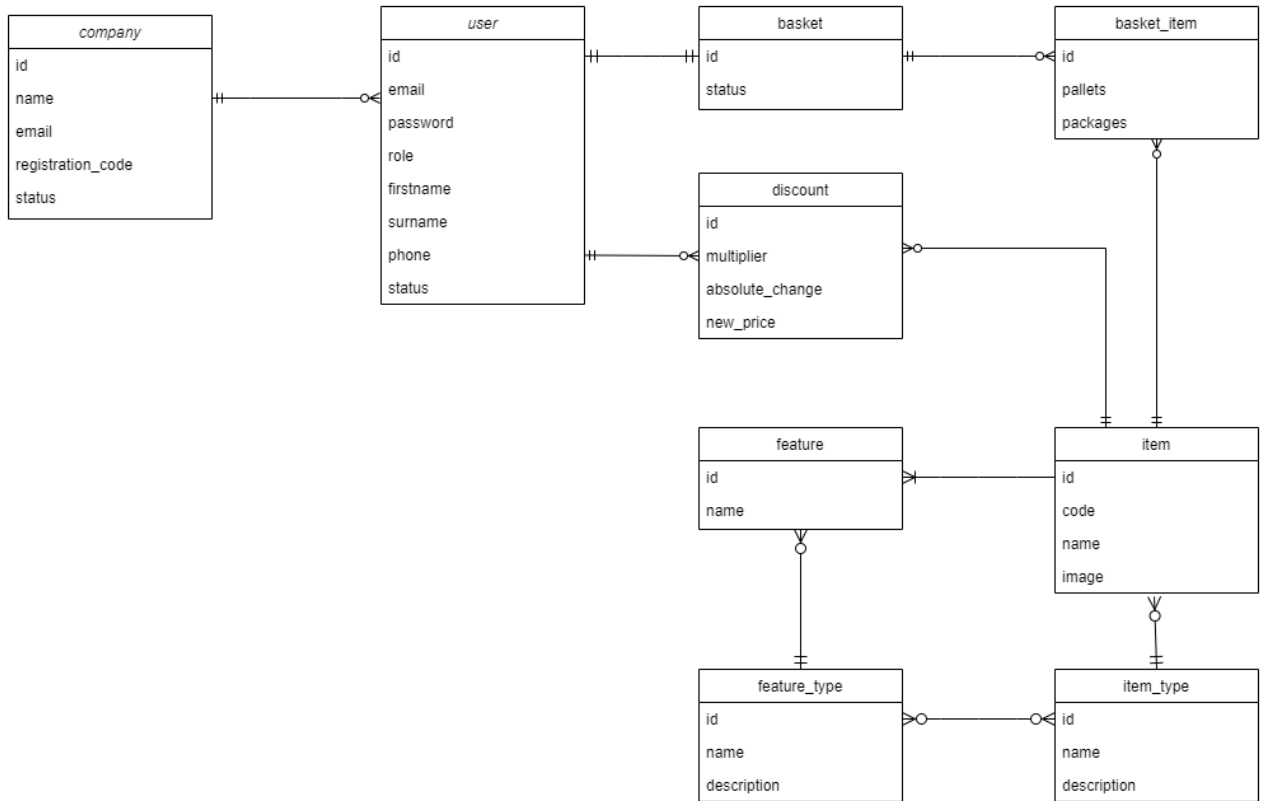


Figure 13 Table relations

Appendix 6 Postman

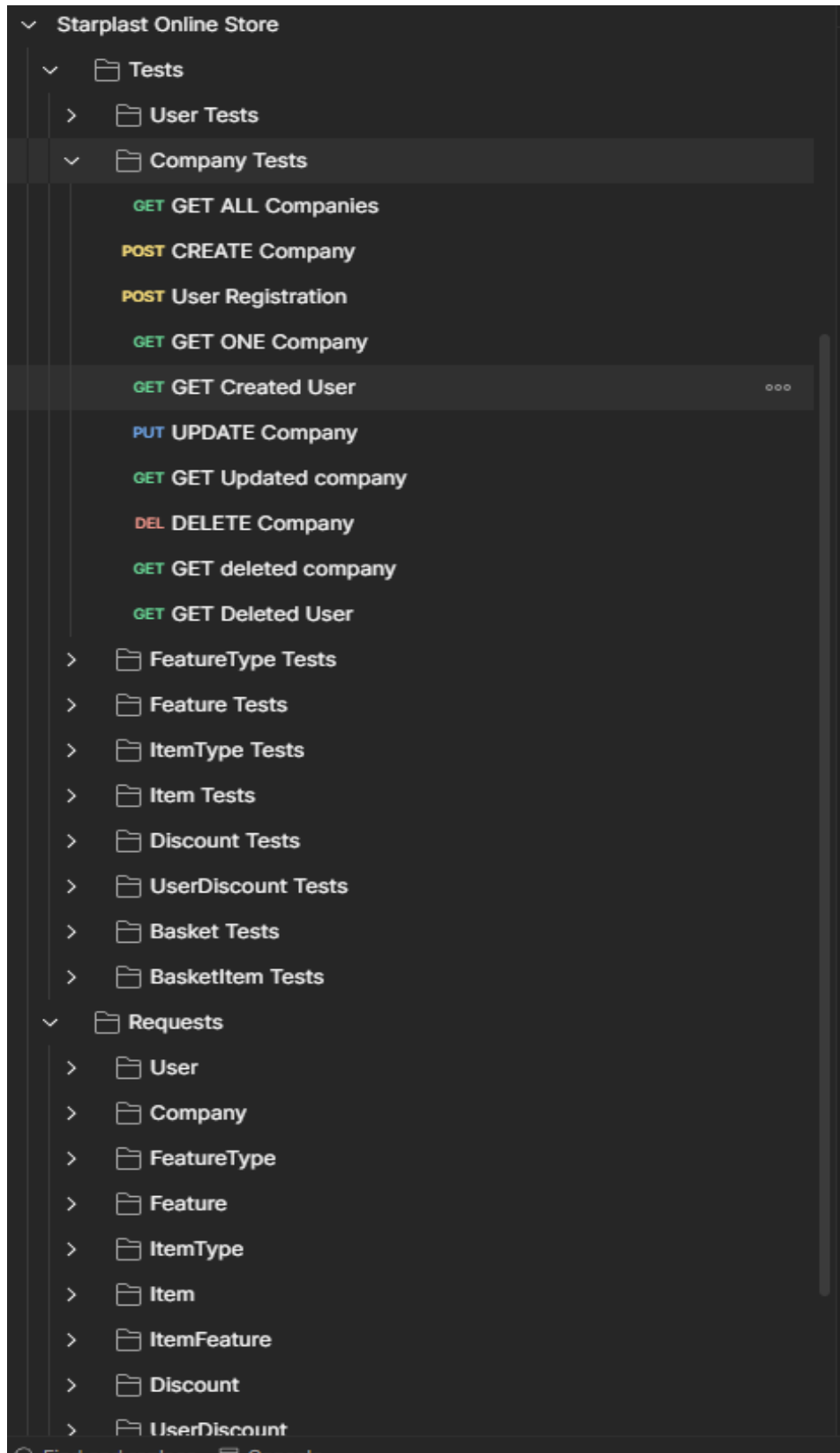


Figure 14 Postman collection structure and a test sequence example

```
Starplast Online Store / Tests / Company Tests / CREATE Company

POST {{baseURL}}/company

Params Authorization Headers (8) Body Scripts Settings

Pre-request
Post-response

1 pm.test("Response status code is 200", function () {
2   pm.expect(pm.response.code).to.equal(200);
3 });
4
5 pm.test("Response time is less than 200ms", function () {
6   pm.expect(pm.response.responseTime).to.be.below(200);
7 });
8
9 pm.test("Registration code is a non-empty string", function () {
10  const responseData = pm.response.json();
11
12  pm.expect(responseData).to.be.an('object');
13  pm.expect(responseData.company.registration_code).to.be.a('string').and.to.have.lengthOf.at.least(1,
14    "Registration code should be a non-empty string");
15 });
16 pm.test("Status is a non-empty string", function () {
17  const responseData = pm.response.json();
18
19  pm.expect(responseData).to.be.an('object');
20  pm.expect(responseData.company.status).to.be.a('string').and.to.have.lengthOf.at.least(1, "Status should be a
21    non-empty string");
22 });
23 pm.test("Request fails due to data duplication", function () {
24  const { method, url, headers, body } = pm.request;
25
26  pm.sendRequest({
27    url: url.toString(),
28    method: method,
29    header: headers,
30    body: body
31  }, function (err, response) {
32    pm.expect(response).to.have.status(400);
33  });
34 });
35 pm.test("Company default status is PENDING", function () {
36   pm.expect(pm.response.json().company.status).to.eql("PENDING")
37 });
38
39 pm.test("Set test variables", function () {
```

Figure 15 Example of a Postman test (POST api/company)

Appendix 7 Backend

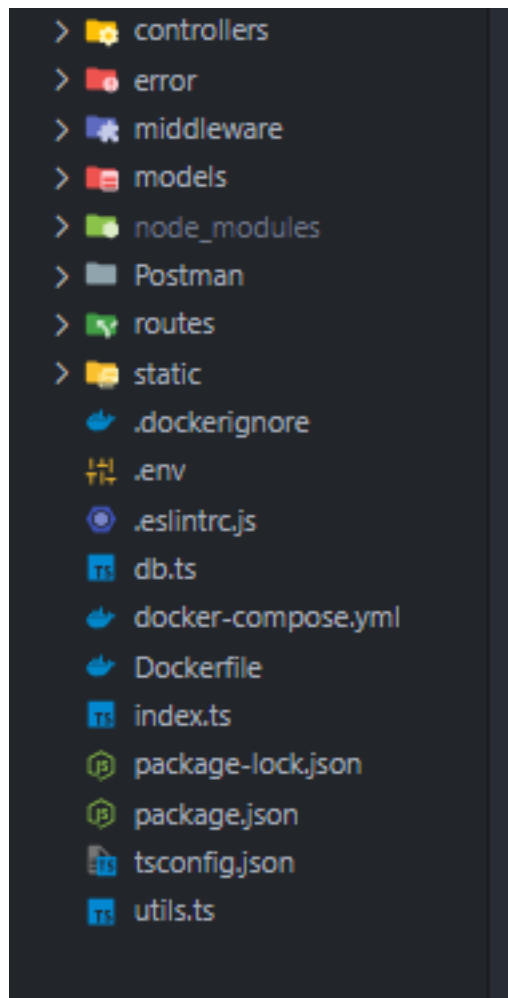


Figure 16 Backend file structure

```
46   async getOne(req, res, next){
47     const {id} = req.params
48     try {
49       const company = await Company.findOne(
50         this: {
51           where: {id}
52         }
53       )
54       if(!company){
55         return next(ApiError.notFound("Company with this ID does not exist"))
56       }
57       return res.json(company)
58     } catch (error) {
59       next(ApiError.badRequest(error.message))
60     }
61   }
```

Figure 17 Example of a request

```

1  import {Router} from 'express'
2  import userController from '../controllers/userController'
3  import authMiddleware from '../middleware/authMiddleware'
4  import checkRole from '../middleware/checkRoleMiddleware'
5
6  const router = Router()
7
8  router.post('/registration', userController.registration)
9  router.post('/login', userController.login)
10 router.get('/auth', authMiddleware, userController.check)
11 router.get('/', checkRole('ADMIN'), userController.getAllUsers)
12 router.get('/:id', checkRole('ADMIN'), userController.getOneUser)
13 router.put('/:id', checkRole('ADMIN'), userController.updateUser)
14 router.delete('/:id', checkRole('ADMIN'), userController.delete)
15
16 export default router
17
18 |

```

Figure 18 Example of a router

```

1  import jwt from "jsonwebtoken"
2
3  export default function(req, res, next){
4      if(req.method === "OPTIONS"){
5          next()
6      }
7      try {
8          const token = req.headers.authorization.split(' ')[1]
9          if(!token){
10             return res.status(401).json({message: "Not authorized"})
11          }
12          const decoded = jwt.verify(token, process.env.SECRET_KEY)
13          req.user = decoded
14          next()
15      } catch (e) {
16          res.status(401).json({message: "Not authorized"})
17      }
18  }

```

Figure 19 Example of a middleware

Appendix 8 Test Frontend

Items ✕

Search by Name

<input checked="" type="checkbox"/>	ID	Code	Name	Quantity per Package	Quantity per Pallet	Image	Base Price	Item Type ID
<input type="checkbox"/>	abbef9cf-dded-47bb-bb25-8318c389d77c	test bottle 1 code	test bottle 1	1000	5000	fb594776-96ec-449e-8691-8ed97bad5680.jpg	0.02	1
<input type="checkbox"/>	d509333d-54da-4062-8df8-de844033408c	TEST_06	1.5 L Liviko Clear6	100	800	d51b1195-6bc8-4aa0-9ff8-8f45aad9f9e1.jpg	0.04	1
<input checked="" type="checkbox"/>	ebea9b88-bab8-459e-9068-d613916ee6fc	PET-00500-PCO-28-003	0.5L Soome Clear	264	2312	62688cbf-7bd5-4bc9-b752-b56709d71512.jpg	0.05	1
<input type="checkbox"/>	d9afe815-b583-499b-b87a-5abf67e72299	TEST_10	TEST BOTTLE 10	100	800	90a26d3b-6e31-41f3-9d5b-b29c1e8d8c7b.jpg	0.01	11




1 of 4 row(s) selected.

Delete Edit Previous Next

Add Item Close

Figure 20 Example of a table in admin panel in test frontend


Starplast Connect to Database Login

Bottle		
Bottle Cap		
test item type 1		
test itemType5		
test itemType8		
test itemType10		
TEST ITEM TYPE 1		
test itemType7		

test bottle 1 1.5 L Liviko Clear6 0.5L Soome Clear

Figure 21 Items fetched from the backend displayed in the test frontend

Starplast Connect to Database Admin Logout



1,5 L Liviko Clear6
 ID: d509333d-54da-4062-8df8-de844033408c
 Code: TEST_06
 Quantity per package: 100
 Quantity per pallet: 800
 Base Price: 0.04

Price: 140

Total:

Packages:

Pallets:

Add to Basket

Description

Neck Diameter: 28 mm

Volume: 1,5 L

Material: PET

Color: Clear

Figure 22 Item detailed view and item count calculator in test frontend

Edit Item ✕

Bottle ▾

Выберите файл Файл не выбран

Add new feature

Material ▾	PET ▾	Delete
Volume ▾	0.5 L ▾	Delete
Neck Diameter ▾	28 mm ▾	Delete
Color ▾	Clear ▾	Delete

Close

Save Changes

Figure 23 Example of an edit form in admin panel in test frontend

Starplast [Connect to Database](#) [Admin](#) [Logout](#) [Basket](#)

BASKET



 1,5 L Liviko Clear6	Packages: 3 Pallets: 4	Price: 140	<input type="button" value="X"/>
 0,5L Soome Clear	Packages: 7 Pallets: 1	Price: 208	<input type="button" value="X"/>

Figure 24 Basket in test frontend

Appendix 9 Frontend external design

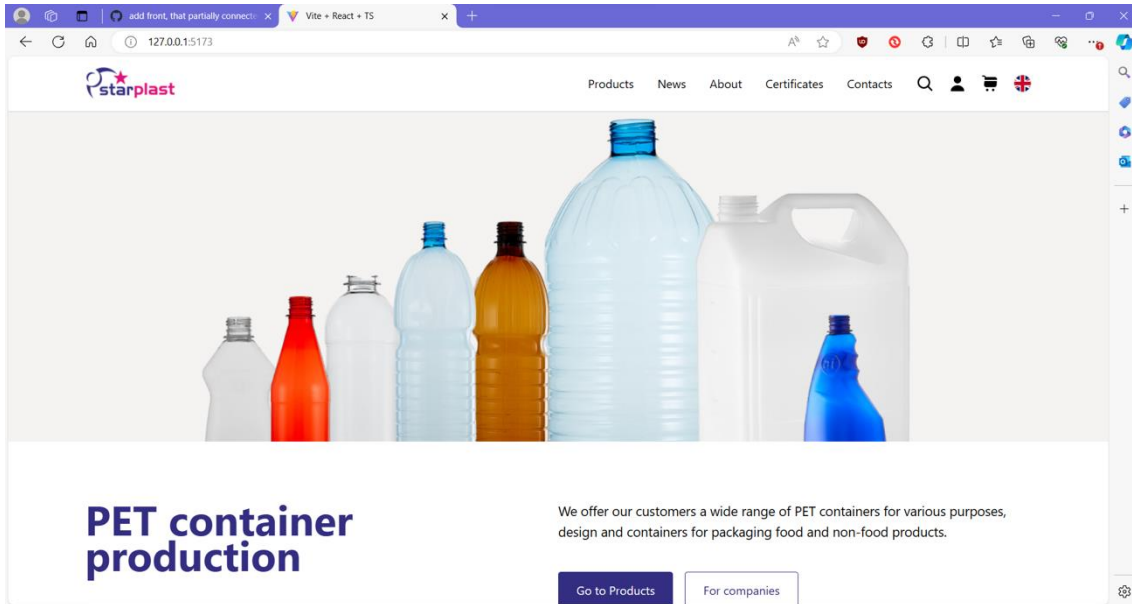


Figure 25 Home page in frontend external design

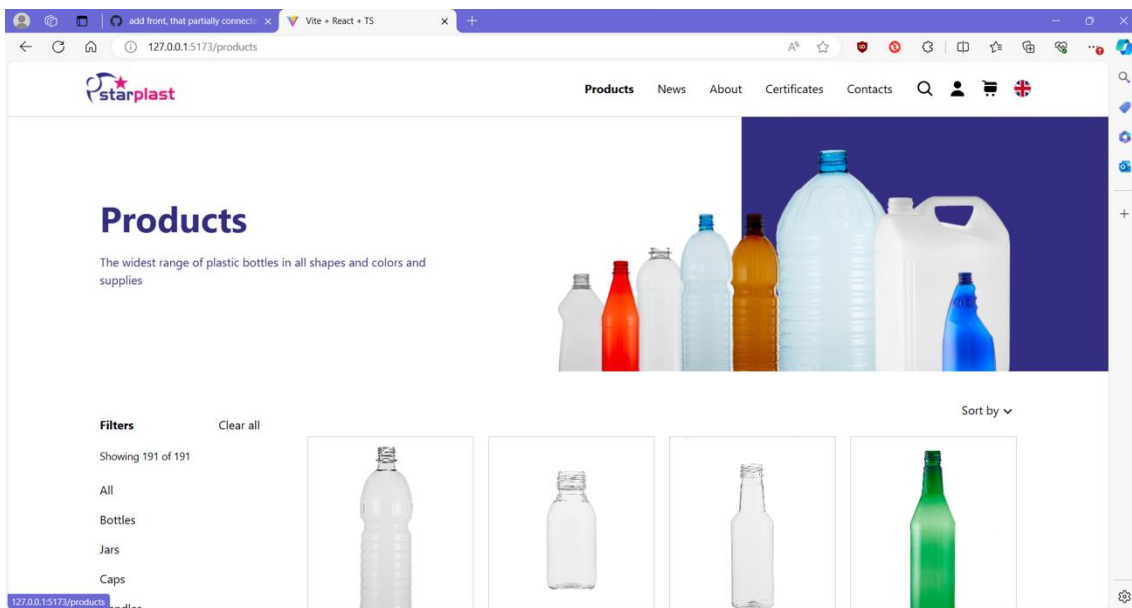


Figure 26 Product page in frontend external design

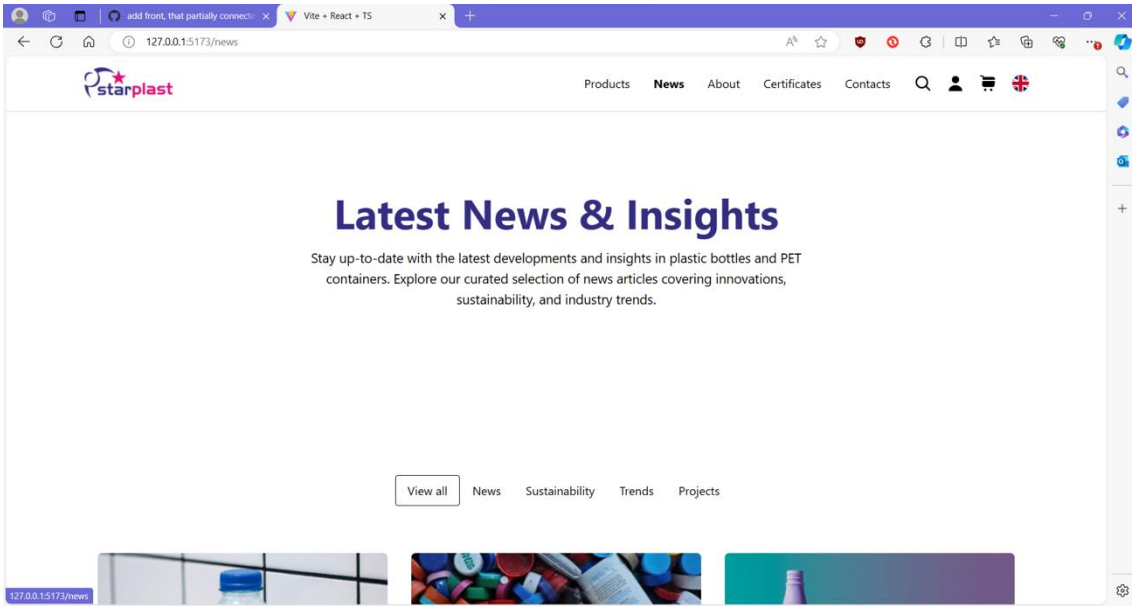


Figure 27 News page in frontend external design

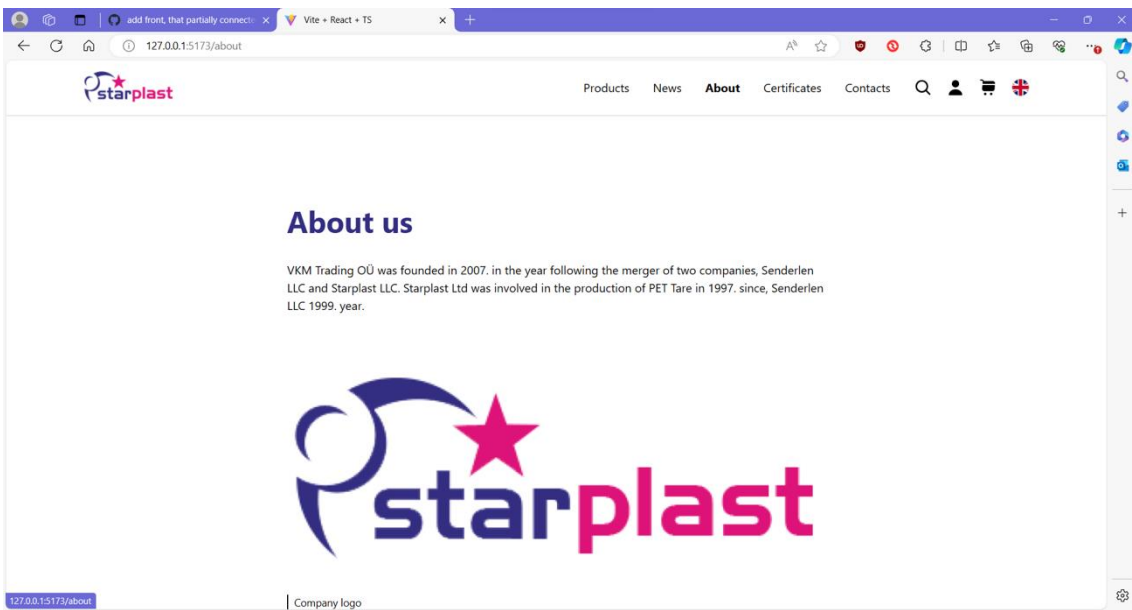


Figure 28 About us page in frontend external design

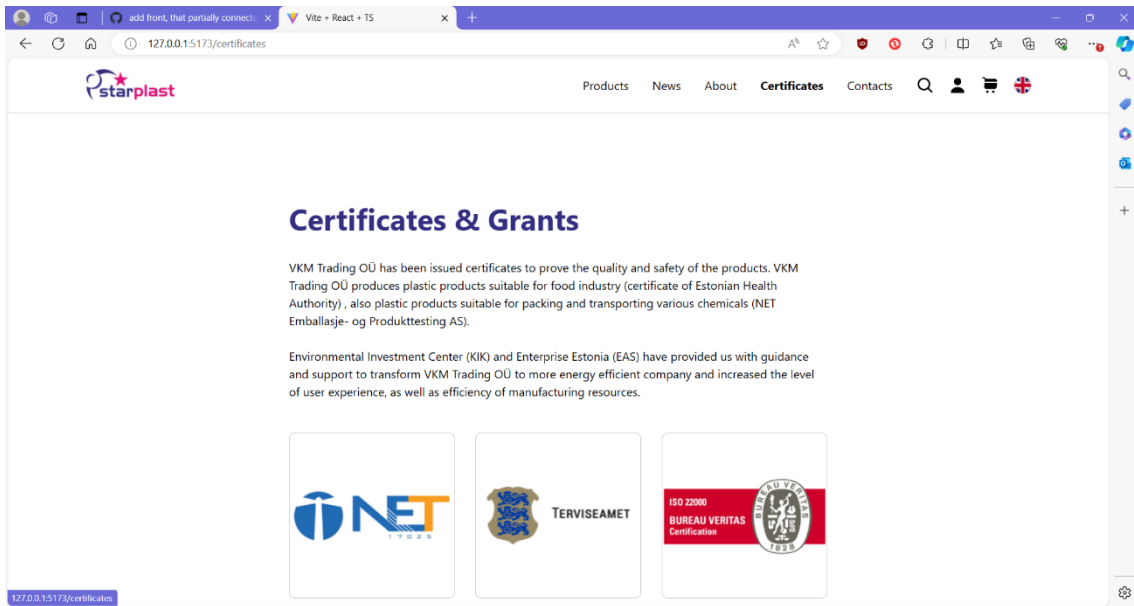


Figure 29 Certificates page in frontend external design

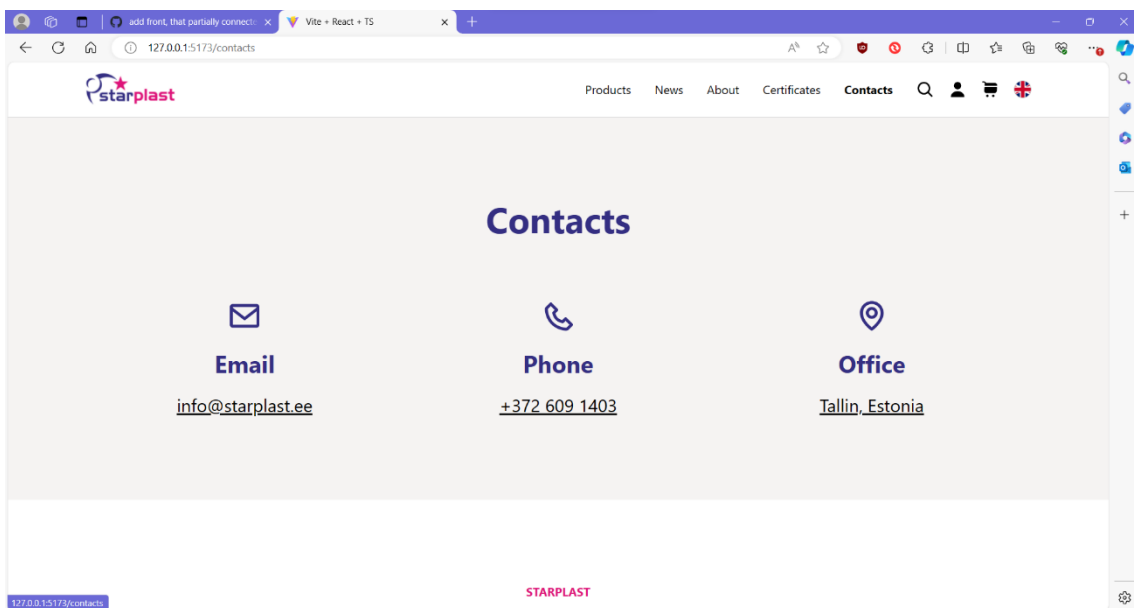


Figure 30 Contact page in frontend external design

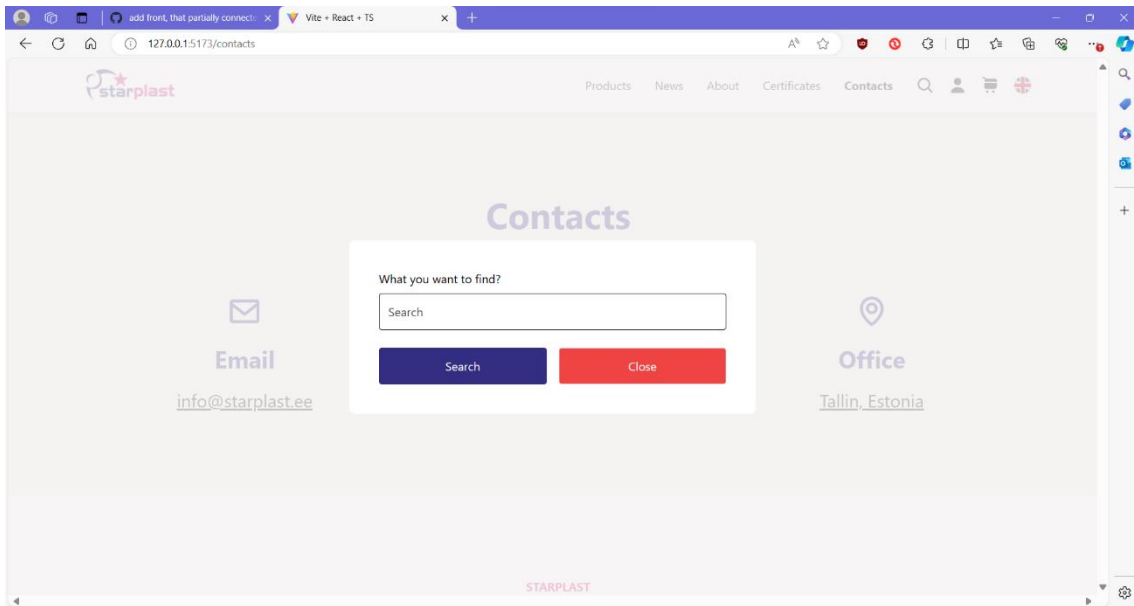


Figure 31 General finder in frontend external design

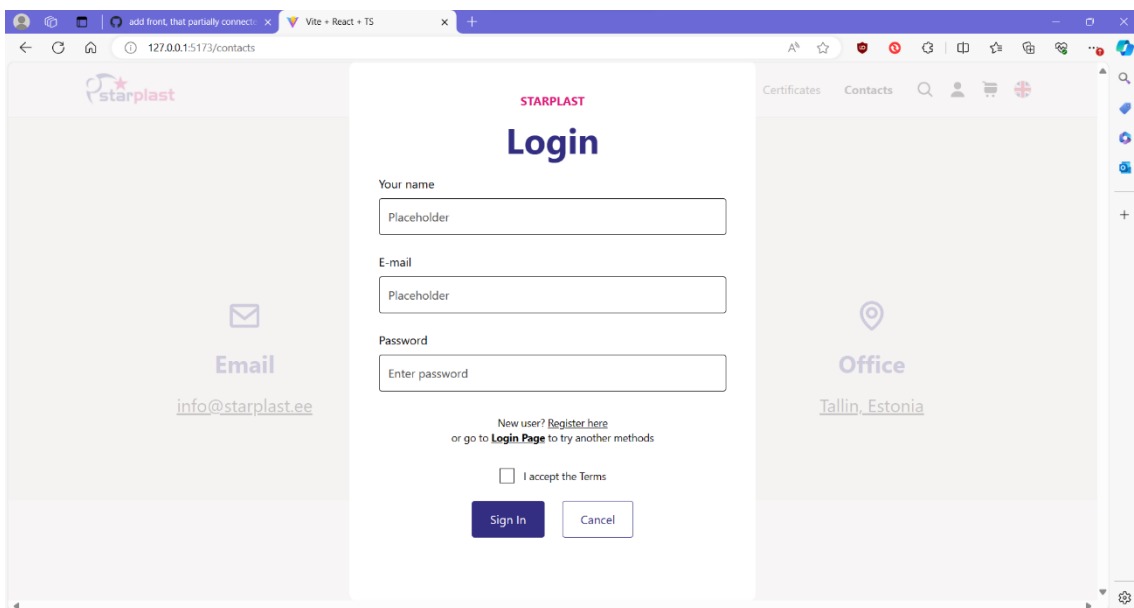


Figure 32 Login in frontend external design

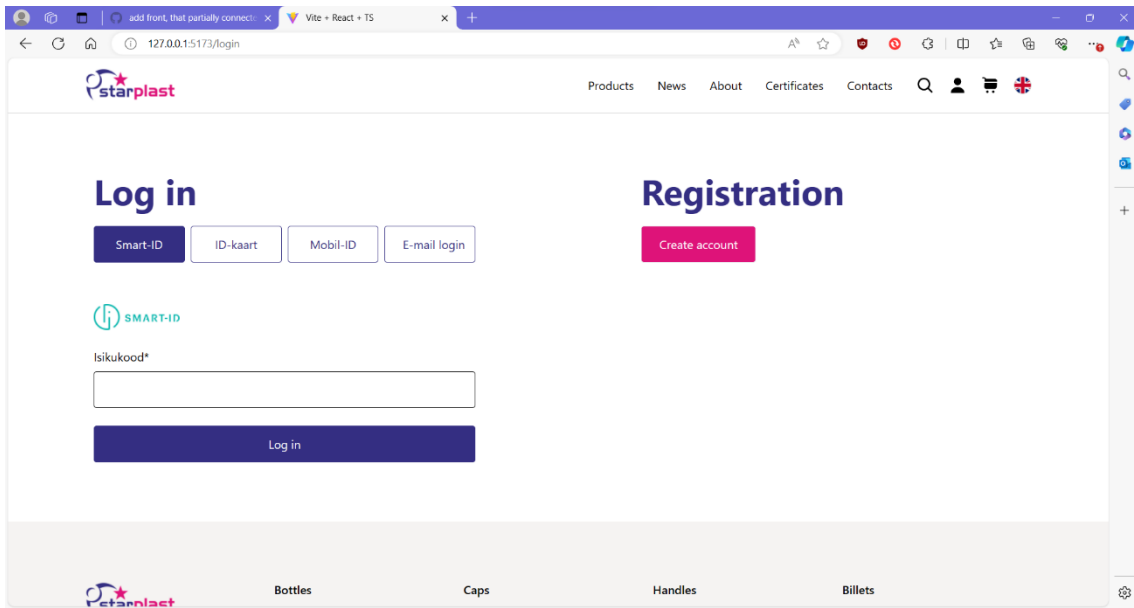


Figure 33 Login page in frontend external design

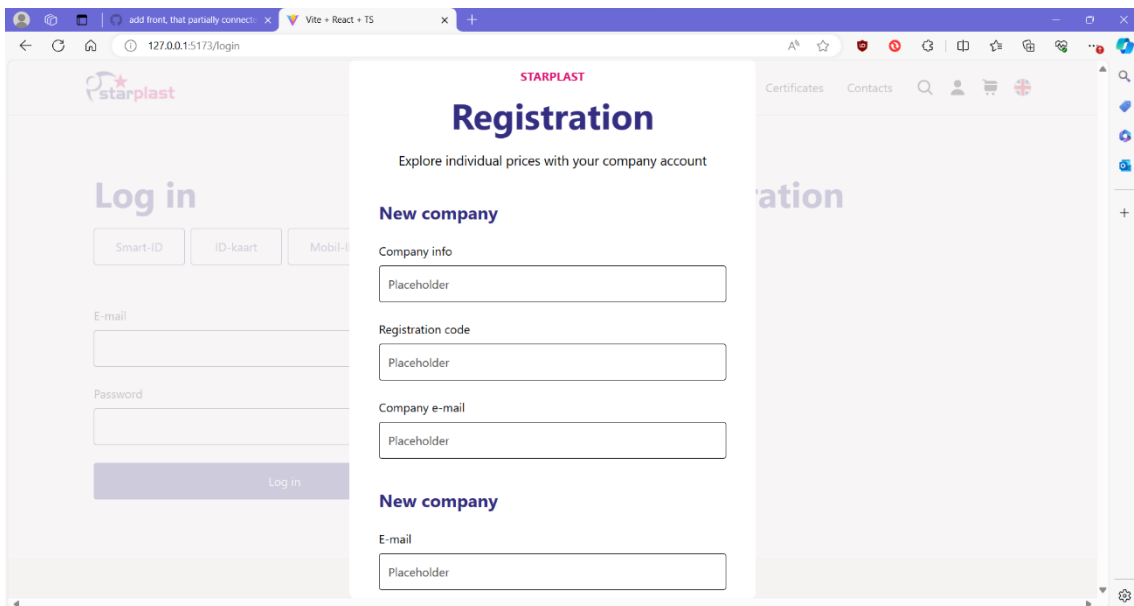


Figure 34 Registration in frontend external design

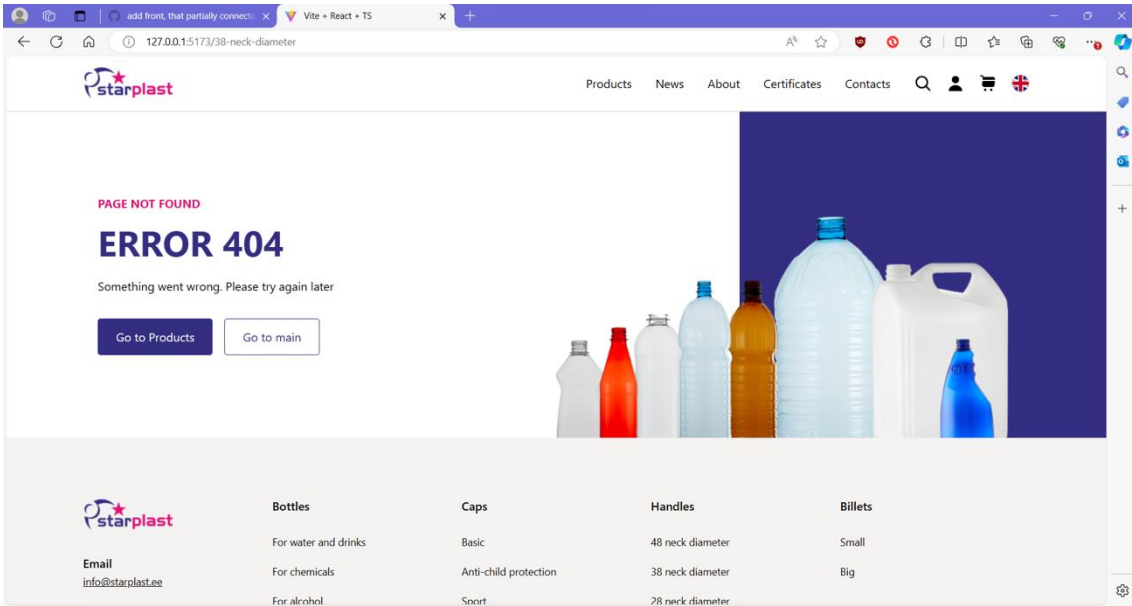


Figure 35 Error 404 (Not found) page in frontend external design

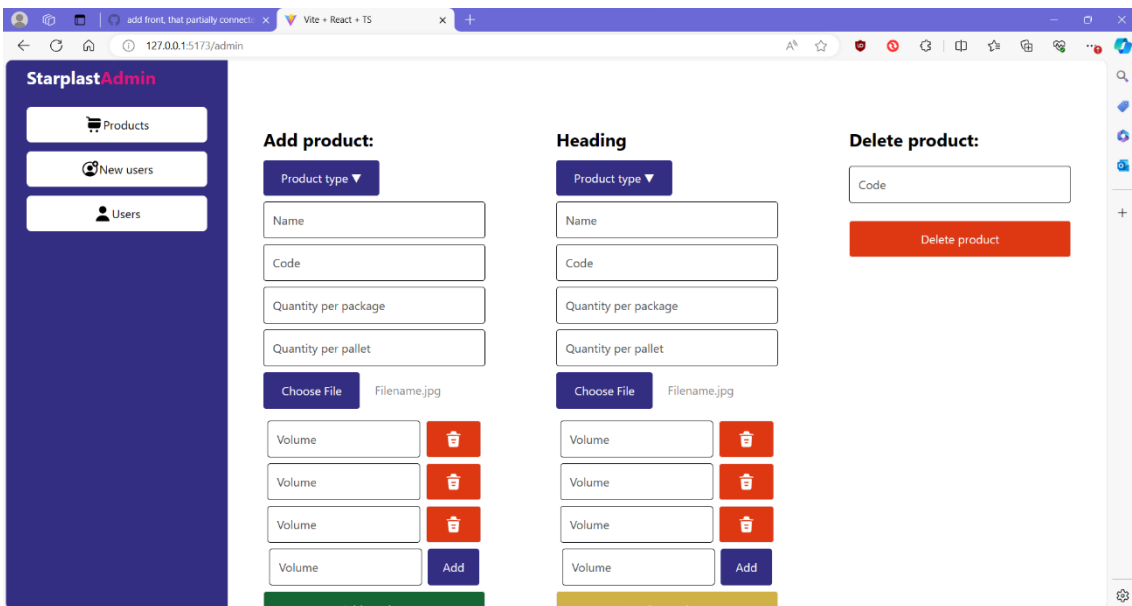


Figure 36 Admin panel Products page in frontend external design

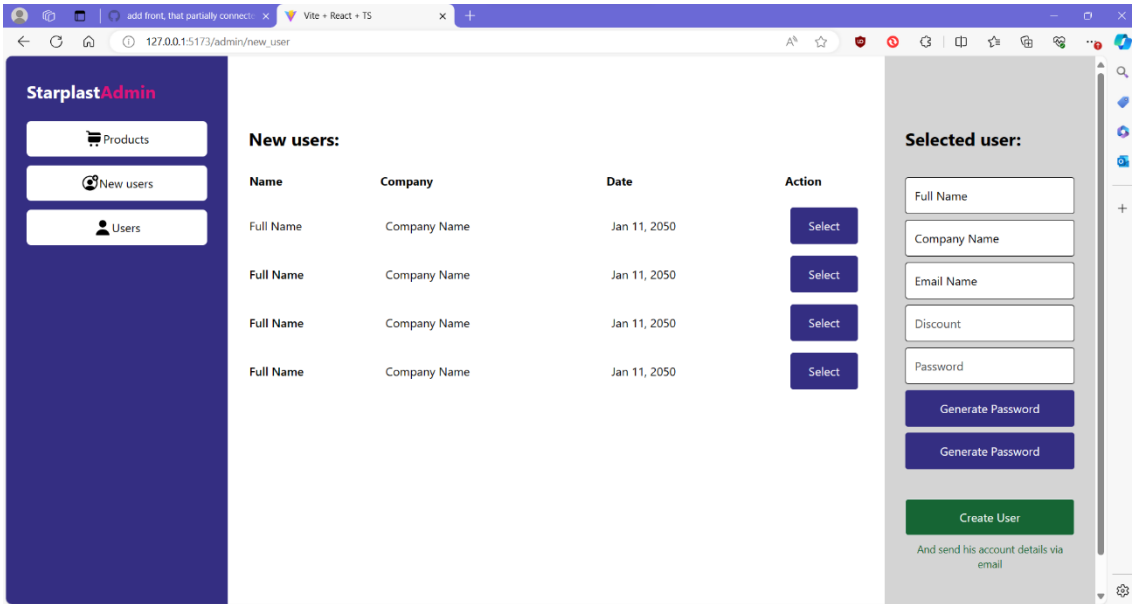


Figure 37 Admin panel New User page in frontend external design

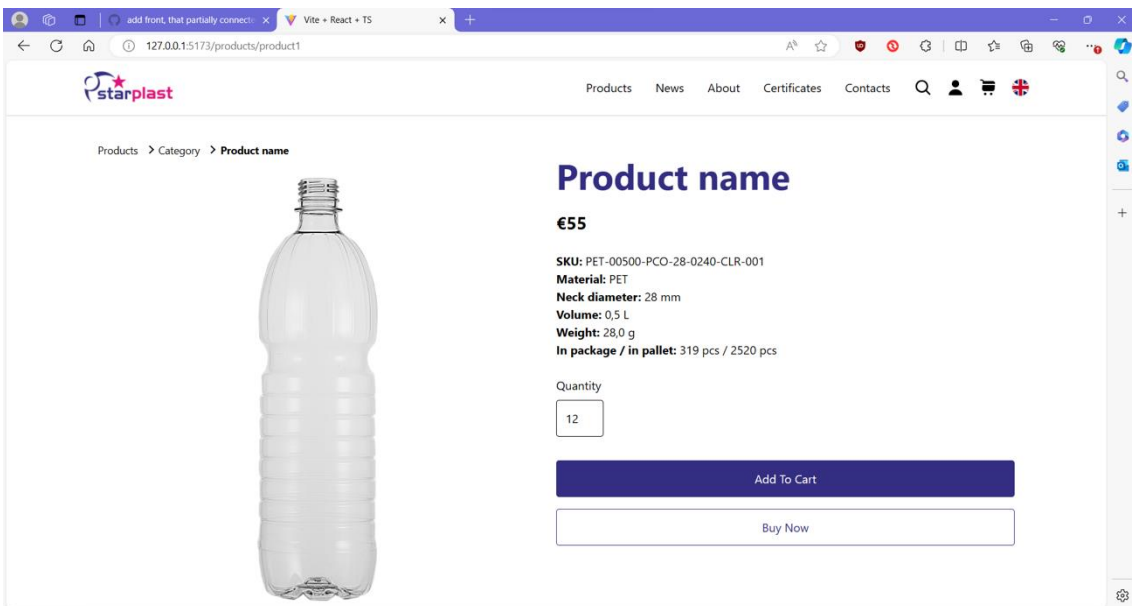


Figure 38 Unit product (item) page in frontend external design

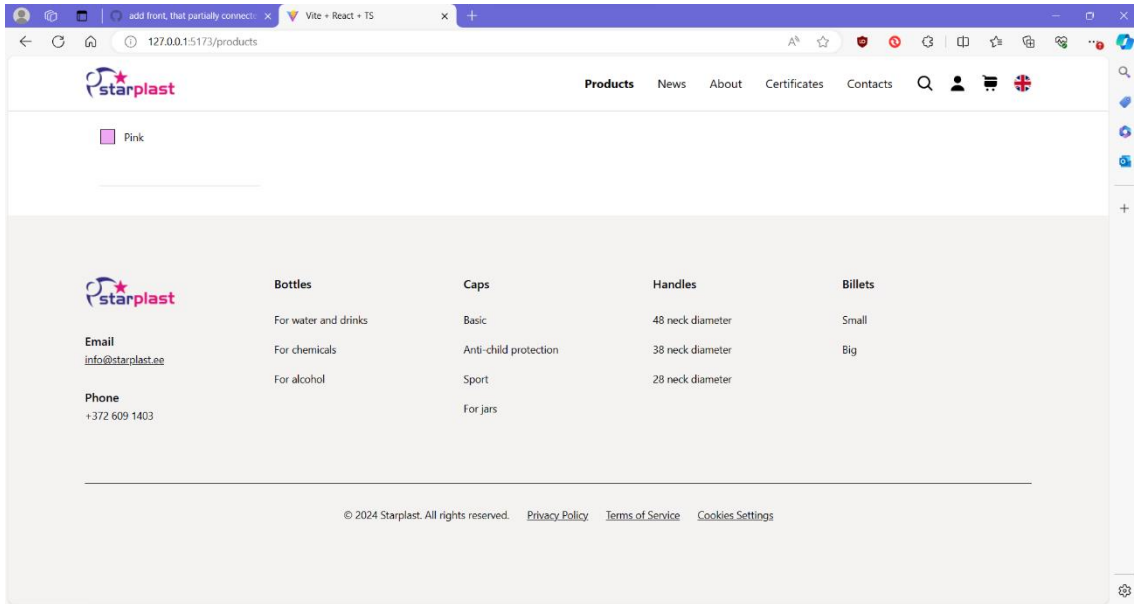


Figure 39 Footer in frontend external design