

UNIVERSITY OF TARTU  
Faculty of Science and Technology  
Institute of Computer Science  
Computer Science Curriculum

Martin Hans Keskküla

# Developing an E-commerce Platform for the Estonian Market

Bachelor's Thesis (9 ECTS)

Supervisor: Mohamad Gharib, Phd

Tartu 2024

# **Developing an E-commerce Platform for the Estonian Market**

## **Abstract:**

In response to the shortage of e-commerce platforms catering to Estonia, this thesis presents the development of a proof-of-concept solution aimed at providing comprehensive support to users. Prior to platform development, an investigation into the primary competitors in the market was conducted, elucidating the essential features desired by users. The resultant platform encompasses all necessary functionalities, striving to offer a holistic solution. A similar competitor, ShopRoller, was identified within the Estonian market, showcasing similarities in appearance yet lacking certain features present in the developed platform. While the developed application boasts greater feature completeness, it is noted to be less mature in terms of user experience (UX). User testing revealed a preference for enhanced user-friendliness, indicating potential adoption if such improvements were implemented.

## **Keywords:**

Web Application, Ruby, Rails, PostgreSQL, e-commerce

**CERCS:** P175 Informatics, systems theory

## **E-poe platvormi arendamine Eesti turule**

### **Lühikokkuvõte:**

Vastusena Eestile suunatud e-kaubanduse platvormide vähesusele esitatakse käesolevas lõputöös proovilahenduse väljatöötamine, mille eesmärk on pakkuda kasutajatele iga-külget tuge. Enne platvormi arendamist uuriti esmaseid konkurente turul, selgitades välja kasutajate poolt soovitud olulised funktsioonid. Saadud platvorm hõlmab kõiki vajalikke funktsioone, püüdes pakkuda terviklikku lahendust. Eesti turul tuvastati pea-

mine konkurent ShopRoller, mis oli välimuselt sarnane, kuid millel puudusid teatud funktsioonid, mis on olemas väljatöötatud platvormil. Kuigi väljatöötatud rakendus on funktsioonide poolest täiuslikum, on see kasutajakogemuse (UX) poolest vähem välja arendatud. Kasutajate testimine näitas, et eelistatakse suuremat kasutajasõbralikkust, mis viitab potentsiaalsele kasutuselevõtule, kui sellised parandused rakendatakse.

**Võtmesõnad:**

Veebirakendus, Ruby, Rails, PostgreSQL, e-kommerts

**CERCS:** P175 Informaatika, süsteemiteooria

# Contents

<b>1</b>	<b>Introduction</b>	<b>6</b>
<b>2</b>	<b>Analysis of competing platforms</b>	<b>8</b>
2.1	Shopify . . . . .	8
2.2	WooCommerce . . . . .	9
2.3	ShopRoller . . . . .	10
2.4	Comparison with planned application . . . . .	11
<b>3</b>	<b>Application requirements</b>	<b>13</b>
3.1	Functional requirements (FR) . . . . .	13
3.2	Non-functional requirements (NFR) . . . . .	15
<b>4</b>	<b>Tooling used</b>	<b>16</b>
4.1	Ruby on Rails with server-side rendering . . . . .	16
4.1.1	Why server-side rendering over a single page application . . . . .	16
4.2	Postgres . . . . .	17
<b>5</b>	<b>MVP of the platform</b>	<b>18</b>
5.1	Architecture of the platform . . . . .	18
5.2	Security . . . . .	19
5.3	Application overview . . . . .	19
<b>6</b>	<b>Validation</b>	<b>26</b>
6.1	Methodology of testing . . . . .	26
6.2	Results of testing . . . . .	26
6.3	Changes made . . . . .	27
6.4	Further developments . . . . .	27

<b>7 Conclusion</b>	<b>29</b>
<b>References</b>	<b>30</b>
<b>Appendix</b>	<b>31</b>
I. Tasks . . . . .	31
II. Licence . . . . .	32

# 1 Introduction

More and more stores are creating online shopfronts for their businesses. However, having an entirely custom-built solution is often feasible only for the biggest chains. For this reason, many smaller stores as well as solo entrepreneurs decide to build their storefront on an e-commerce platform. These platforms abstract away some parts of the process, such as handling payments, creating products, translations and security. Such examples include Shopify [Shoa], WooCommerce [Woob], ShopRoller [Shoc], BigCommerce [Big] and many more.

Building a storefront on an existing platform comes with caveats as each of the more popular platforms has various shortcomings. These range from being difficult to set up and hard to maintain to having bad localization and outrageous pricing. As such, there is a niche to fill by having a platform that prioritizes the functionality and being feature-rich while still being user-friendly and affordable to smaller clients.

The goal of this thesis is to document the building of the minimum viable product of a new e-commerce platform, which aims to provide a good user experience by supporting local languages and being both abstract enough to appeal to less tech-savvy users as well as having the option to customize as much as possible. Due to the time and resource constraints of the thesis, the end product is intended as a market study to see if the predicted niche exists before building an application ready for production environments.

Chapter 2 analyses and describes the pros and cons of popular competing platforms while chapter 3 uses the info to provide a list of functional and non-functional requirements that the planned application must support. Some of these are left as future work due to being out of scope of the thesis and are clearly marked as such.

Chapters 4 describes the tech stack used to create the platform. It also explains the decisions made and the reasons for avoiding a single page application.

Chapter 5 provides an in-depth overview of the final product along with screenshots and a description. It also provides a short overview of the architectural choices made during the development process.

Chapter 6 describes the validation process. This includes the methodology of testing, the results of the testing process as well as the changes made to the final application as a result of the testing.

## **2 Analysis of competing platforms**

The prerequisite for building a high-quality platform is to know what the customer needs and wants. Therefore, it is useful to know the advantages and disadvantages of competitors. This work compares the application being created with the most popular platforms on the market: Shopify, WooCommerce and ShopRoller. In the comparison, the "standard" package or its closest equivalent is generally considered. Such a choice was made so that it would also be possible to compare the prices of services.

### **2.1 Shopify**

Shopify [Shoa] is one of the biggest e-commerce platforms out there. Their platform allows setting up the entire e-commerce flow: they provide tools to do everything from creating a web page and marketing all the way to handling payments and analytics. On the website of the platform, the main limitations of the standard package are the creation of accounts for page administrators and the calculation of taxes. The cost of the standard package is €32 per month. [Shob]

The biggest advantage of Shopify is the easy setup of the platform, because a lot of basic functionality is available, and you can often find help in their add-ons store (see Figure 1) [REF]. This is also one of their disadvantages. Adding additional features is painless, but using them comes with additional fees that can make the initial monthly fee much higher. Figure 1 shows that almost all the most popular applications are free to install or free plan available, i.e. generally their commercial use is paid. It is not within the scope of this work to provide a Shopify-like app store, as this requires third parties to build apps for it.



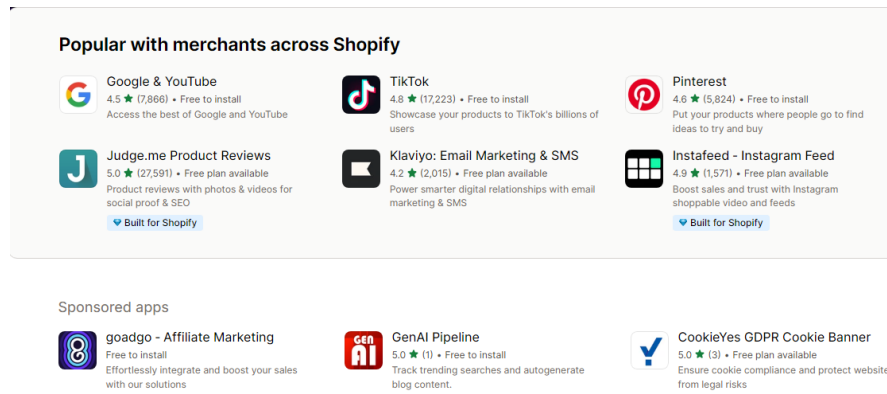


Figure 1. Screenshot of the Shopify App Store.

## 2.2 WooCommerce

WooCommerce or Woo [Woob] is an e-commerce platform based on WordPress. 27% of the million largest e-stores have been built with it [4]. Unlike Shopify, Woo's standard package allows you to create unlimited user accounts and also offers shipment tracking and shipping price calculation. They lack some important functions from the standard package, such as setting minimum and maximum products for orders and email-based marketing. The standard package costs \$39 or 36€ per month. Woo's website also offers the option to exchange currency, but this option did not work correctly for the author of the work.

It is important to note that WooCommerce also offers a free solution [Wooa] that needs to be set up by the customer. Since therefore all security, software updates and other system management that are not directly related to e-commerce are done by the customer, this possibility is not discussed in this work.

## **2.3 ShopRoller**

Shoproller [Shod] is an e-shop platform from Estonia. Unlike other platforms, they do not offer the ability to add unlimited products or add unlimited languages. A package with a price comparable to other platforms, which costs 42€ (35€ without VAT), allows you to add only 2 languages, which is, for example, too few in the context of Estonia. In Estonia, as a rule, there are 3 languages to choose from: Estonian, English and Russian.

ShopRoller also offers services to help set up an e-shop or do marketing [Shoe]. On other platforms, setting up an e-shop is made easy enough that a separate service is not needed. However, offering marketing is a typical additional service that exists on all other platforms as well.

Unlike other platforms, ShopRoller does not charge service fees [Shod]. ShopRoller only charges a monthly fee for using their platform and fees for additional services ordered.

## **2.4 Comparison with planned application**

In general, all competing platforms offer several different options. It is important to consider that it is not possible to include all of them within the scope of this work. Therefore, you have to choose those options that are more important for the customer. Ho and Chuang [HC23] found online payment capabilities, return capability, product add feature, product management feature, product description feature, order management feature, and ease of use to be the most important. Table 1 shows a comparison between different platforms along with the proposed platform.

On the platform created as part of the work, the creation of an add-on store has been omitted, because there is no initial need for it. The main priorities are creating basic functionality for the platform - managing products and orders, making payments, designing the appearance.

The proposed application does not intend to compete with Shopify or Woo, as they have a very strong ecosystem that cannot be built within the scope of work. While the base functionality can be built almost identically, both have a wide variety of add-on modules that offer enough variety to make competition impossible. A realistic competitor is ShopRoller, whose target group is smaller businesses within Estonia.

Table 1. Feature comparison with the planned application.

	<b>Shopify</b>	<b>Woo</b>	<b>ShopRoller</b>	<b>Planned platform</b>
Online payments	X	X	X	Future work
Adding products	X	X	Limited amount	X
Product management	X	X	X	X
Order management	X	X	X	X
Manual orders	X	X	X	
Cart restoration	X			X
User accounts	X	X	X	X
Blog posts	X	Additional module		X
User groups	X	Additional module		X
Add-on store	X	X		
Custom translations	X	X	Limited amount	X

## **3 Application requirements**

### **3.1 Functional requirements (FR)**

#### **FR1: The system must support an account system for admins.**

A store needs to be managed by an authenticated person, which requires an account to ensure privacy. It must not be possible to edit a store without the correct permissions.

#### **FR2: The system should allow purchases to be fully completed online.**

A big part of the convenience of e-commerce is that orders can be placed online. For that, a way to pay should be included. As this means implementing support for a third party payment processor, this is considered out of scope for this thesis and left as future work.

#### **FR3: It must be possible to create, edit and remove products.**

The store owner must be able to create, edit and remove products. These products should be clearly marked and should also include an option to be hidden from the user.

#### **FR4: The system must give an overview of placed orders.**

It is important to have a clear overview of all placed orders in order to know which orders have been handled, completed or failed.

#### **FR5: The system must support restoring a shopping cart.**

The store must allow users to be able to create a shopping cart, leave the website and later restore their shopping cart. It should also provide an option to remind registered users with a filled cart to complete their purchases.

**FR6: The system must support an account system per store.**

The platform should allow users to register at stores in order to take advantage of special deals.

**FR7: The system must allow to create blog posts.**

A big part of search engine optimisation is creating blog posts to draw more visitors to the site. For this, it must be simple to create and show blog posts.

**FR8: The system must allow adding discounts and prices based on the user type.**

It's important to allow some users to have better prices than others, such as first time shoppers having a small discount or customers with a contract getting a special price.

**FR9: The system must support custom translations for content.**

As a big portion of Estonias population speaks primarily Russian, it must be possible to set a locale for the store. For this, it's necessary that product descriptions and names can also be translated to the users language.

**FR10: The system must support custom pages and designs.**

The admin should be able to create their own custom pages with their own designs. It's important that they can use any scripts or styling they wish without interference from the platform.

## **3.2 Non-functional requirements (NFR)**

### **NFR1: The system must look modern and clean.**

This makes the application easier to view as modern design principles are pleasant to look at.

### **NFR2: The system must be easy to use and intuitive.**

Usability is important in a space as competitive as e-commerce. If the system is not intuitive and easy to use, users will have no reason to migrate from other platforms.

## 4 Tooling used

It's important to know what sort of tech stack the application will be built with. This chapter aims to introduce the tools used and provide insight on why these selections were made.

### 4.1 Ruby on Rails with server-side rendering

The website will be built using Ruby on Rails with server-side rendering. This choice was made primarily due to the authors familiarity with the framework as well as the speed that it allows prototyping. Similar applications have also been made using Ruby on Rails before, such as Shopify.

One of the most common myths around Rails is that it's extremely slow, especially compared to more modern languages and frameworks. In reality, this isn't an issue as modern computers are extremely fast and there are plenty of services that are used by millions of people, such as GitHub [Hes], Gitlab [Git] and Shopify [Mü].

#### 4.1.1 Why server-side rendering over a single page application

One of the most popular front-end development methodologies is creating an API-based SPA or single page application. This is intentionally avoided for this application as SPAs tend to have very bad search engine performance due to them being generated dynamically with JavaScript on the users side. This is because search engine crawlers often only run minimal JavaScript and might not be able to parse the complete website. [Car21] By using server-side rendering, this problem is avoided entirely as the HTML document is generated on the server before being sent to the user.

The main downside of server-side rendering is that the load on the server is typically heavier than having a separate single page application. This however is unnoticeable



unless the amount of requests made gets very big, which as mentioned before is out of scope for this thesis.

## **4.2 Postgres**

The database system chosen for this project is Postgres. The choice was made because it's open-source and battle-tested in many other real-world applications. Thanks to these facts, it's expected to have little to no problems with the database and whatever problems arise should be solvable with the massive wealth of documentation available.

Other choices considered briefly were SQLite and Cassandra. SQLite was not chosen as it is not as feature-complete as Postgres. Cassandra was not chosen as Postgres has more support and the author has worked with it more in the past.

Another reason for the adoption of Postgres is that the project uses a database-backed worker handling system instead of something like Redis. With the tools offered by Postgres, it should be just as capable as its Redis-backed counterparts while offering better stability.

## 5 MVP of the platform

This section provides an overview of the completed MVP of the application that was used for the initial validation and testing phase. The application's source code is available at <https://github.com/gCoreByte/starry-skies>.

### 5.1 Architecture of the platform

The application is built around the idea of stores - everything besides the admin accounts are owned by a single store. Figure 2 shows how every record besides the admin account is owned by the store in one way or another. This enables the admin to easily own multiple stores and also has the advantage of being able to get every record associated with a singular store easily.

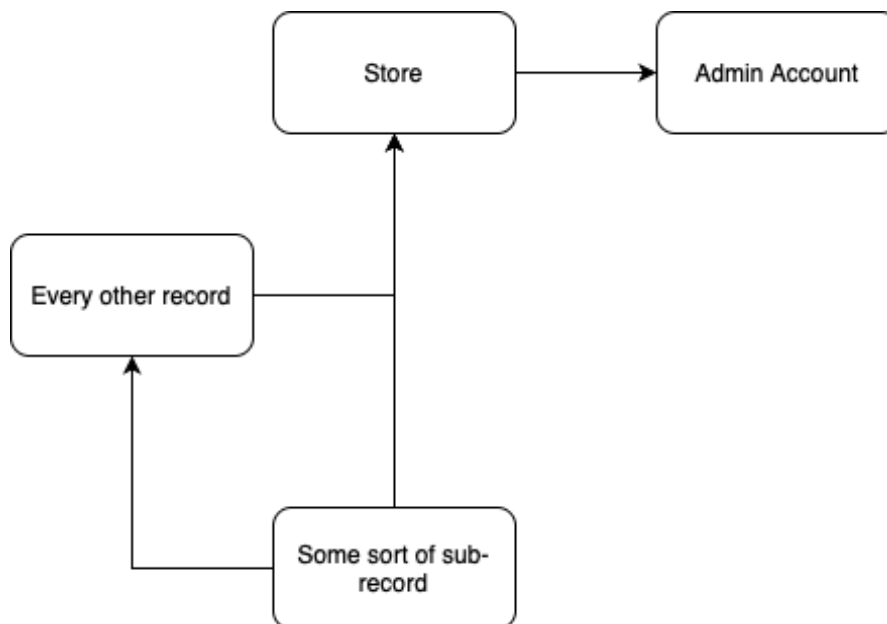


Figure 2. Generalised architecture of the store.

This choice also makes it simpler to delete user data. By having a cascade delete start from the store, it's guaranteed that all user data will be removed. In the event that a

soft-delete is preferred in the future, it's trivial to change the cascade delete to a nullify.

## 5.2 Security

As e-commerce involves handling card numbers and other various transactions, security is extremely important. For the scope of this proof-of-concept application, no real financial data is entered and thus the risk is minimal. For future works when real financial data is handled, it's important to first verify the security of all admin actions.

## 5.3 Application overview

In this section, the various views of the application are described. This includes explaining the various choices made.

### Landing page

This is the first page the user will interact with. It is extremely minimal and only serves as a warning that the platform is not production-ready. For future work, it's one of the highest priorities to fix.

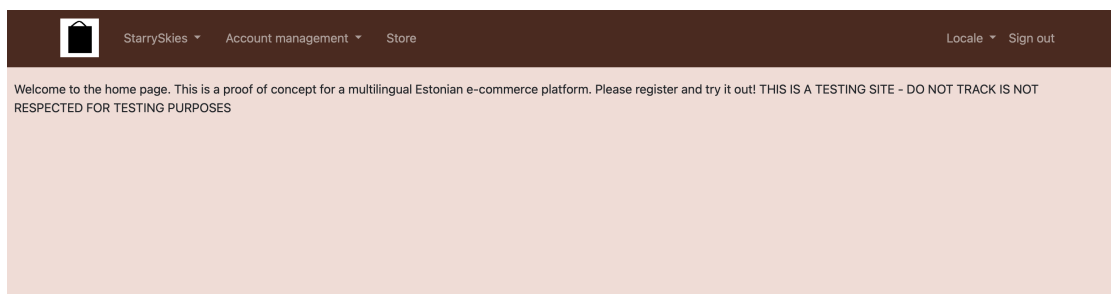
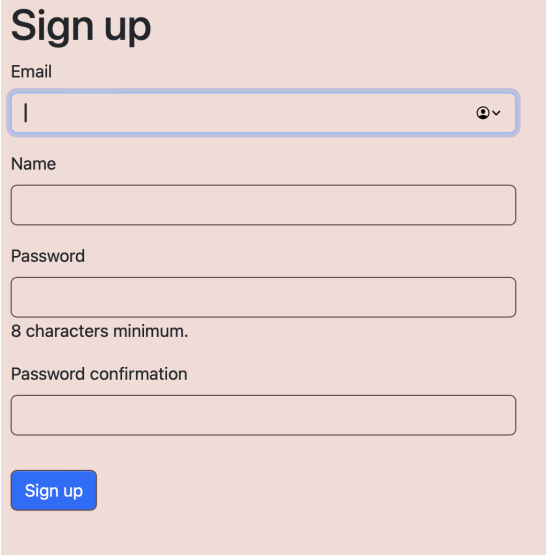


Figure 3. Landing page

## Authentication

In order to create a store, the user must first create an account and then log in. The registration form is shown in figure 4. The password is stored securely in the database and the user is allowed to log in using the form in figure 5.



The 'Sign up' form is displayed on a light pink background. It features a title 'Sign up' in bold black text. Below the title are four input fields: 'Email' (with a blue border and a password icon), 'Name', 'Password' (with a note '8 characters minimum.' below it), and 'Password confirmation'. A blue 'Sign up' button is positioned at the bottom left of the form area.

Figure 4. Sign up form



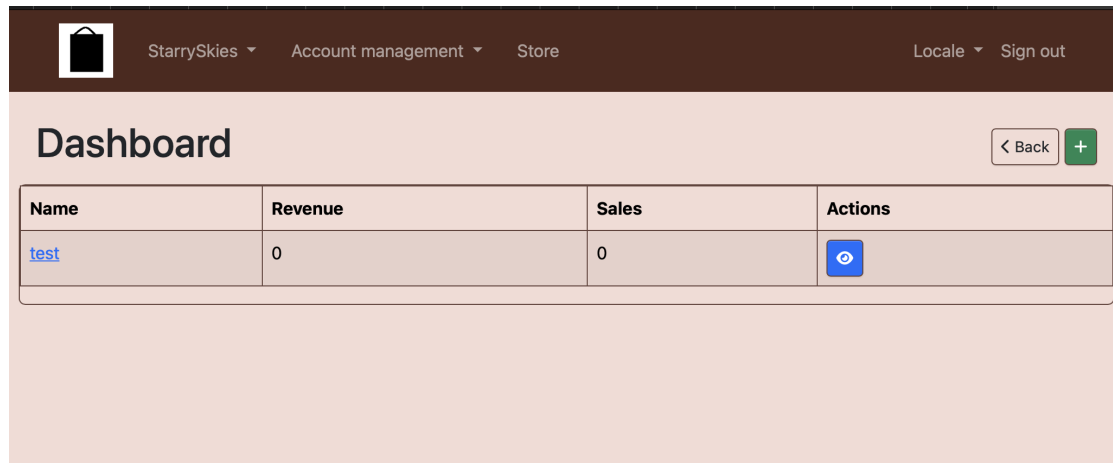
The 'Sign in' form is displayed on a light pink background. It features a title 'Sign in' in bold black text. Below the title are two input fields: 'Email' (with a blue border and a password icon) and 'Password'. At the bottom, there are two blue buttons: 'Sign in' and 'Sign up'.

Figure 5. Sign in form

As authentication is not the primary focus of the application, email verification, changing and password resets were not implemented.

## Dashboard

After logging in, the user is shown a list of all of their stores. This dashboard is shown in figure 6. From this view, the user can see the stores sales and revenue. They can also create new stores or open up a store to edit it.



The screenshot shows a web application dashboard. At the top is a dark brown header bar containing a shopping bag icon, the text 'StarrySkies', and two dropdown menus labeled 'Account management' and 'Store'. On the right side of the header are 'Locale' and 'Sign out' links. Below the header, the main content area has a light pink background. It features the title 'Dashboard' on the left and two buttons, '< Back' and a green '+', on the right. Below the title is a table with four columns: 'Name', 'Revenue', 'Sales', and 'Actions'. The table contains one data row with the values 'test', '0', '0', and a blue circular icon with a white eye. Below the table is a large, empty light pink rectangular area.


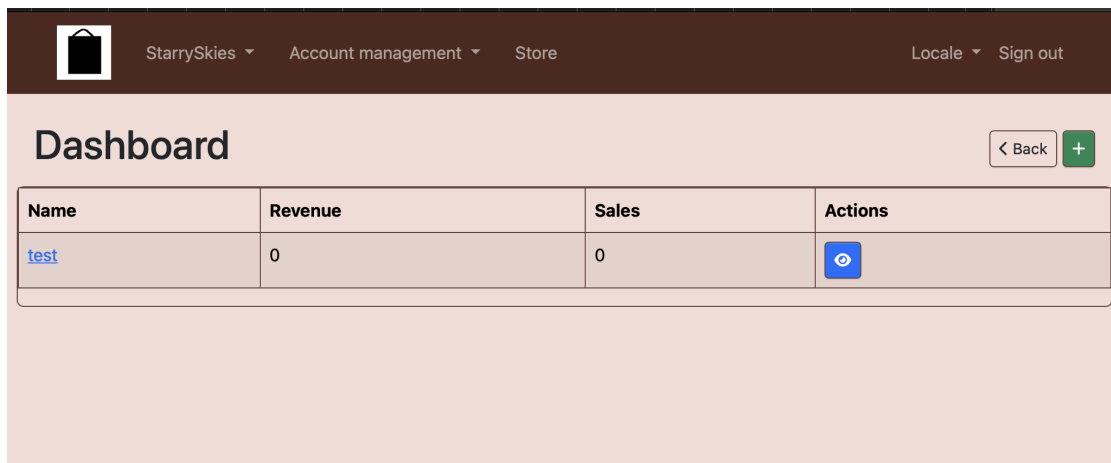
Name	Revenue	Sales	Actions
test	0	0	

Figure 6. Dashboard

## Store view

Upon opening a store, the user is presented with the stores data, a URL to take them to their currently available store and buttons that allow the user to edit various aspects of their store such as the products, pages and blog posts. As this is quite complex, there is also a guide page to introduce the user to the platform and the possibility to create a example store to show various possibilities.



The screenshot shows a web interface for a store named 'StarrySkies'. The top navigation bar includes links for 'Account management' and 'Store', along with 'Locale' and 'Sign out' options. The main section is titled 'Dashboard' and features a table with store data. The table has four columns: 'Name', 'Revenue', 'Sales', and 'Actions'. A single row is visible with the name 'test', revenue of 0, and sales of 0. An 'Actions' button is present in the last column. To the right of the table are 'Back' and '+' buttons.

Name	Revenue	Sales	Actions
<a href="#">test</a>	0	0	



Figure 7. Store view

## Product view

Products are the center of the store - without products, there is no store. Products consist of 2 parts: the base product and then a product version. The product only contains the key, a unique identifier used to keep the products distinct. The product version contains all of the products unique attributes and translations.

The product view can be seen in figure 8 and the corresponding product view in figure 9. On the right side of the product version view are the products categories and prices. These do not have their own special views.

apple


[< Back](#)



Product details

Key	apple
Created by	CoreByte
Updated by	CoreByte
Created at	2024-05-11 22:49

Active version

Active version	<a href="#">Version 1</a>
----------------	---------------------------

Product versions 






Version	Activated at	Deactivated at	Actions
<a href="#">Version 1</a>	2024-05-11 22:49		 


Figure 8. Product view



Product version details






Version	1
Weight	
Height	
Length	
Width	
Activated at	2024-05-11 22:49
Activated by	CoreByte
Deactivated at	
Deactivated by	
Created at	2024-05-11 22:49
Created by	CoreByte
Updated at	2024-05-11 22:49
Updated by	CoreByte

Product prices



Price	User group	Actions
11.34	loyal	
17.06	special	

Product categories 


Name	Created at	Actions
<a href="#">Fruits</a>	2024-05-11 22:49	

Figure 9. Product version view

## Pages

The second most important part of the store are its pages. The page system consists of 2 separate parts, similarly to products. First is the page, which contains the URL that is used to identify the page, key and status. These can be seen in figure 10. The second part of the pages logic are the page templates. These have a based on value, which tells the platform what record to use. They also have translations, which are used to display different languages to users depending on their locale. The page template view can be seen in figure 11.


Page		<a href="#">&lt; Back</a>	
Page details		Page template details 	
Key	index	Key	index
Status	live	Status	active
URL	index	Based on	store
Created by	CoreByte		
Created at	2024-05-11 22:49		

Figure 10. Page view







index		<a href="#">&lt; Back</a>	
Page template   		Page translations 	
Key	index	Locale	Actions
Status	active	en	  
Based on	store	et	  
Created by	CoreByte	ru	  
Created at	2024-05-11 22:49		
Updated by	CoreByte		
Updated at	2024-05-11 22:49		

Figure 11. Page template view

The page translations contain all the actual content for the pages. They are written in HTML and allow using scripts and custom CSS. In addition to HTML, it is also possible



to reference the based on record from the page itself. This allows the user to create a single template for a record and then dynamically fill the content.

## **6 Validation**

This section describes the validation and testing of the created platform. This is necessary to determine whether the platform satisfies the requirements, is comparable to existing platforms and meets the users expectations.

### **6.1 Methodology of testing**

The validation of the platform is conducted by having users attempt to solve some sort of problem or achieve some kind of goal by using the platform. Their progress will be tracked in the system itself by tracking their mouse movements and navigation log. This ensures that there is as little bias as possible and the data is more objective than with user interviews. Users will also be asked to fill out an exit survey in order to get a better idea of possible user experience issues. The tasks are based on real-world issues that might be encountered. They can contain multiple steps and should not take longer than 5 minutes. The data is examined to see if there are any problematic areas for the user, e.g the user cannot find the correct button on a page.

### **6.2 Results of testing**

Most users managed to complete 6/8 of the tasks given. The very first testers discovered various bugs in the system. These included the store subdomain saving with capital letters, which meant that the store was inaccessible and store orders not loading. These were fixed hours after being reported to get accurate results about the platform.

Of the 8 tasks given, the biggest challenges were tasks 7 and 8. For task 7, only 1 tester managed to complete it unassisted using only the guide. For task 8, only 2 testers managed to complete it. The other tasks did not pose a challenge to a majority of the users.

For task 7, the biggest complaint was the inability to understand how page translations work and how to access them in the store. To solve this, the guide should be improved. One user also suggested creating a video guide or a step-by-step tutorial. Both of these would help solve the problem, but would not address the base issue of the system being unintuitive. Solving this is left as future work as it would require a new design.

For task 8, the main complaint was the inability to understand the task. When the task was explained to them, most users managed to complete the task unassisted. As such, this task is counted as completed.

As feedback was optional, not all users gave some. Of the users that did give feedback, around 70% disliked the design of the page, but were happy with the features offered. This was echoed from their test results, where users were able to solve the tasks but often had to refer to the guide or search the page. This echoes the sentiment given for task 7 and solving this is left as future work.

### **6.3 Changes made**

As a result of the validation, not many changes were made. Some translations were improved and bugs were fixed, but improving the design would be a refactor that goes beyond the scope of this thesis. Once the design is improved, it has the potential to be a competitive entity in the Estonian e-commerce market.

### **6.4 Further developments**

The main priority for further development is improving the front-end design. The design is not optimal and users do not enjoy using the software purely because of it. It's likely that this would require a dedicated web designer to avoid common pitfalls and follow modern principles.

The second priority for future developments would be implementing a proper payment solution. The most likely choice would be Stripe or a similar solution, which offer a payment solution for platforms.

For other developments, the author would prioritise a what-you-see-is-what-you-get editor for websites. This would drastically lower the skill floor for people, who are not familiar with HTML.

## 7 Conclusion

As there is a shortage of e-commerce platforms that support Estonia, this thesis aimed to create a proof-of-concept solution that would support everything a user needs. It succeeded in developing a proof-of-concept platform, which allows users to build their own online store.

Before the development of the platform, research was done into the main competitors on the market. This research showed which features users mostly want and need based on their presence in other platforms. The final application was built to fill the users needs based on this research and incorporates many features that competitors on the Estonian market lack such as blog posts and user groups.

The main competitor of the finished platform in Estonia would be ShopRoller. While developed platform has more features, it is significantly less mature on the user experience front. During testing and validation of the platform, many users noted that they are impressed with the feature-richness of the developed platform, but complained that the initial learning curve is steep, even with the provided guides. They also reported that they would likely use the platform over its Estonian competitors if it was more user-friendly.

For future work, the highest priority should be improving the general appearance of the platform and making navigation easier. This would make the platform significantly more appealing to new users and solve the biggest issues. Secondly, a proper payment processor should be introduced to make the platform production-ready.

## References

- [Big] BigCommerce. Bigcommerce homepage.
- [Car21] Daniel Cartland. Common single page application (spa) crawling issues & how to fix them. 4 2021.
- [Git] Gitlab. <https://about.gitlab.com/blog/2018/10/29/why-we-use-rails-to-build-gitlab/>.
- [HC23] Shu-Chun Ho and Wei-Li Chuang. Identifying and prioritizing the critical quality attributes for business-to-business cross-border electronic commerce platforms. *Electronic Commerce Research and Applications*, 58:101239, 3 2023.
- [Hes] Adam Hess. Building github with ruby and rails.
- [Mü] Philip Müller. Under deconstruction: The state of shopify’s monolith.
- [Shoa] Shopify. Shopify homepage.
- [Shob] Shopify. Shopify pricing.
- [Shoc] ShopRoller. Shoproller homepage.
- [Shod] ShopRoller. Shoproller pricing.
- [Shoe] ShopRoller. Shoproller services.
- [Wooa] WooCommerce. Open source ecommerce platform for wordpress.
- [Woob] WooCommerce. Woocommerce homepage.

# Appendix

## I. Tasks

### Testing

Welcome to the test group for my thesis. Below is a list of tasks that you should accomplish. Please do your best to complete them all if possible. Any issues, impossible or difficult tasks etc should be reported to me at martinhans.keskkula@gmail.com. Participation is voluntary and no identifiable information is collected. Please note that adblockers MUST be turned off while participating in the tests in order to allow the necessary data to be collected. Thank you!

To begin, please continue to <https://corebyte.ee>.

### Tasks

1. Create an account. You can choose the attributes freely.
2. Log in to the account you just created.
3. Create a store with a subdomain and name of your choosing. For simplicity, keep the example store checkbox ticked.
4. Create a product of your choosing. It should be activated.
5. Create a blog post.
6. Add prices to a product that differ based on the users attributes.
7. Create a page that has different content based on the users language.
8. Add some products to your cart and restart your browser. Your items should still be in the cart.

## **II. Licence**

### **Non-exclusive licence to reproduce thesis and make thesis public**

**I, Martin Hans Keskküla,**

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

#### **Developing an E-commerce Platform for the Estonian Market,**

supervised by Mohamad Gharib.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.
3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.
4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Martin Hans Keskküla

***dd/mm/yyyy***