Marketplace for Circular Bioeconomy

Beka kokhodze

a52257

Project work presented to Escola Superior de Tecnologia e Gestão de Bragança to obtain the Master's Degree in Information Systems under the double diploma program with the Tbilisi State University.

Work carried out under the guidance of:

Professor Maria João Tinoco Varanda Pereira

Professor José Eduardo Moreira Fernandes

Professor Manana Khachidze

Bragança

2022

Dedication

The successful publishing of this project was achieved by many people who were helping me and pushing me to do my best.

I want to thank all of my professors without who I wouldn’t be able to finish this project. Thank all of you for your hard work and dedication to this success.

Resumo

Impactos ambientais, poluição e outros tópicos relacionados ao meio ambiente tornaram-se altamente relevantes no mundo de hoje. Com milhares de fábricas produzindo o máximo possível e durante o processo criando subprodutos que tem como destino final os aterros sanitários gerando um grande impacto ambiental. Diversos autores e empresas já estudam a possibilidade de encontrar melhores maneiras de reciclar e reutilizar esses subprodutos, a fim de reduzir o impacto em nosso ecossistema. Como alternativa, foi realizada a criação de um website que tem a função principal de oferecer a empresas ou particulares que anunciem seus subprodutos onde talvez alguém tenha interesse em usar este produto para outros fins para que não seja despejado em aterros. Para a criação do website tecnologias foram utilizadas, como React Next.js e Firebase Firestore.

Palavras-chave: Firebase, ReactJs, Next.js, Typescript, No-SQL, Firestore, Bioeconomia Circular.

Abstract

Environmental impacts, pollution and other topics related to environment became highly relevant nowadays. With several manufactures producing the maximum possible and creating byproducts that have landfills as its final destination, generating a huge impact on nature. Various authors and companies already research possibilities to reuse and recycle byproducts, in order to reduce the disposal of harmful components in the ecosystem. As an alternative, a website was developed with a goal to allow companies and individuals to offer their byproducts for those who might be interested and that could use it for alternative purposes and not to be discarded in landfills. For the website creation were used technologies as React Next.js e Firebase Firestore.

Keywords: Firebase, ReactJs, Next.js, Typescript, No-SQL, Firestore, Circular Bioeconomy, Environment, E-commerce, pollutions, byproducts.

General Index

List of Acronyms/Abbreviations

EBN - European Bioeconomy Network.

CRUD - Create, Read, Update, Delete.

ES5 - JavaScript ECMAScript 5.

TWB - Two-way binding.

DI - Dependency Injection.

CLI - command-line interface.

Virtual Dom - Virtual Document Object Model.

JSX - JavaScript XML.

DOM - Document Object Model.

MVC - Model View Component.

CLR - Common Language Runtime.

CI/CD - Continues integration and Continues delivery.

DevOps - Development Operations.

SSG - Server-side generation.

SSR - Server-side render.

SPA - Single page application.

JWT - JSON Web Token.

PR - Pull Request.

Index of Figures

Table Index

Chapter 1 Introduction

From the start of the Industrial revolution all around the world, the main idea of producing and creating new products was to produce as much as possible, as many as possible and as fast as possible, there was no thought on what impact it could have on the environment. The ecosystem was very different for example a century ago, but the rapid development of human beings changed totally.

A couple of decades ago countries realized that it was harmful to sustaining life on earth because of air pollution, In oceans, tons of plastic waste products created artificial islands, tones of animals died and etc. list is long. Usually, we people have this problem that we don't see until it is too late, the process is started to recover and manage waste and production much smarter, with more care for the ecosystem, progress is huge but not enough. So, we need to find better and newer ways to manage waste products for empowering bioeconomy. Therefore the main objective of these work was create platform that allowed companies create announcements about their byproducts of their processes and help other companies to connect them with each other to stimulate bioeconomy.

1.1 State of Art

Helping the environment and Circular bioeconomy are not an innovative statement, nowadays there are many companies trying to achieve the same goal in different ways. One of them is Continued Fashion which received a grant from European

Union and is oriented toward recircling clothes because clothes are essential product and material that is used for this creation can be recycled easily.

Bioma will be one example of the methods to stimulate our Circular Bioeconomy using byproducts created from any production line, we can see other similar projects like Bioma on European Bioeconomy Network's (EBN) website (https://eubionet.eu), the most interesting is AgriMax, oriented on food waste that is being created and getting spoiled, they are trying recycle these products and create something new.

The problem of almost all of the platforms is that it is closed minded, they are oriented on only one aspect, and it isn't available for all people, they can't post, order, see and filter products easily. Because of that, Bioma will try to purify and simplify the relationship between companies or individuals.

1.2 The problem statements

One of the biggest struggles that companies face is recycling byproducts. Most of the time their final location is a landfill, or they are burned and it has a massive impact on air pollution. The main reason for this is companies don't want to spend extra money on nature and they are pointed to survive at first and after that to have a high income. with that idea, there is a way to create a marketplace where they will be able to place their byproducts, so-called for us - products.

The main idea is that for one company that is a byproduct maybe it is material for another company for manufacturing products. so that there will be three-win positions:

1. Reduction of pollution in our ecosystem;

2. Seller companies will get an efficient disposal of unneeded byproducts;

3. Buyer company will receive the materials at a much cheaper price.

And in order to make things easier, the buyer need to know the exact location, price and type of byproducts, speeding up the entire process.

1.3 Solution

The easiest and most comfortable is to create an online platform, in this case, a website, where companies will be able to upload their products/byproducts/scraps, they will control waste products, in addition to helping our ecosystem. In the other hand, buyers will get necessary materials to manufacture things they want.

Chapter 2 Requirements analysis and specification

2.1 Introduction

This chapter presents all the functional and non-functional requirements needed to develop the application. It presents use case diagrams and user stories. The chapter ends by detailing the site maps for the different users, the class diagram and the database schema.

2.2 Functional specifications

Platform needs to be an e-commerce like platform where users will be able to see products, do some Create, Read, Update, Delete (CRUD) on the products that were created by them, create orders and see notifications about the platform updates or personalized notifications.

There are possibilities of: see all products and attach some filters on them, being super friendly for users, they can easily locate product that they want and desired location of it. After choosing one product they will be able to see product's detailed information and even give rating so that other users can use it.

The Platform will have several types of users, this chapter will present users with a full description of their responsibilities and functionalities that they will have.

2.2.1. Guest user

When user is not yet registered, there will be only with access to public information. The access to using the capabilities of the platform, marketplace, etc. will not be available, but users will be able to add items to the cart and start order flow, in order to finalize it they need to log in or create an account to complete the order. It is needed because this way we will have two-way communication with clients and we can give updates to them.

2.2.2. User

User has already created an account and now she/he can place orders, add a primary address, and add new products to sell or for demand, in short, this is the user on which platform needs to work, these users will create entire e-commerce. This type of user can place two types of products on our website, it can be sell or buy.

2.2.3. Moderator

Has control over the content present on the platform, which he/she can change or remove, as well as insert latest content. They can confirm and review orders, but no cancelling or final approval.

2.2.4. Administrator

Has full control over the system and handles its maintenance, including adding moderator users and managing users. This user can confirm and cancel orders. Administrators count shouldn’t be high, max 4, after it, communication gets hard on prominent level

2.3 Non-functional specifications

Even though non-functional requirements don’t have any influence on the main functionalities, it still plays a vital role at the performance and how easy to support the platform is. Website will be multilingual so that all users around the world can use it.

2.3.1. Extensibility

One of the most crucial details in any application, is it ensures how easy it will be to add new functionalities to the platform without creating problems, because the main thing is to follow its rules and best practices, all of these are followed in current codebase.

2.3.2. Scalability

When the platform is getting bigger and more popular, at the same time it means an increased load on the servers that can cause slowing down. Thanks to the systems that we have here it isn't a big problem; it was designed to be scalable.

2.3.3. Ease of use

Graphical interface is important when we have communication with people who we don't know about their background, so we need to try to create the easiest to use interface so that everyone will be able to use it.

2.4 Use case diagrams

Use cases are an important part for designing any complex system like this one. An use case is a description of how a person who actually uses that system will carry out a goal. It helps to define how each user actor will and should interact with.

Figure 1 presents the general use case diagram. All the different actors of the system are present in this diagram (Guest User, User, Moderator, Administrator) and in the following parts I will explain in detail every use case per actor.

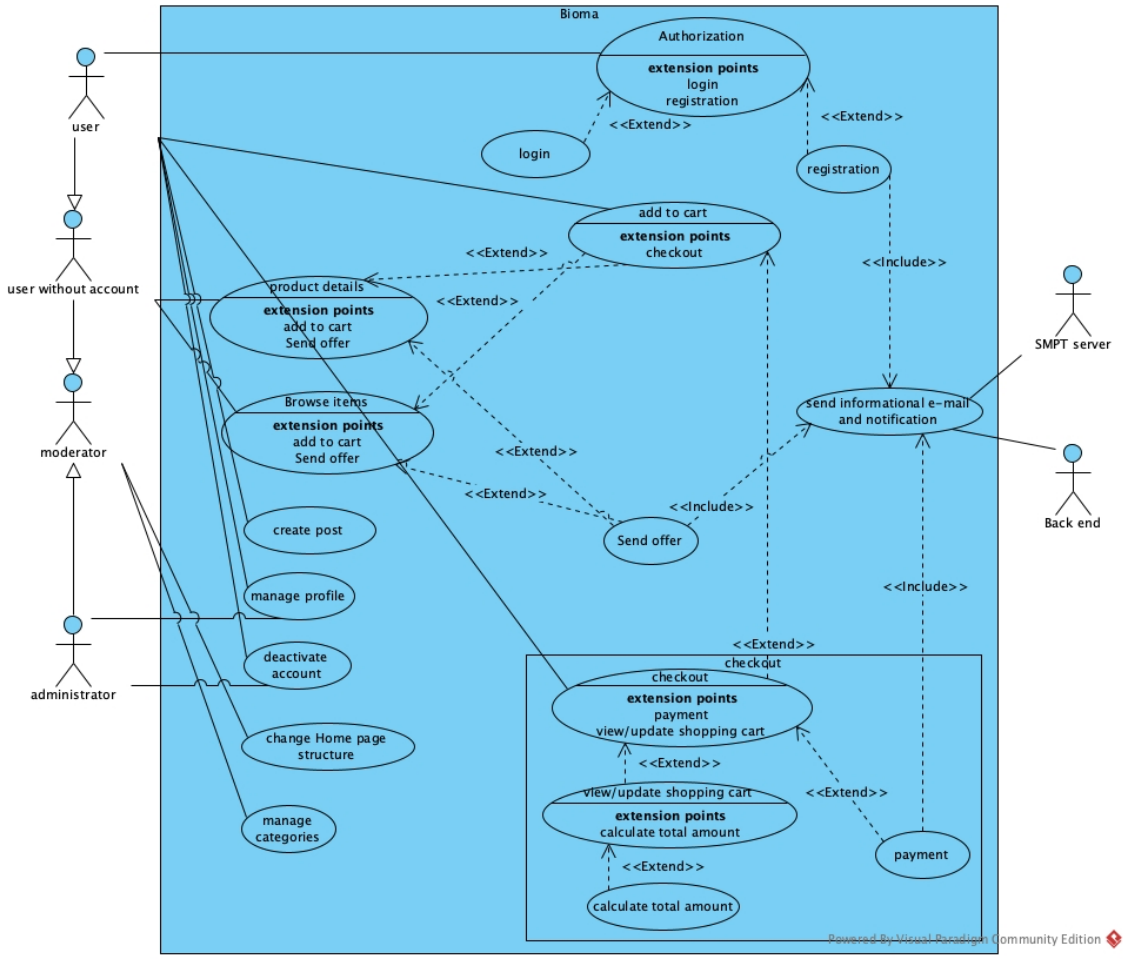


Figure 1: General use case diagram Actors' identification

This part presents the main actors who will play a significant role in creating the entire ecosystem on the platform.

2.4.1.1 Guest users

Are an important player in E-commerce, we need to try to develop the retention rate of that kind of user. They will create awareness in the community. Guest users will have the ability to browse in the platform and see product details. Users will be able to add products to the cart and after login/onboarding, their cart will be still there. If they want to order a product, to do a checkout, they need to log in/create an account. No user should be able to order products without an account, focusing in targeting and communication. They will be able to add a product to the wish list too which is similar to the cart but it isn't saved for future order, user need to add that product to the cart after it.

2.4.1.2 User

The Source of these users are Guest Users; this is one of the main reasons why we need to be oriented toward those users. They are the main engine of the platform; they will create orders and they will keep the website alive. We need to try for sure to increase the retention rate of that kind of user. We can call Loyal customers too; at some point in future, they will become one if we have a stable platform and high usability. They will be the ones who can create order and see historical data.

2.4.1.3 Moderator

When there are many people somewhere, they need to be controlled and guided. Moderators will play a huge role in persisting stability of the platform, there will be many harmful users who need to be filtered and checked. Moderators will complete all orders if there are any problems. They are the ones who have communications with customers, they can manage user's profile and edit product details. They can see websites as normal user, but they can't create orders. Deactivating users accounts is their responsibility

2.4.1.4 Administrator

These users are people who fix problems that can't be fixed by Moderators, these users will have permissions of everything. One of their main responsibilities is to handle Moderators. Administrators inherit all the permission from moderators.

2.5 Site Map

To provide an easy tracking process, the following sections will develop a site map for every actor contributing to the application.

2.5.1. Site Map for Guest User

This is sitemap for Guest user, first page where users will get on our website will be home page, and they will be able to see these main pages.

Figure 2: Representation of site map for guest user.

2.5.2. Site Map for User

After a successful login, user will have additional available functionalities, one of them is profile page and checkout.

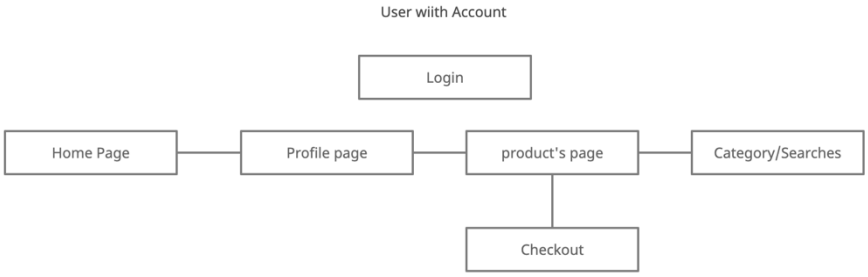


Figure 3: Representation of site map for user.

2.5.3. Site Map for Moderator/Administrator

When administrator/moderator logs in they can see websites main panel, so called user's side, and admin panel, last one is place where they will spend most of the time. In addition, they will have access to all the main control pages that can create the user's side website.

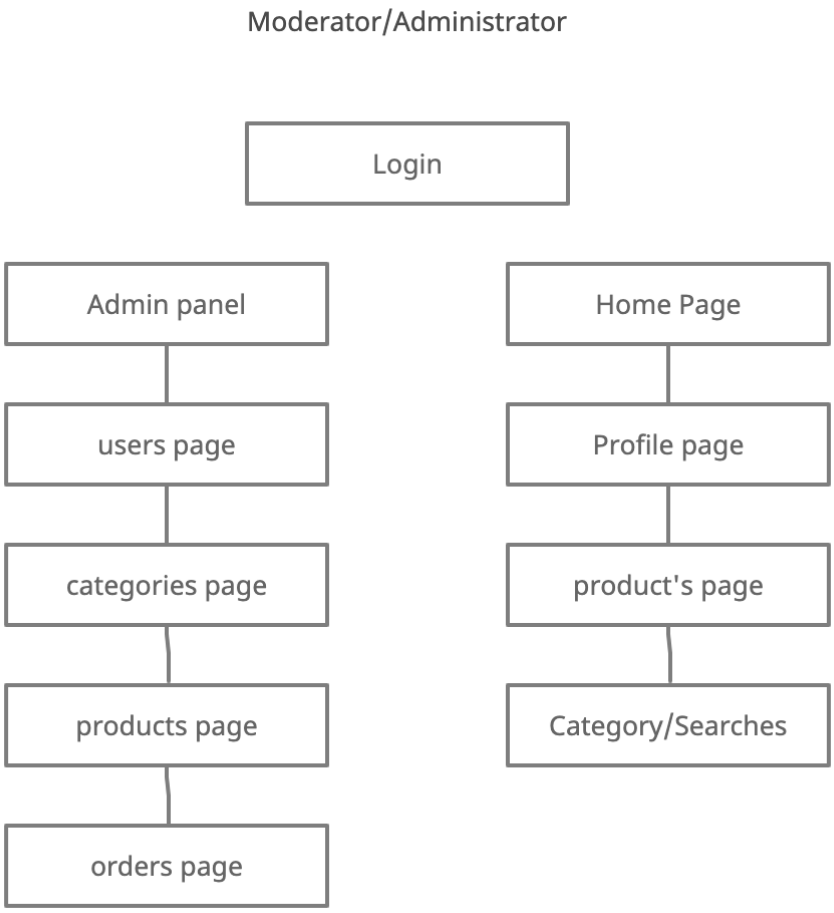


Figure 4: Representation of site map for moderator/administrator.

2.6 Class diagram

The class diagram of the web application presented by Figure 5, describes the platform's diagram which gives us a clear view of the application and helps us to analyze and describe the responsibilities of the system, the structure of the platform by showing their attributes, system's classes, and the relationships between classes.

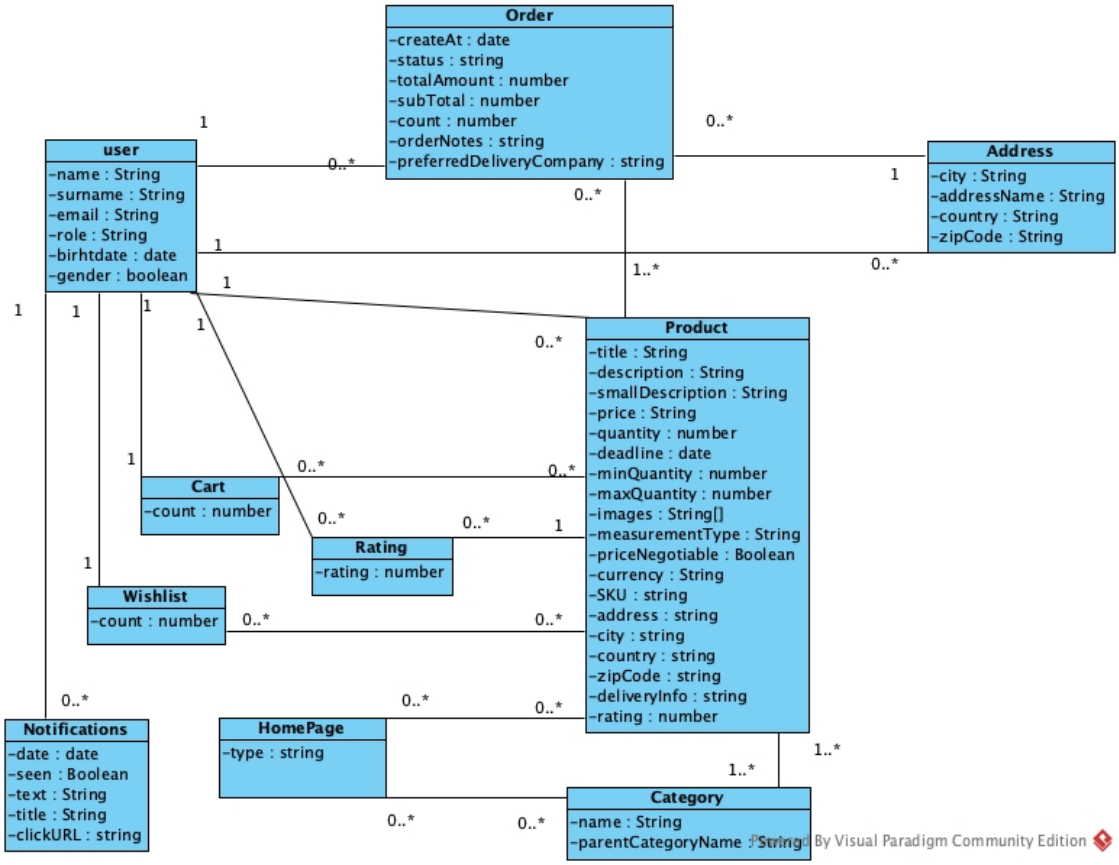


Figure 5: Class diagram of database.

2.7 Database Entity Relationship Diagram

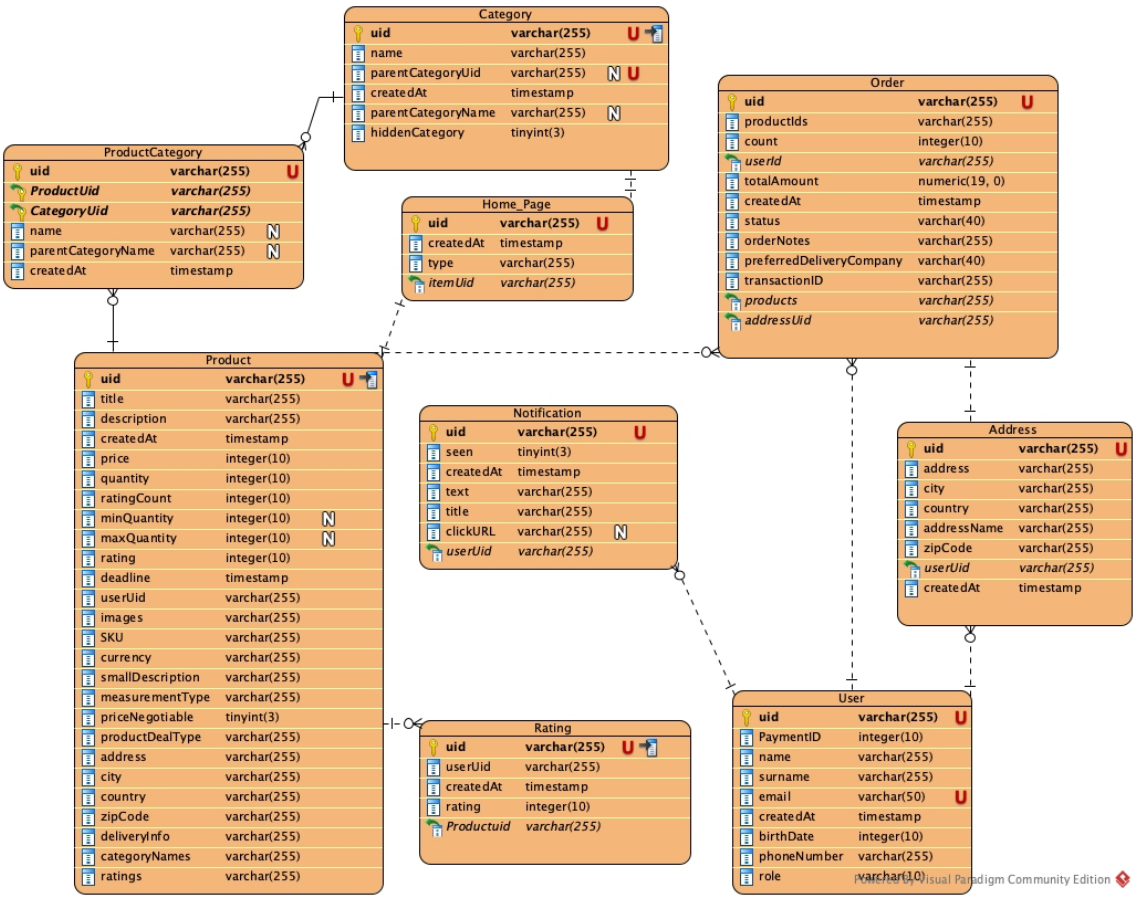


Figure 6: BE database entity relationship diagram.

Database is built with Firebase's no-SQL database where we have ability to put documents inside other documents so that it will become another collection and we can do queries on it. This is mostly profitable for example in ratings system, we need to create new document where we need to add product identifier so that we can identify on what product rating was given, but now we can do it easily inside product document itself, please see image bellow it will help to imagine what it means.

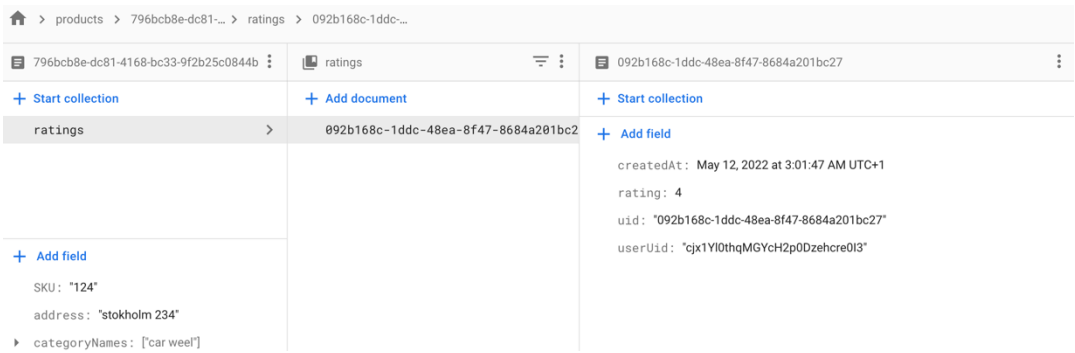


Figure 7: Image for showing how nested collections look.

We need to have many to many connections between categories and products, product can have many categories and categories can have many products, in no-SQL database which is default database language of Firebase, it is possible to achieve same relationship with using one additional collection which is *productCategory.*

For manipulating on home page, we need to create new collection where we will let users to add or remove specific product or category, which gives us ability to have dynamic home page structure.

2.8 Conclusion

This chapter has named all the system's actors and the functionalities that the software should offer. The next chapter will describe tools and technologies that will be used to develop this platform and make it as efficient and fast as possible.

Chapter 3 Technologies and development tools

The main topic of this chapter is tools that could be used for building platforms like this and to show their pros and cons compared to each other. At first, I will start talking with top Front-end frameworks according to GitHub stars, StackOverflow scores and google Trends. Next one is Back-end technologies and libraries that can support platform's functionality.

3.1 Front-End Frameworks

There are many front-end frameworks and libraries, but there are couple of them that are on top of them, one of the ways how we can sort them in this case according to popularity GitHub starts and Google Trends.

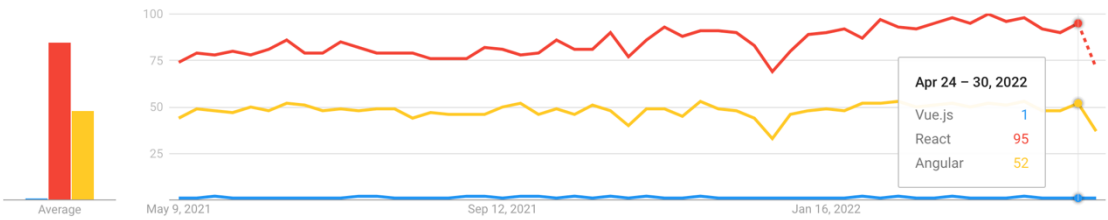


Figure 8: Popularity of frameworks according to Google Trends.

Source: https://trends.google.com/trends/explore?cat=31&q=Vue.js,React,Angular.

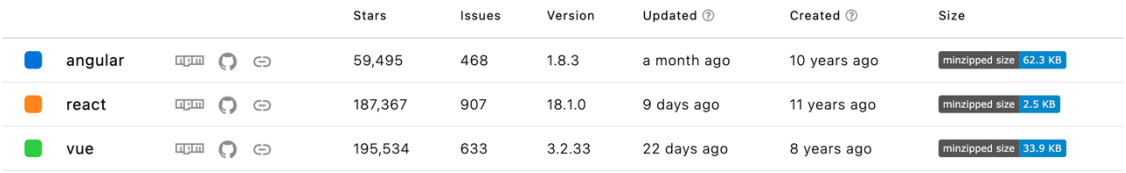


Figure 9: Overall comparison of frameworks according to: Stars, Issues, Version, Updated, Created and Size.

Source: https://www.npmtrends.com/@angular/core-vs-vue-vs-react

From the above images, is possible to see that nowadays there are two big frameworks fighting with each other and for this platform, the best one needs to be chosen from these frameworks.

3.1.1. Angular 1 and 2

Angular is a famous JavaScript ECMAScript 5 (ES5) based open-source front-end web application framework and developed by Google in 2010 (Wikipedia, AngularJS). Angular's first development intention was to help web designers in creating a persistent web form with more efficiency in time and in resources. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application components clearly and succinctly.

Angular 1's core concept was two-way data binding in web browsers reducing the backends' data processing responsibility in web servers. In other words, Two-way binding (TWB) is a concept that means Angular will update your Model automatically, or as necessary, when users input data into your forms. This tight-coupling between input values and their corresponding variable representation in the Model makes handling data much easier—without having to watch for user-driven events (Dayley, 2014).

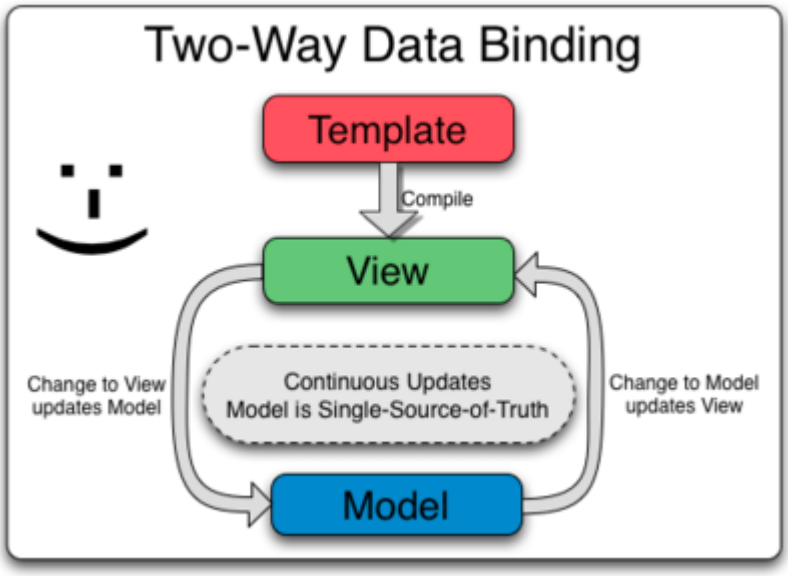


Figure 10: Angular two-way data binding flow.

Source: https://medium.com/front-end-weekly/what-is-2-way-data-binding-44dd8082e48e

AngularJS (called Angular 1) has many downsides: (1) Dynamic applications didn't always perform that well with Angular. One of the problems was also the size of the JS bundle that was loading in browser, because it was SPA (single-page application, in easy terms there was no reloading on URL change, everything page navigation was fast) all pages needed to load in one go. (2) As AngularJS is a versatile instrument, there is always more than one way to complete any task. This has produced some confusion among engineers. (3) Lack of listeners count, there is a limitation on how many listeners you can instantiate. (4) As two-way data binding it is slowing down the app, for updating it is checking input values two times and it has frame drops after that. (5) lack of support for mobile devices.

Angular 1 was the first framework that gave developers flexibility to create modular UI with ease, but because of the initial design problems for the second version Google developers team rewrote core code entirely. The design problems were logical, because it was the first of its kind and it was impossible to create the perfect framework right away.

The list of changes is below (Kumar, 2017):

- Angular 1.x Controllers are gone in Angular v2. We can say that Controllers are replaced with "Components" in Angular 2.

- Angular 1's core concept was $scope but you will not find $scope in Angular 2.0 and above.
- Updated version added event binding as one-way binding from view to component which was non-existent in Angular 1.
- It becomes TypeScript base language from JavaScript base language, which is a strict syntactical superset of JavaScript developed by Microsoft.
- Angular 2 uses Hierarchical Dependency Injection system which is the major performance booster of it.
- One of the biggest advantages of the new Angular is Dependency Injection. In Angular 2, Dependency Injection (DI) is there but now there is a unique way to inject dependencies. As everything is a class in Angular, DI is achieved via a constructor.

The list is ongoing because it was a major rewrite.

Consequently, AngularJS (Angular 1) isn't a suitable candidate to be used in this project, and Angular 2 has superior performance and developer friendly design pattern which is perfect for big enterprise companies. It supports Server-side rendering which is needed for SEO optimization.

On the other hand, one of the biggest problems of angular 2 is verbose and complex, it has steep learning curve and information about command-line interface (CLI) is small in their documentation. One small example of it is a situation when you want to create component, probably you will end up with 5 separate files to inject them for one component.

3.1.2. React

React is a JavaScript, open-source library created by a collaboration of Facebook and Instagram, it was initially released in 2013. Its aim is to allow developers to create fast user interfaces easily. The current stable version is 18.X, released in October 2022. React makes no assumptions about the rest of the technology stack used, thus it's easy to try it out on a small feature in an existing project.

It's currently adopted in production by big companies like Facebook, Instagram, Yahoo!, Airbnb, and Sony. Its popularity keeps growing and more and more companies will adopt React as their main library.

Due to React's superior features and performance, Meta released it as a library, it only had core logic, which means, how state of the component should be handled, they introduce Virtual Document Object Model (Virtual Dom) for the first time in web development history, it gave superior boost to its performance.

React isn't a Framework. There is much confusion about it, some developers think that React is Framework. But, React itself is core handler of state and binding to elements. For rendering something on website you need to add another package *react-dom*(developed by Meta), which gives the ability to manipulate with Real Dom. though it also gives developers or companies to use react on different platforms, one of the example is React-Native, mobile app development framework. which is based on React and uses JSI and fabric for communication with native view elements. It doesn't offer all the components you'll find in projects like Ember or AngularJS. In fact, many refer to React as just the V in MVC. It comes from JavaScript XML (JSX) that gives developers the ability to write everything in one JS file.

The logic of relationship between Virtual DOM and Real DOM isn't super complex, we can easily describe how React updates state and keeps binding to elements in dom. The virtual DOM is a tree based on JavaScript objects created with React that mimics a DOM tree. Every time you want to change something in the DOM, React employs a diff algorithm that only re-renders the DOM nodes that have changed. This algorithm is used for efficient re-rendering because DOM operations are typically slow, at least compared to executing JavaScript statements. In this way, React can be very fast, which is especially important considering that the CPU of mobile devices are far less powerful than desktop devices.

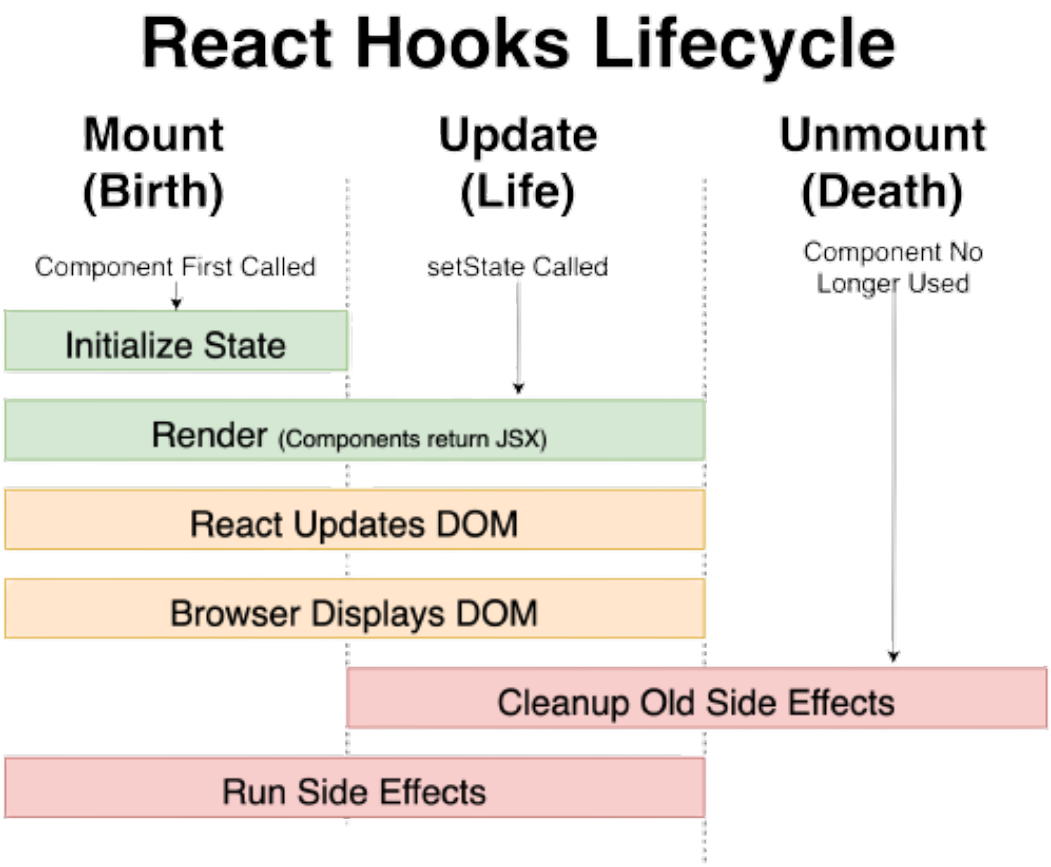


Figure 11: React's component lifecycle.

Source: https://blog.bhanuteja.dev/the-lifecycle-of-react-hooks-component

Lately React introduces new way of creating components which called hooks (First in class), which recovered and revamped popularity so that other libraries started creating hooks (Flutter, Vuejs).

Nowadays React is at its maximum, stability and its comprehensive API gives all companies speed and dynamic codebase that has influence on how good the product is, in this exact case website. By using different tools, we can add SEO support. It is a suitable candidate to be used on our platform.

One framework that is based on React is NextJS which gives us the ability to create SPA with Server-side rendering. For me, it is the ultimate tool to use on building every website, support of SSR gives us ability to optimize SEO which is important for ecommerce websites. Another wonderful thing that comes with NextJS is its speed, every page is lazily loaded on the user's machine, which means that pages are loaded on demand, on user's request. In other words, on the building process of project, JS bundles are separated per page so there is no necessity to load every pages' code at once.

3.1.3. Vue

Vue.js is a lightweight JavaScript library, created by Evan You. Before and during the creation of Vue, Evan worked for Google. While he was at Google, he was always concerned how heavy and not flexible Angular was and he had everyday connection with it every day because it is Google's main Front-end framework, one day he decided to create new library which would be lightweight and would have good features and design patterns of other Fron-end libraries. On 2014 he shared his work with teammates and others on GitHub and Vue's journey starts from here. Although Evan considers Vue to be more like React, as its core idea is one way data binding and components, as in React. Vue was created with simplicity and user experience in mind, which means that if user knows the basics: HTML, JavaScript and CSS it would be super easy to publish websites(Youtube, Vue.js: The Documentary). Vue as React they are based on Virtual DOM, which makes their performance close to each other, so in the end decisions need to be made according to developer experience.

First stable version of Vuejs(Vue 2), which become super popular because of its simplicity. The community got big quickly and companies started implementing it in their products (Alibaba, BMW...). But there were still some problems, lack of support of Typescript was game changes for some companies (also for me too). In their next release which was named Vue 3, they improved and added new features. One of the biggest ones were (1) they changed diff algorithms that were deciding when user state needed to be updated. (2) they added better support for TypeScript and SSR (Server-Side Rendering).

Vue's rendering and mounting aren't far different from React, they share almost same methods;

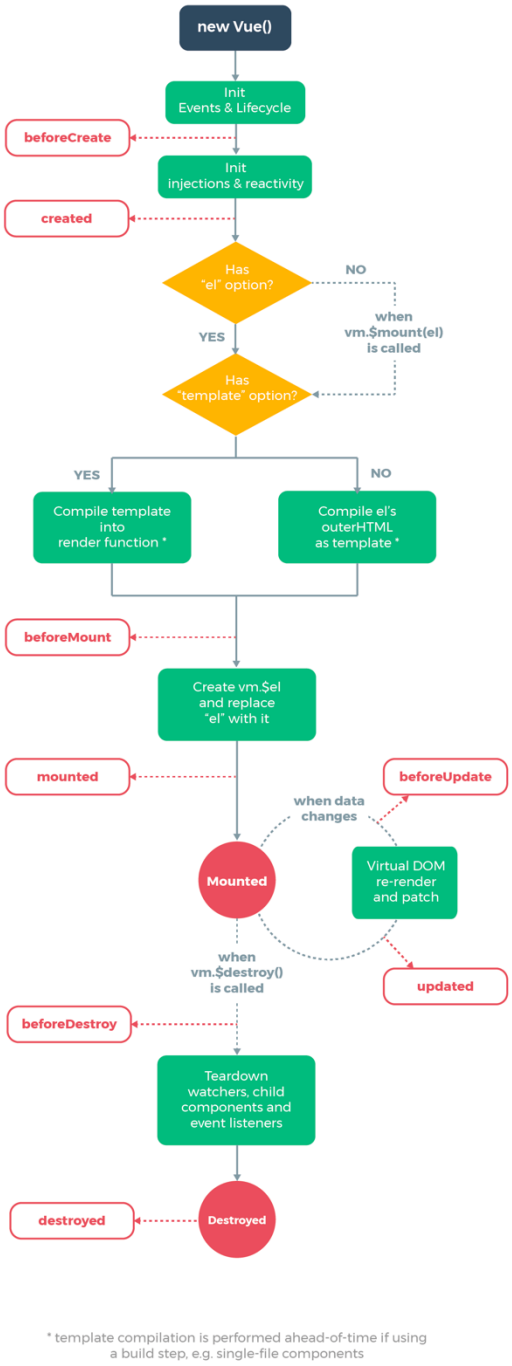


Figure 12: Vue.js's component lifecycle.

Source: https://www.digitalocean.com/community/tutorials/vuejs-component-lifecycle

For companies who are searching for stability and no extra headache maybe Vue isn't a good option, it doesn't have a big company on its back, compared to React and Angular they are supported by Meta and Google. They are supported by small companies and mostly by the community. It is opinionated like Angular and sometimes has not flexible syntax.

3.2 Back-end Frameworks

It is hard to decide which Back-end language is the best, they need to be fit per project, differences are wide sometimes and hard to compare, but one of the workable way is to compare GitHub starts and Stackoverflow scores. This way we can have an overall look at the framework that is popular in community and most of the time community has huge power on framework's ecosystem.

	Express (NodeJS)	Laravel	Firebase	ASP.NET Core
Stars	56.9k	69.6K	4.7k	28.1k

Table 1: Stars count according to GitHub.

3.2.1. Express (NodeJS)

Express is an open-source framework created in November 2010; it is one of the best framework created on Nodejs. Nodejs is an asynchronous event-driven JavaScript runtime, it is designed to build scalable network applications. This framework is not very opinionated in its nature and is used by many developers and companies worldwide. It helps with creating web and mobile applications back-ends with a minimalist approach. It is an important part of the MERN Stack development. The MERN stack is a software bundle used in web development and it consists of MongoDB, Express, React and Node.js. When using MERN stack the whole application, client-side and server-side, and is written with JavaScript or Typescript. Even more, many other full-stack and server-side frameworks are based on ExpressJS, including Sails, NextJS, ItemsAPI, Metro, Sail. Mosty used design pattern with Express is Model View Component (MVC).

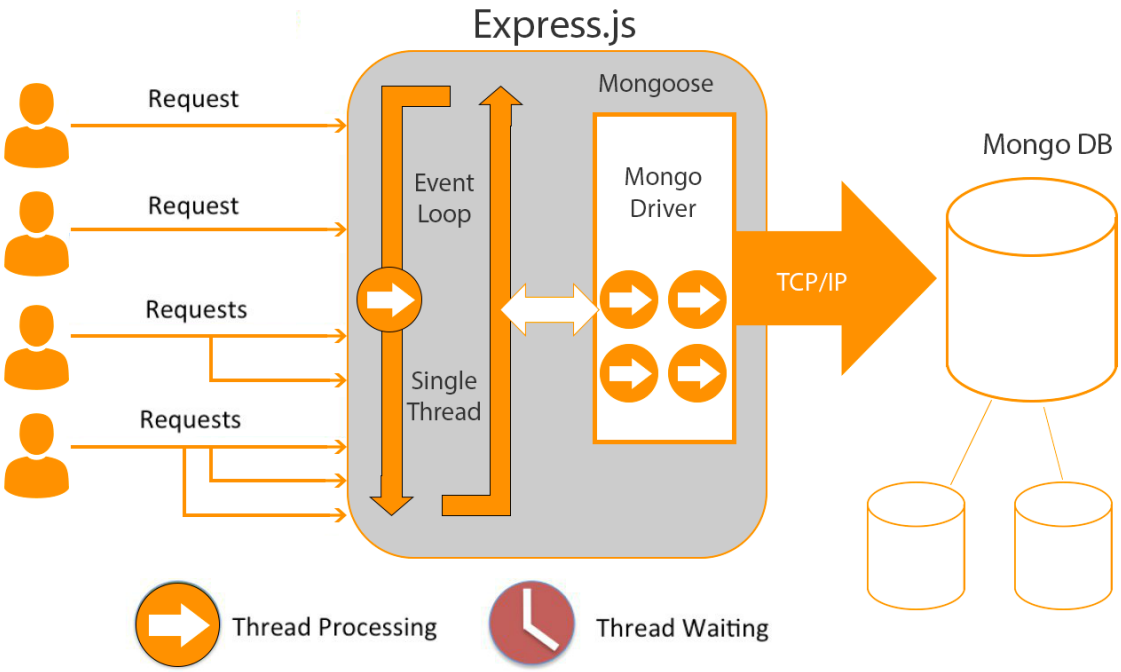


Figure 13:Flow of processing request in Express.js.

Source: https://learnfrenzy.com/blog/top-25-expressjs-interview-questions--answers

Express gives a wide variety of HTTP utility methods to build modular API. It gives us routing abilities based on URL and HTTP methods. There are many ready-to-use plugins and libraries that can speed up development speed and ability to deploy BE faster.

3.2.2. Laravel

Full-stack platform, meaning it can be used for both client and server programming. It’s possible to say that Laravel is the source of PHP, because it is based on it and therefore PHP is still popular. It was created by Taylor Otwell and initially released in June 2011. It achieved stability and wide use in its fifth version. After that Laravel popularity continues to increase. Laravel is a progressive framework, which means it can be used by any developer, no matter of expertise. However, PHP must be required to write code on it. Laravel a "progressive" framework. By that, Laravel grows with you. The first steps in web development with Laravel are super easy because of its libraries and their documentations give all developers the relief not to get overwhelmed (Laravel documentation). It is incredibly scalable. Thanks to the scaling-friendly nature of PHP and Laravel's built-in support for fast, distributed cache systems like Redis, horizontal scaling etc. Laravel reuses the existing components of different frameworks,

which helps in creating a web application. The web application thus designed is more structured and pragmatic.

The latest and stable version of Laravel is 9, in the last updates there were several performance and security updates, with built-in tools and command line interface almost, everything can be done from Terminal.

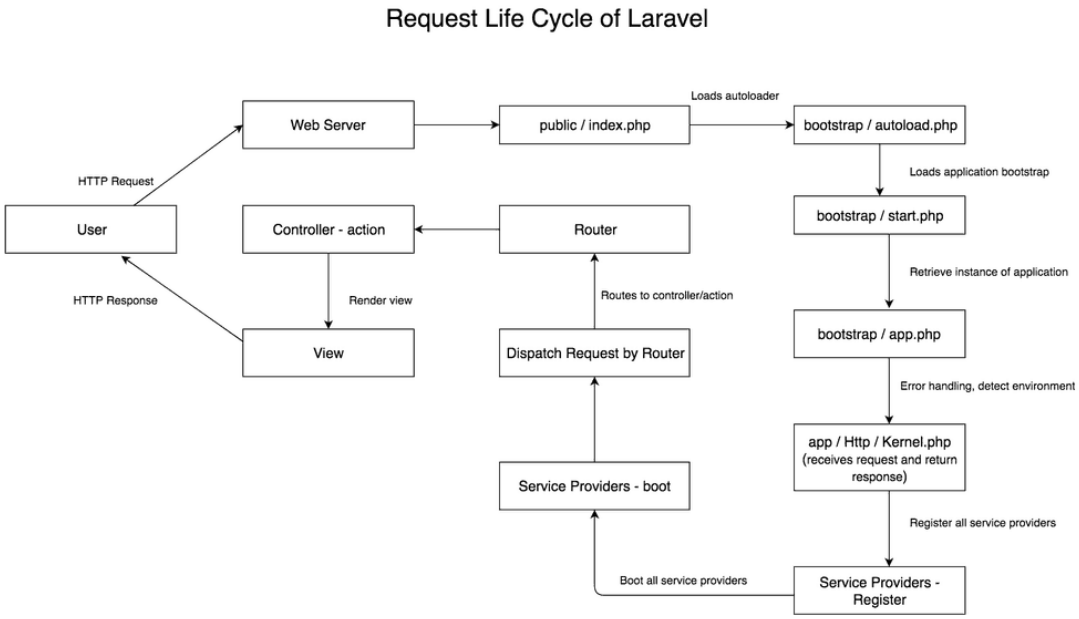


Figure 14: Flow of request handling in Laravel.

Source: https://augnitive.com/request-life-cycle-of-laravel-how-laravel-works-under-the-hoods/

Laravel offers you the following advantages when you are designing a web application based on it:

- The web application becomes more scalable, owing to the Laravel framework.
- Considerable time is saved in designing the web application, since Laravel reuses the components from other frameworks in developing web application.
- It includes namespaces and interfaces, thus helps to organize and manage resources.

3.2.3. Firebase

Firebase is an app development platform that helps you build and grow apps and websites with ease. Backed by Google and trusted by millions of businesses around the

world. It was originally an independent company founded in 2011. In 2014, Google acquired the platform and it is now their flagship offering for app development (Wikipedia, Firebase). Firebase is a great tool for managing databases without any headache, it has many features that can be used well in any platform, one of the biggest tools is Cloud Firestore, with it we have always synchronized data between every platform or device, we have ability to do manipulation on database from Front-end, which give huge opportunity to make and generate queries as we want. With the onboarding/login flow integrated in it, it is easy to connect it to the Front-end. It gives us the ability to use it as SMPT server or send SMSs on phones. Tools of analytics can track users' actions and crashes or errors that can occur on the website. in the Figure 15 there is a full list of features that Firebase can give us:

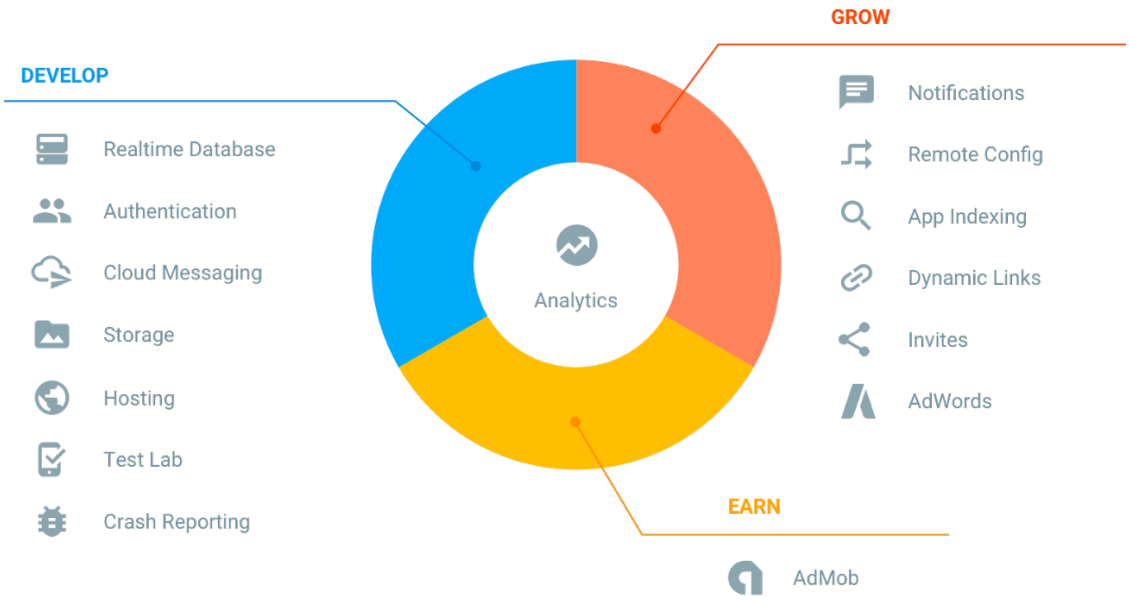


Figure 15:Features of firebase.

Source: https://www.pngfind.com/mpng/iooTixi_all-features-those-makes-firebase-incredible-firebase-advantages/

With Firebase there is no need to think about performance or synchronizing data, you can use it as storage place of platform assets. Which you can reach within milliseconds. one massive thing is that it is seamlessly integrated in Google's ecosystem, for example Indexation for SEO, adMob and cloud functions.

Firebase looks like an excellent choice for platforms who are looking for fast growth with all tools around, as far that you just need to grab them.

3.2.4. ASP.NET Core

ASP.NET Core MVC is a radical shift for web developers using the Microsoft platform. It emphasizes clean architecture, design patterns, and testability, and it doesn't try to conceal how the Web works. ASP.NET Core MVC is a web application development framework from Microsoft that combines the effectiveness and tidiness of MVC architecture, ideas and techniques from agile development, and the best parts of the .NET platform. It uses the Common Language Runtime (CLR). It was first released in 2016 and is a complete rewrite that unites the previously separate ASP.NET MVC and ASP.NET Web API into a single programming model. But indeed, the main reason was losing popularity in community because core code wasn't open source and it was expensive, so they created .NET Core.

In Figure 16 there is an overall architectural overview of framework:

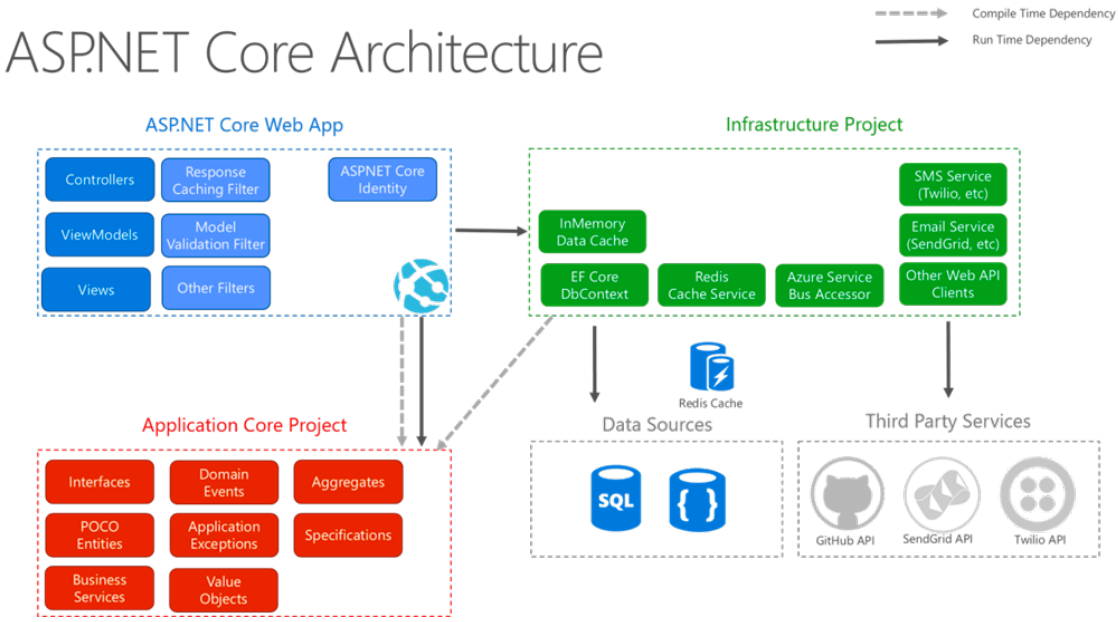


Figure 16:General flow of processing request.

Source: https://www.c-sharpcorner.com/article/asp-net-core-2-architecture-design-pattern-ideology/

ASP.NET Core main writing language is C# which is supported by Microsoft itself and is really similar to Java. At the same time as being BE framework it gives us ability to write Full-stack codes inside like Laravel. There are several pros and cons, but for me game changer was recompiling entire project every time.

3.3 Development tools and technologies definition

For our platform we need two tools to develop stable and performant applications: first if Front-end framework and second for Back-end. Front-end framework gives us the ability to show users all the necessary information that can be used for their future actions and give smooth UI/UX. Back-end technology handles giving FE all the info required by page or the flow, BE is main, core place of the platform, it is controlled from there and with bad BE technology, supporting platform will be hard, data need to be consistent and stable.

3.3.1. Front-end Framework

There are many ways to compare Front-end frameworks to each other, but in the end, we end up with performance comparison, as it is based on facts and there is no other opinion about it. Because of that let's look at Stefan *Krause's repo about front-end frameworks* (Stefan *Krause, 2021*) *benchmarking*, comparison is based on memory usage and cold startup rendering speed.

Name	vanillajs	vue-v3.2.26	react-hooks-v18.0.0	angular-v13.0.0
consistently interactive a pessimistic TTI - when the CPU and network are both definitely very idle. (no more CPU tasks over 50ms)	1,956.1 ± 0.9 (1.00)	2,105.9 ± 0.8 (1.08)	2,555.7 ± 0.8 (1.31)	2,817.4 ± 22.4 (1.44)
main thread work cost total amount of time spent doing work on the main thread. includes style/layout/etc.	163.1 ± 2.5 (1.00)	180.0 ± 13.6 (1.10)	197.7 ± 20.3 (1.21)	320.9 ± 9.6 (1.97)
total kilobyte weight network transfer cost (post-compression) of all the resources loaded into the page.	150.3 ± 0.0 (1.00)	195.3 ± 0.0 (1.30)	260.1 ± 0.0 (1.73)	294.5 ± 0.0 (1.96)
geometric mean of all factors in the table	1.00	1.16	1.40	1.77

Name	vanillajs	vue-v3.2.26	react-hooks-v18.0.0	angular-v13.0.0
ready memory Memory usage after page load.	1.5 (1.00)	1.7 (1.14)	1.8 (1.23)	2.2 (1.51)
run memory Memory usage after adding 1000 rows.	1.4 (1.00)	3.4 (2.33)	4.1 (2.85)	4.3 (2.94)
update every 10th row for 1k rows (5 cycles) Memory usage after clicking update every 10th row 5 times	1.5 (1.00)	3.7 (2.51)	4.6 (3.15)	4.6 (3.12)
replace 1k rows (5 cycles) Memory usage after clicking create 1000 rows 5 times	1.7 (1.00)	3.7 (2.20)	4.8 (2.88)	4.9 (2.89)
creating/clearing 1k rows (5 cycles) Memory usage after creating and clearing 1000 rows 5 times	1.2 (1.00)	1.7 (1.39)	2.3 (1.85)	2.8 (2.31)
geometric mean of all factors in the table	1.00	1.83	2.25	2.47

Figure 17 and 18: DOM mounting metrics and DOM updating metrics for on usage.

Source: https://krausest.github.io/js-framework-benchmark/2022/table_chrome_ 101.0.4951.41.html

The first image describes Startup metrics using by Lighthouse with mobile simulation, which means how fast CPU, JS thread, DOM, can interact with each other.

Second image (Figure 18) shows an experiment, “Ready memory” means the quantity of memory usage after page load and “run memory” means the quantity of memory usage after adding 1000 row. Larger time and memory mean that it holds more features and functions, but it will spend more time loading. As you can see there is one additional column (VanillaJs) which doesn’t occur in the document, it means native JavaScript, API that is created for manipulation on DOM, frameworks are based on that API, we can say that they are wrapper of the JS API.

From the benchmarks we can clearly see that VanillaJS is absolute winner, but speed isn’t always game changer, nowadays the biggest part of development efficiency comes from how flexible and easy it is to manage state and UI on the website, which is hard on vanillaJS.

The best framework according to benchmark is Vue, after that comes React and Angular. As I said Vue and React have similar core logic, but the difference is the way you build the layout. Even more React has NextJS which is brilliant framework for building e-commerce websites. React is well tested in big enterprise projects and has huge community support. Therefore, the front-end framework that was chosen for the development of the web application is React.

3.3.2. Back-end Framework

This platform can have many iterations of development and product revisions, because of that we need easy, flexible, stable and scalable platform, if BE is easily scalable there is no need to compare its performance benchmarks to each other, in any case because of the easy scalability in the end it will not give us noticeable difference. All the frameworks have pros and cons but there is only one platform that satisfies all above-listed requirements- Firebase. Developer experience compared to other platforms is almost perfect, debugging, perfecting queries can be done without any headache. Its scalability and performance come from Google cloud servers and its database language which is No-SQL and looks similar to MongoDB. Therefore, we will use Firebase as the main BE platform for handling important logics and queries that will drive the entire website.

3.4 Other tools and technologies

After choosing Front-end and Back-end technologies now it is time to choose the working environment, task management tools etc. First, our main code editor will be Visual Studio Code created by Microsoft, best one in community. Good hosting services are super necessary so because of that I will use Vercel for Front-end, it is the best platform for hosting NextJS projects, at least they are the creator of it so it must be good, and Firebase is already hosted on Google's servers so we are safe here too. For Back-End NoSQL Database will be used, as it is trusted by Google.

For task management was used Trello with Scrum Agile, a leading method in Web development, it is widely used worldwide and Kanban board for processing the tasks.

Tracking changes in codes are important so we are going to use GitHub as the biggest player in this ecosystem. With Vercel we can build Continues integration and Continues delivery (CI/CD) pipelines and we can publish every commit in specific subdomain. It gives us the ability to test and see commit in real situation for protentional bug fixes.

Chapter 4 Development of Marketplace for Bioma

4.1 Introduction

Development of Bioma Marketplace started in November of the 2021 and ended in May 27-nd 2022, using scrum Agile as development methodology. In total there were several sprints/circles where every spring was lasting for 1 week. In this chapter I will explain in detail how task management was processing. Also, I will talk about the final product, how it become real life product with all features that user can utilise.

4.2 Development Operations (DevOps)

The main principle of the development process was to design task correctly and write clean code so that future editing will be easy, though designing correctly also means that there shouldn't be any further changes in functionality, but in the end, everything is changing, even for future enhancements so clean and good code is necessary.

On Trello I used Kanban Board for most of the sprint to manage tasks. There is all essential column that is necessary for proper task management:

- Backlog – first stage of task creation, here we need to place the general idea of the task
- Design – second stage, here we are adding as much details as possible so that it is ready to be done and there are no blurry moments.
- To-Do – we are ready to start working on it
- Doing – task is in process of doing.
- Code Review – after finishing the task codes need to be reviewed and checked for quality, if everything looks good it needs to be approved and merged to proper branch, according to code environment

- Testing – after merging and releasing it on staging environment, we need to test it against some edge cases that can have fail the flow. After successful testing task is moved to production

- Done – task is finished and changes are in production.

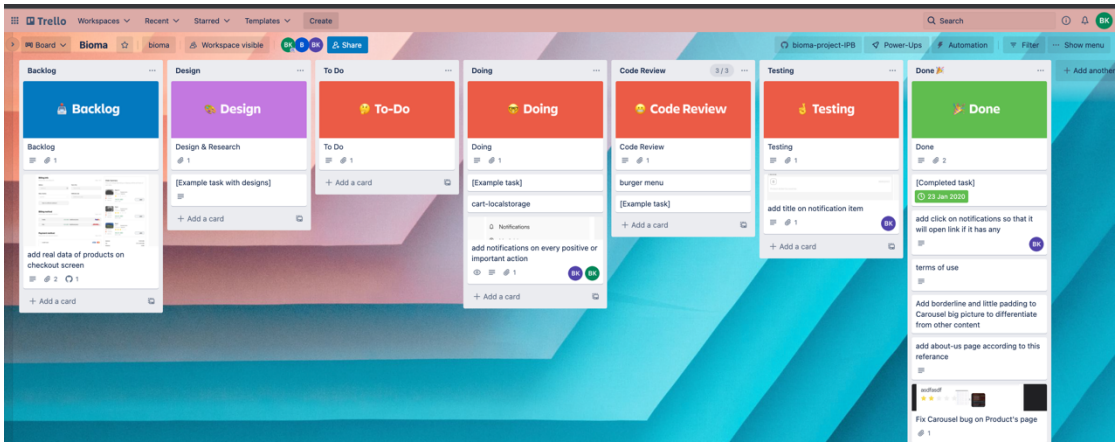


Figure 18:Preview of columns that were used for task management in Bioma Kanban board.

Trello and GitHub can have seamless communication with Power-Up feature from Trello, it gives us the ability to connect GitHub Pull Requests and commits to Trello tasks, which means that moving tasks from one column to another will be done automatically, and tasks will be always up to date which is important for correct Kanban methodology.

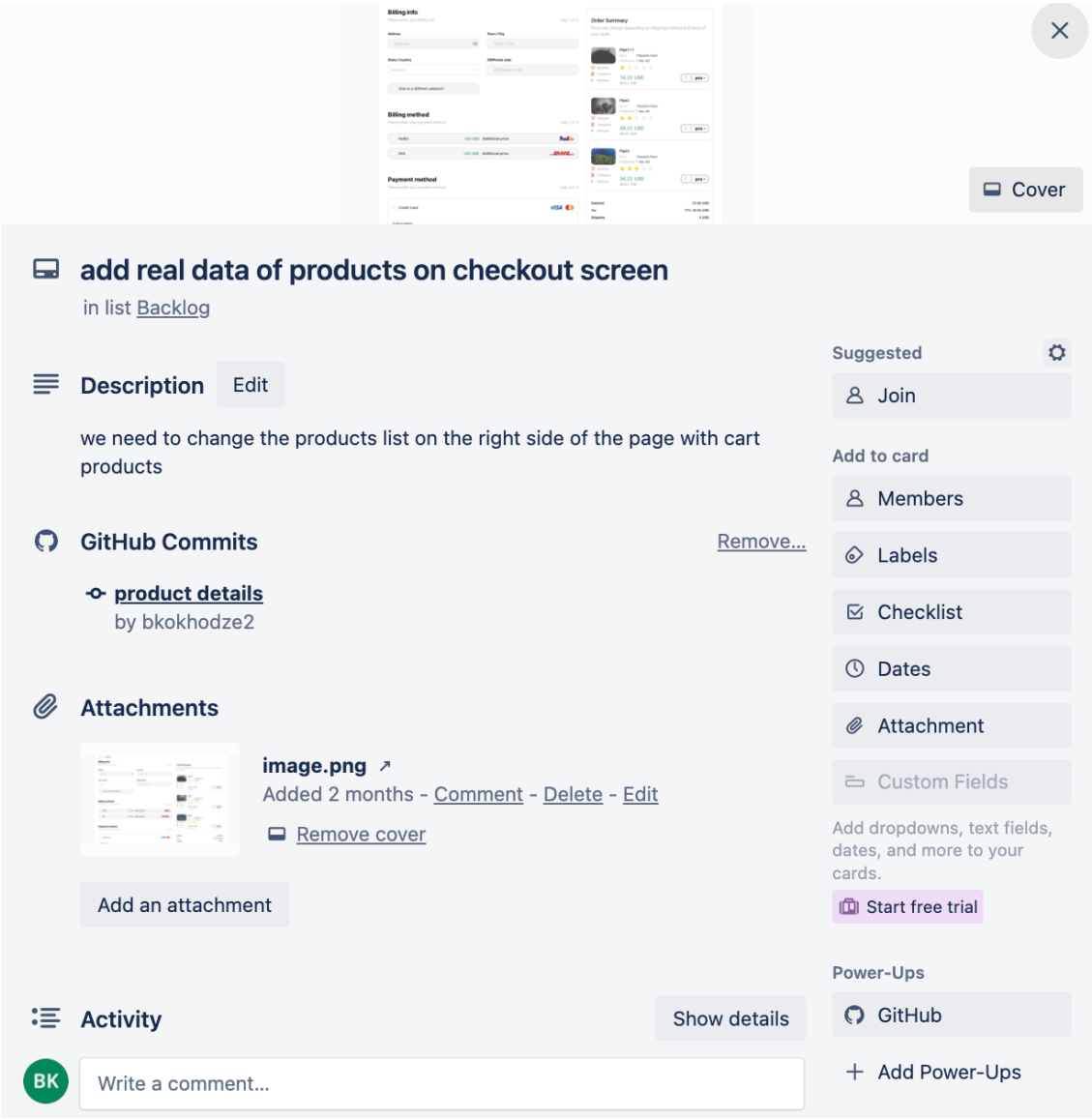


Figure 19:Preview of how task is being written and how GitHub is connected to the task.

Bellow picture represents main branches that were created for Bioma Marketplace. In default, the development and default branches are "dev" and production branch is "master". Usually default branch is "master" but it is incorrect because most of the work will be done on the default branch so it needs to be accessible and all PR need to be aimed automatically at this branch. After codes are merged to "dev" and are tested it is moved to "master" branch by PR. All branches and commits have their own website preview link.

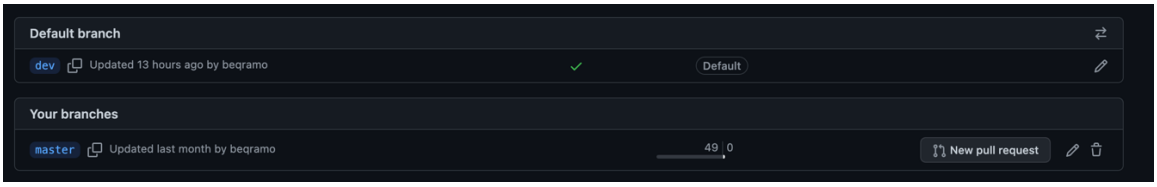


Figure 20:List of main branches used in Bioma.

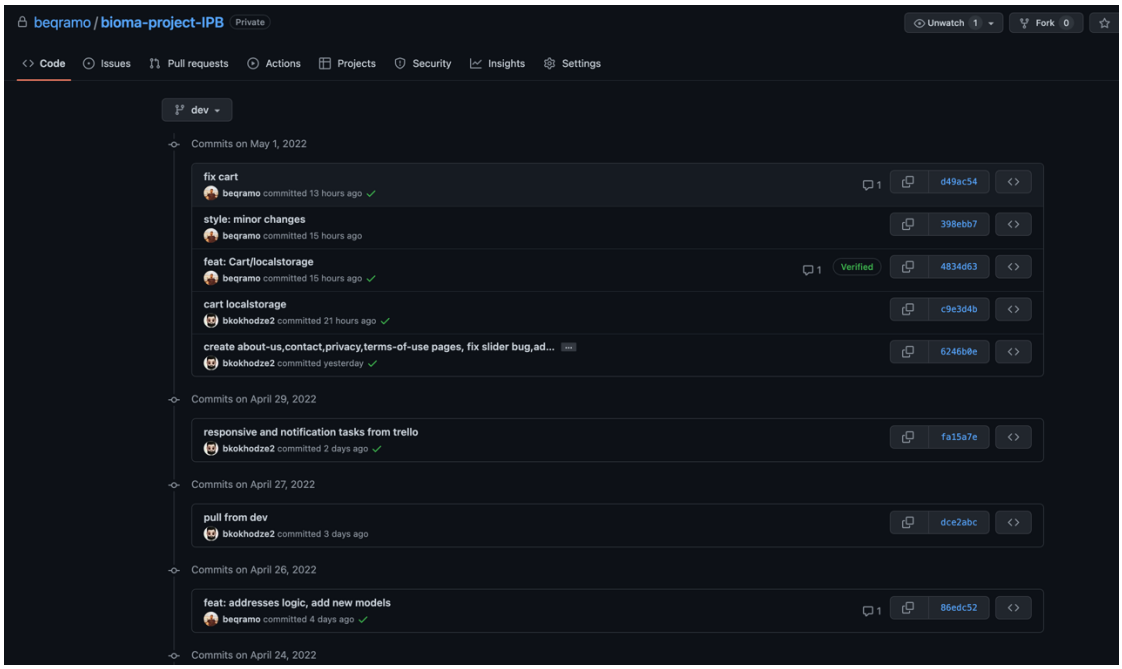


Figure 21:The commits created on my main development branch “dev”.

4.3 Header section of website

When users enter on any page, they will see contact information and about us page, it is standard to have easily accessible info that can be helpful for users at first. This part is static across pages. There is Bioma logo and search input so that users can search for any product that they want according to title, description and location. On the right side we have two buttons – (1) profile button, if user has an account it will navigate to account details. (2) cart button, on click on user can see their chosen products (Figure 23, Figure 24), they are able to change the quantity of product right from popup, but user will not be able to set quantity outside the range that seller set (min quantity and max quantity values). If user is an Administrator/moderator he or she will see a button on the top as “Admin” and on click he/she will be navigated to the admin panel.

Last one is categories; I’m showing all categories available and if user hovers on the category, they will see subcategories of the categories in the dropdown menu.

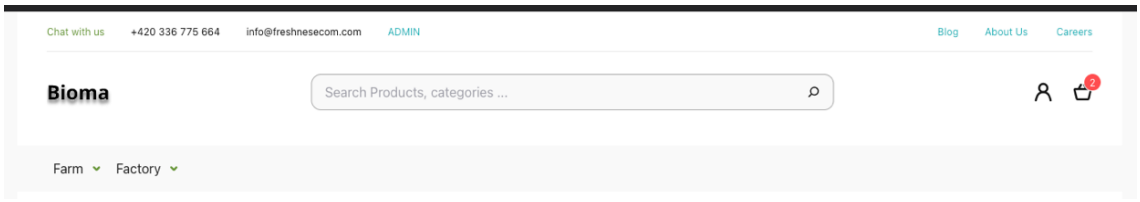


Figure 22: Static header section for all pages.

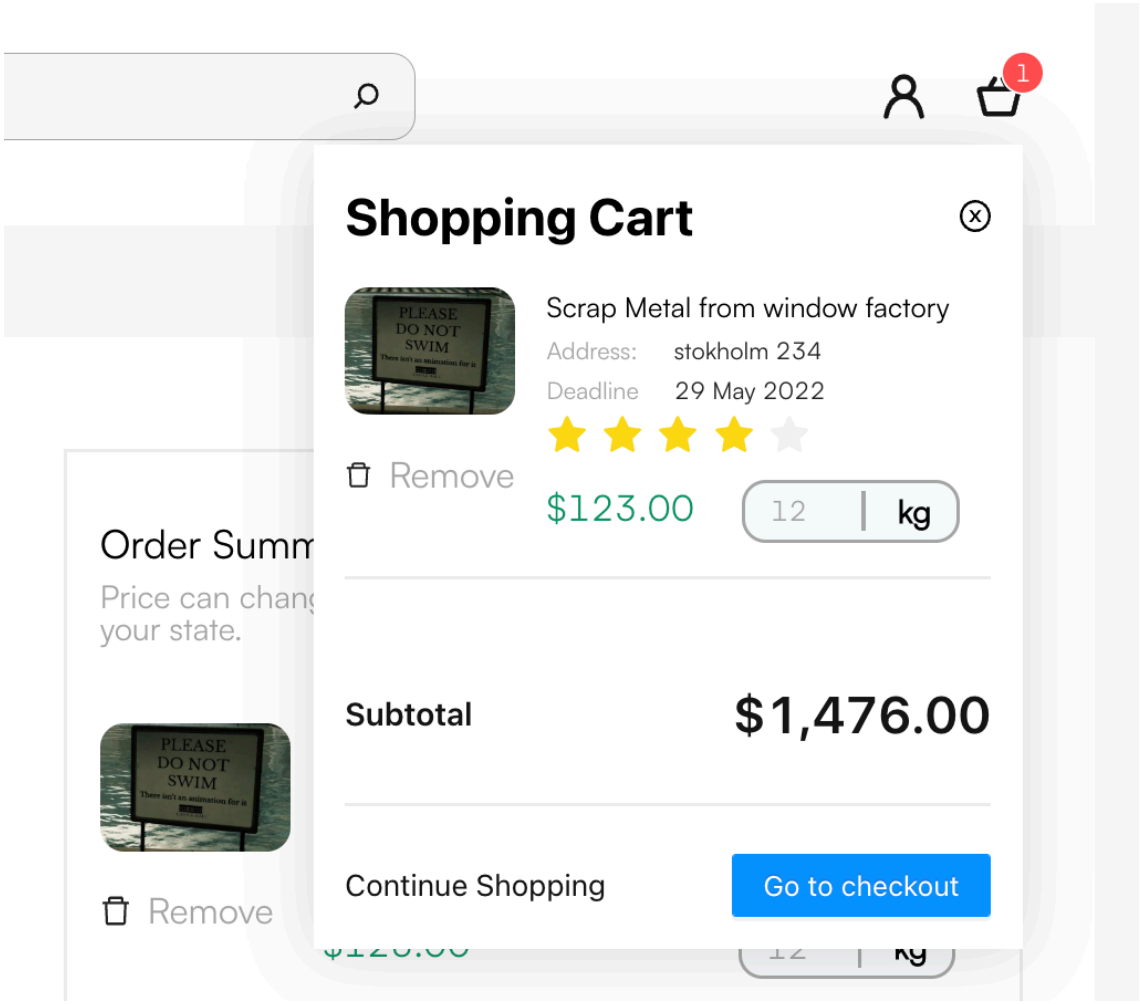


Figure 23:Cart when there are products inside.

4.4 Home page

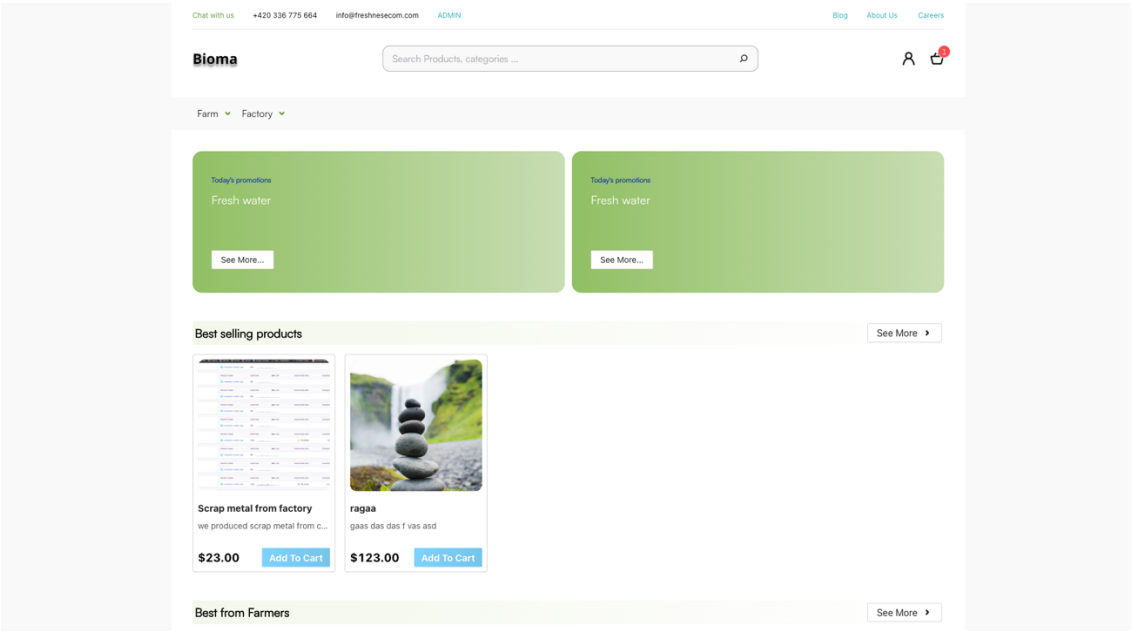


Figure 24:Home page from users' perspective.

The first page that users need to see is a home page, where will be a list of products and pinned categories as sections where users can see most wanted products or best sellers. There is one section called “today’s promotion”, where moderators/administrators will be able to add pinned products and promote them so that they will catch users’ eyes. It can be any product or even any category that moderator or administrator wants.

After it we see a list of products grouped by one property, moderator/administrator can add as many sections as he wants to that list. Indeed, these are products grouped by hidden categories and moderator need to add product to the specific category which is hidden, which means that during the creation of a category there is a checkbox that needs to be marked if it is a hidden category and it will not be visible for users. Now we see *bestselling products* and *best from farmers* which includes products that were added to those categories (*bestselling products* and *best from farmers*) so that moderators will have the flexibility of showing any item they want on the first screen. Every section has a slider in it and the user can swipe on elements.

Each product item has a description, title, image, price and “add to cart” button which triggers cart functionality and adds one product or minimum required quantity by the seller, to the cart. On item, window click users will be navigated to the product’s page where they will see a more detailed view.

4.5 Login/Registration/Forgot Password

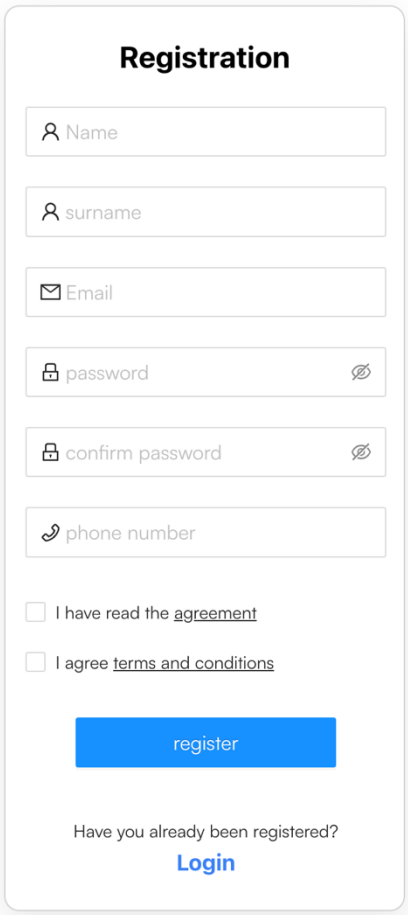
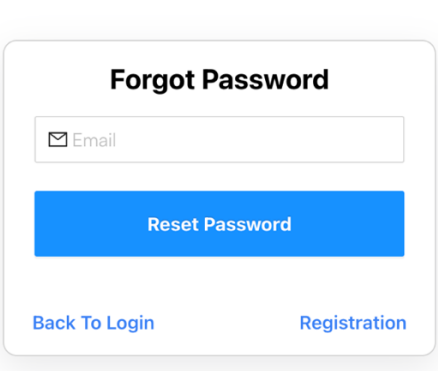


Figure 25:Registration.



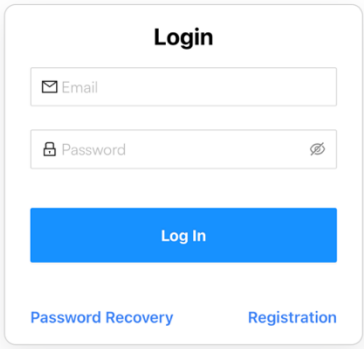


Figure 26:Login component.

Figure 27:Forgot Password.

For registration user need to fill all fields that is shown above, all of them are required. The unique identifier is email so it needs to be unique for all users They need to agree on our terms and agreements. After clicking on “Register” user will receive email to verify his/her account. After that user is navigated to home page. This is a creation flow for administrators and for moderators too, after that user's status need to be changed from admin panel or database by administrator.

On login user need to use unique email and password, if user is administrator/moderator they will be navigated to admin panel, if not on home page. If user’s email isn’t verified, he will not be able to log in, he will see error message.

There are many cases that passwords are getting forgotten so it is essential to have reset password flow. The flow was simplified so now users need to just enter email address and they will receive email with link where they can update passwords. After updating they can use that password for login.

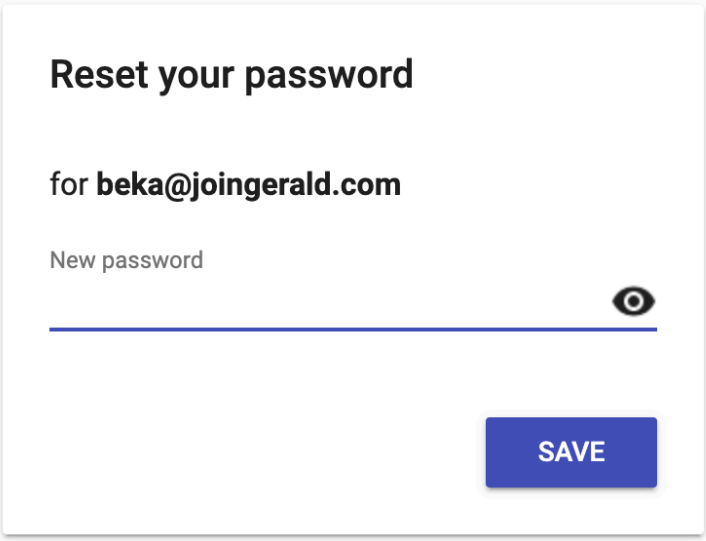


Figure 28:Reset password page’s popup on browser.

4.5.1. User's profile section and My account

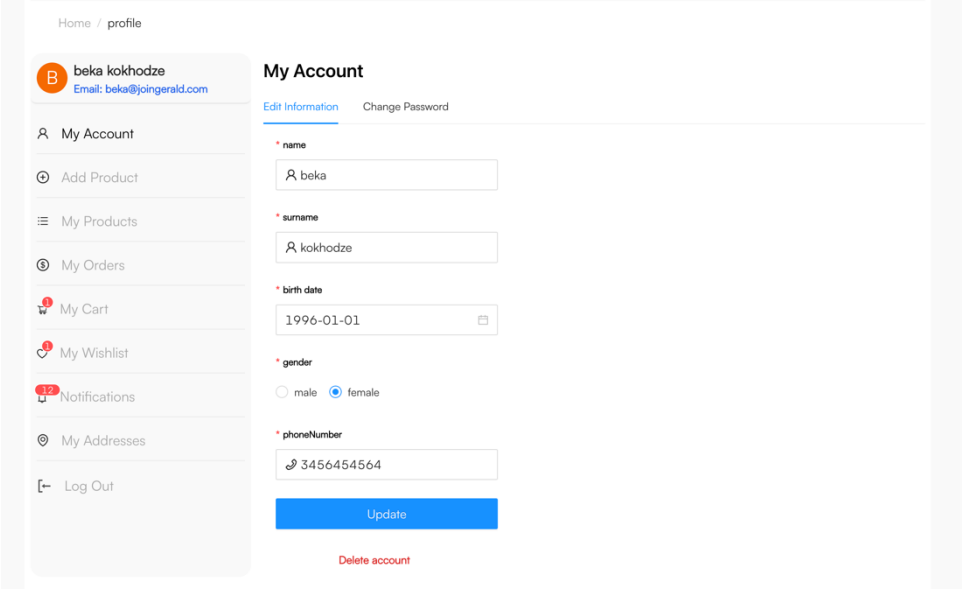


Figure 29:Profile screen for authenticated user.

On avatar click in header user will be navigated above screen where user will be able to do essential functionalities for profile.

There are two tabs – *Edit information* and *Change password*, User can change almost all essential info that he entered during registration, with same validations. Here users will be able to deactivate/delete account. For changing password, they need to put current and repeat new password twice.

4.5.2. Add/Sell product

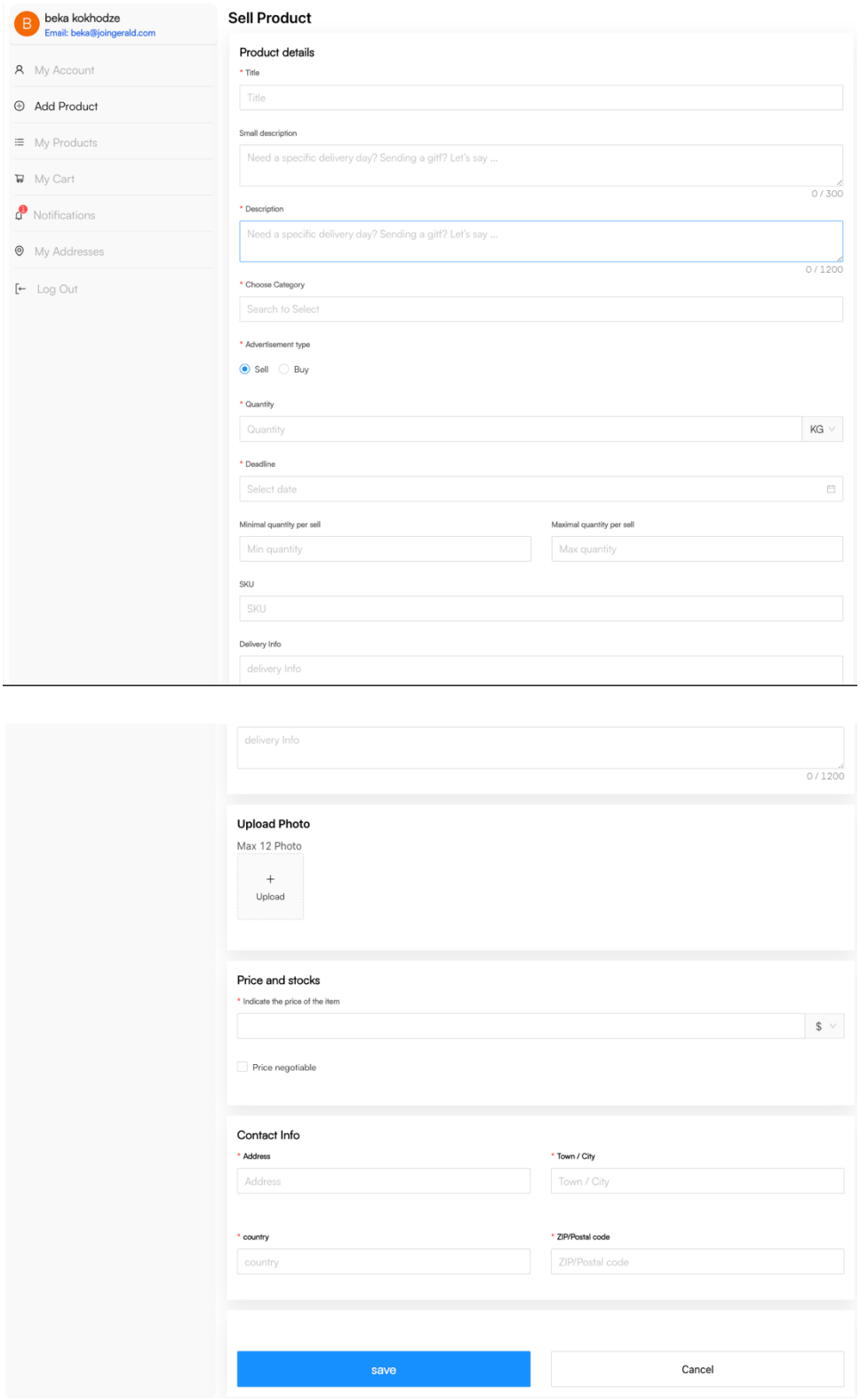


Figure 30:Fields for sell/add product.

Main functionality and idea are based on adding products freely and easily. Usually, it is good to have as many info as possible but it creates boredom so users are

getting tired and giving up on filling all inputs so I tried to minimise required fields so that process would be linear. I will try to explain all fields responsibility in our platform bellow:

- Title – used for as the main identifier of product for user, it needs to be informative and short.
- Small description – many websites doesn't use this field as it is extra work for user, but also it is handy too, seller can promote product shorty which will be shown under the title, we can call it promotional text too. Also, it isn't required, so it is fully depending on the seller.
- Description – this is full product description's place, here seller must put every detail that can be useful for user.
- Advertisement Type – during adding product user has two options, sell or buy, it determines what kind of advertisement it is. Normal user will have different flows for requesting offer according to the advertisement type.
- Quantity – it is required field and it means how many items is available for selling/buying. On the right side it has dropdown which is responsible for choosing true measurements type, quantity need to be in KG, PCS, T etc...
- Deadline – there are times when product is required to sell/buy before any date. Because of that user need to fill this field, after that date, this product will be archived.
- Min Quantity and Max Quantity – user can add min/max how many products they want to sell/buy at once, it is optional but it gives great functionality to sellers/buyer, sometimes 1kg is nothings for such a trading.
- SKU – this is used for product identification, sometimes companies are using their own internal identifiers for products, so now they can add it to the product info for future use.
- Delivery Info – it includes all info connected to delivery, even if it is possible or which companies must be used for this.

- Upload Photo – this field is required, there must be at least one photo of the product. It is bad user experience when watching product without a picture. Largest photos count is 12. During upload they can crop and rotate photos.
- Price – price of one product, on right side it has currency drop-down and user can pick their desired one.
- Contact info – all these details are used to locate where product is based exactly. The reason it is separated into 4 fields is that some countries doesn't have good support of any map location API, so we need to get main info separately.

4.5.3. My product

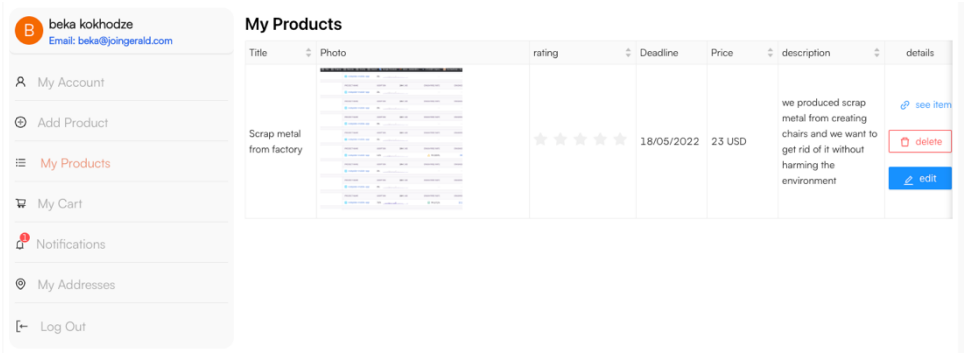


Figure 31:Page for looking products created by user.

After user adds the product, he/she can see the product in this section, user will be able to see earlier product orders too that has already been expired.

4.5.4. My cart

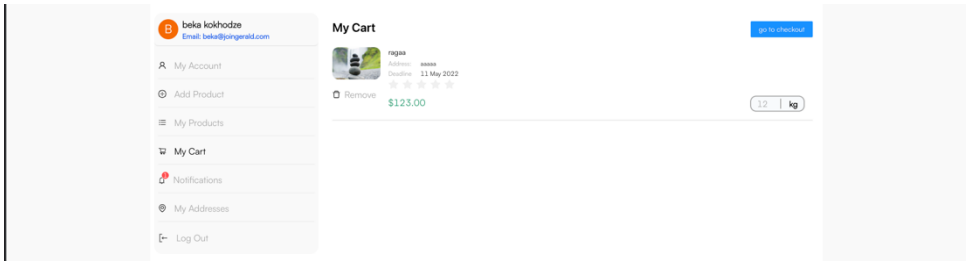


Figure 32:My cart page.

It is realisation of the above-mentioned cart popup as page, user can do same manipulation as on the cart's popup.

4.5.5. Wishlist

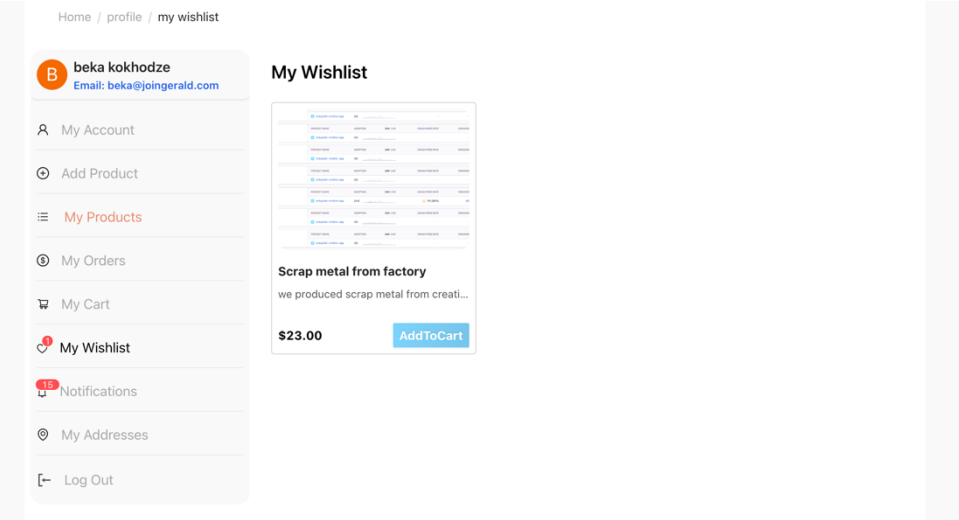


Figure 33:Wishlist page.

User can add items on wish list even it is out of stock so they can check it time to time. Adding on wish list doesn't mean that it will be added on cart too, user manually need to add item to cart even from wish list add too cart button.

4.5.6. Notifications

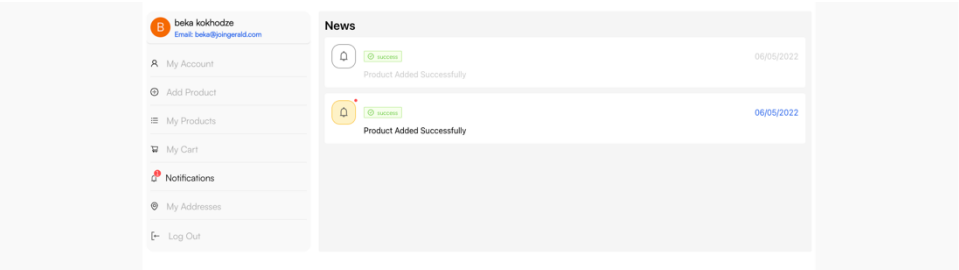


Figure 34:Notifications page for user.

Notifications are the main source of every update on platform, here they will see general updates, like any agreement updates on website, platform changes, product statuses, product orders and notifications about rating. When someone gives any kind of rating, user will receive notification about the rating that seller received and final, average rating. They will receive product created notification when they will add the product and they and after clicking on notification they will see link of the product and on click they will be navigated on this product's page.

On "send offer" they will receive notification for that, there will be written every detail that is necessary to contact to sender, also offer is sent on email too so there will be access on his/her email address.

Users will see badge on the screen like it is on the screenshot if there are any new notifications, to decrease count of the unread notifications' user need to click on new notification and it will automatically mark notification as read.

4.5.7. My addresses

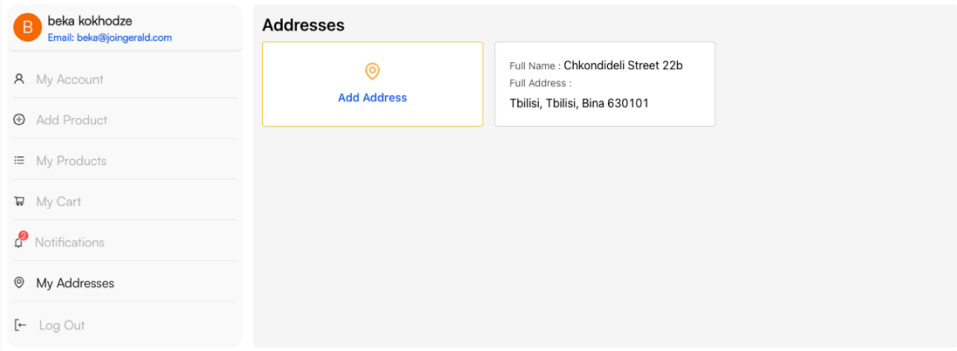


Figure 35:User's addresses list.

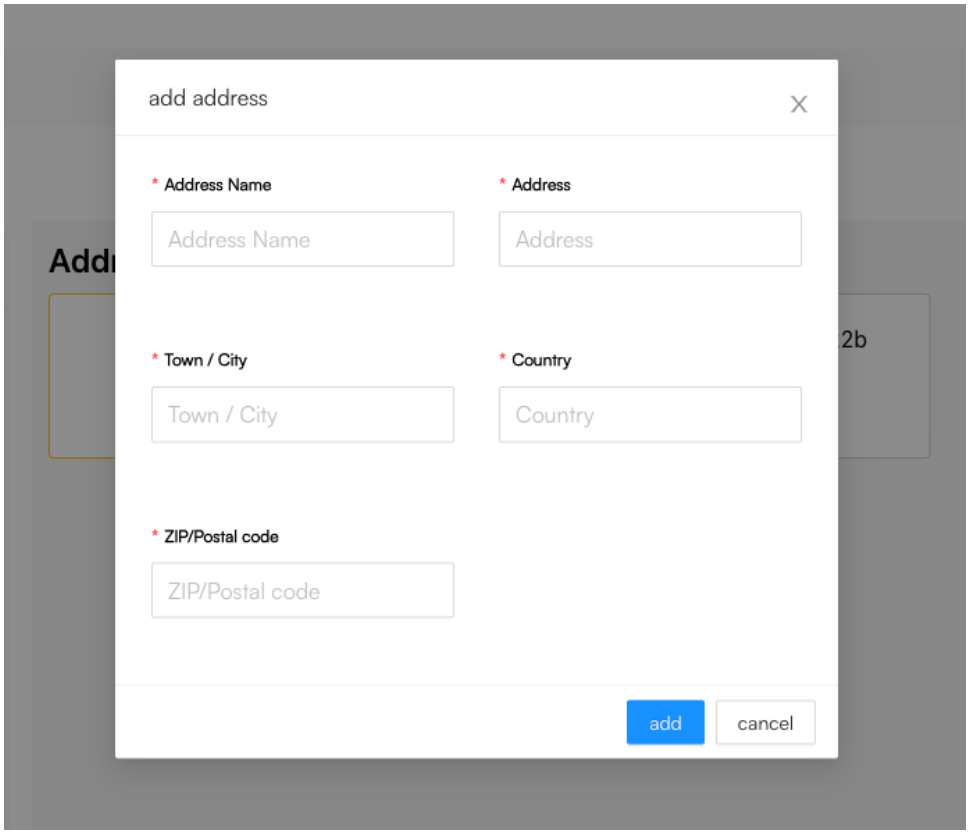


Figure 36:Adding new address for user.

My address is important section for user, it will be used for a location where he or she will receive the order, user can add up to 3 addresses and they will be differentiated by “address name” and they can select it in dropdown on checkout page if they want to use already added address as preferred delivery location.

4.6 Product page

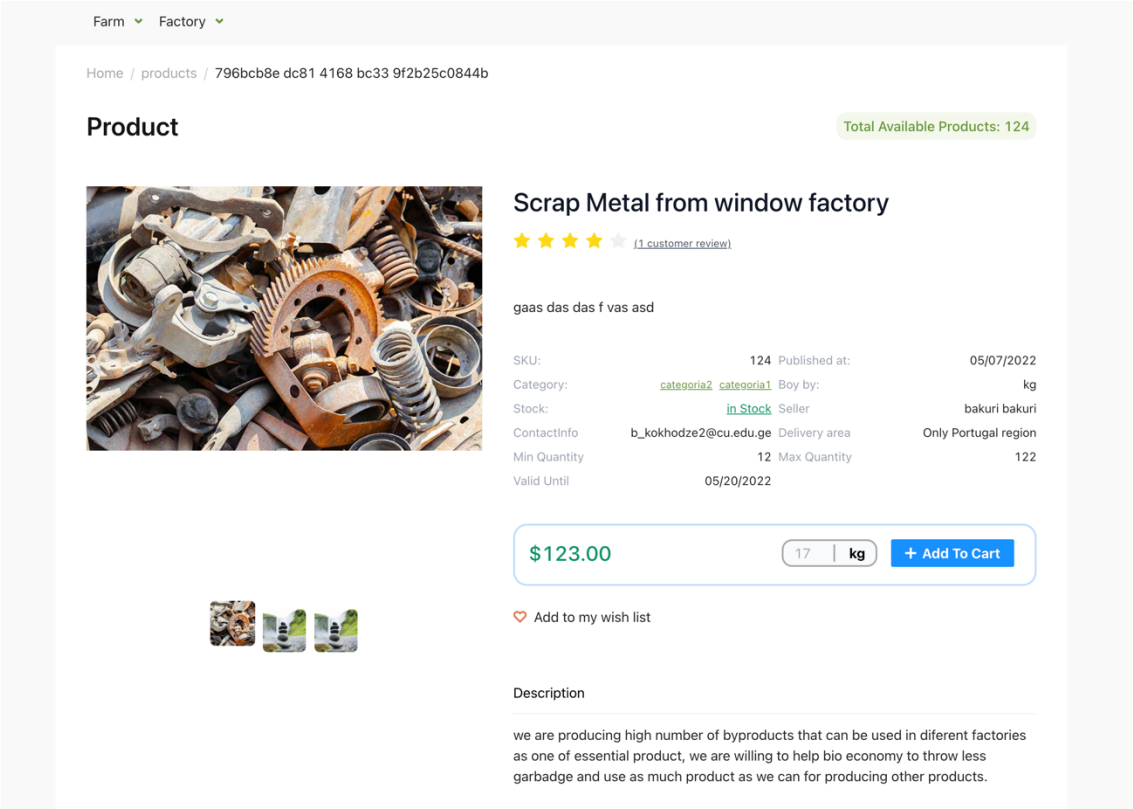


Figure 37:Product detailed page.

One of the most important pages is product’s details page, where you will see every information that you will may need.

I tried not to show all of the info that I have about product so that it wouldn’t be a mess for users. Here users can add products to their wishlist, add products to their cart and then do a checkout, and look on rating and give rating too. After giving a rating user can change the rating, if the user isn’t logged in, he or she will not be able to give feedback, they need to log in so that it will be possible to track the user’s feedback and will be protected from spam feedback.

4.7 Checkout page

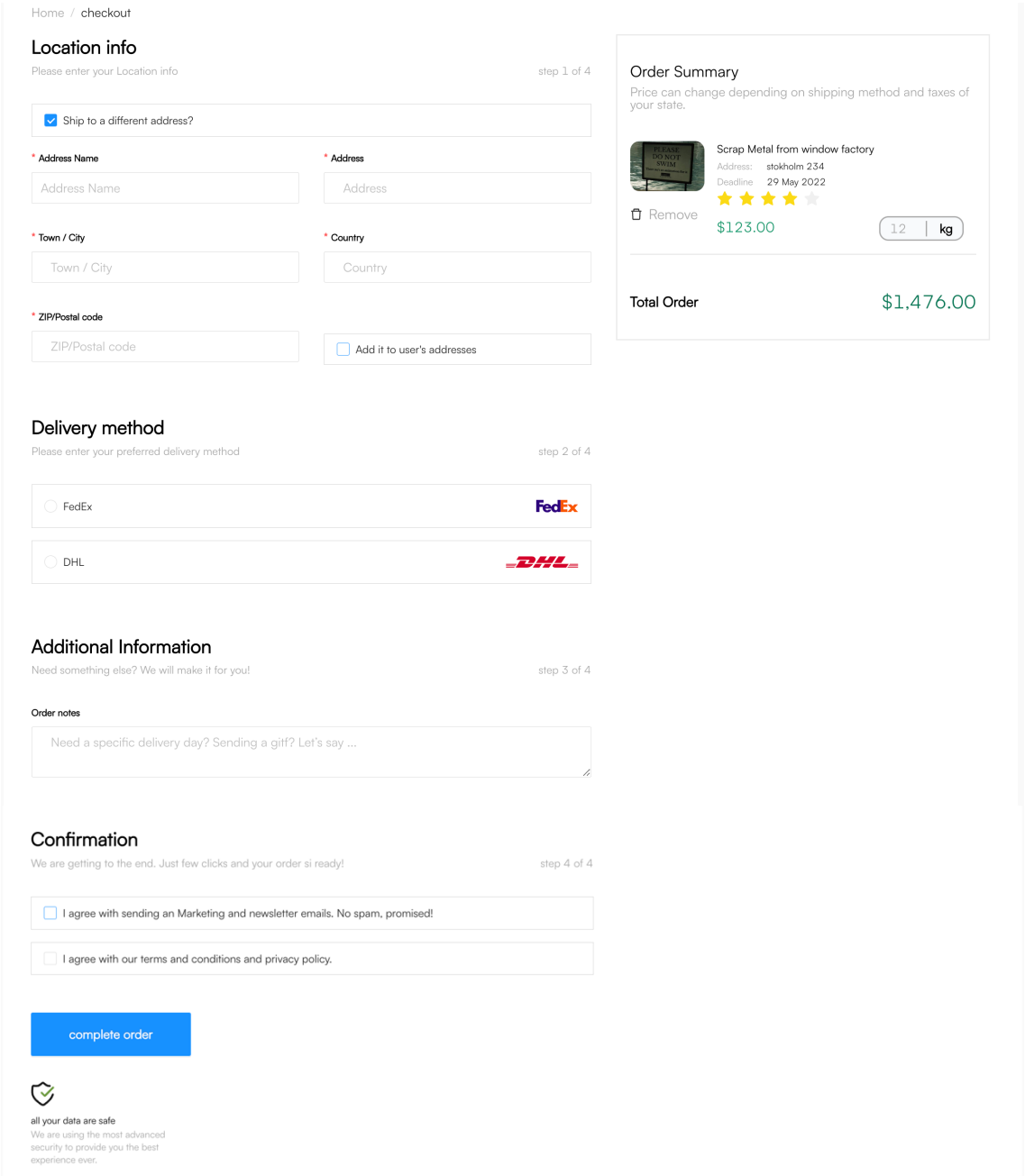


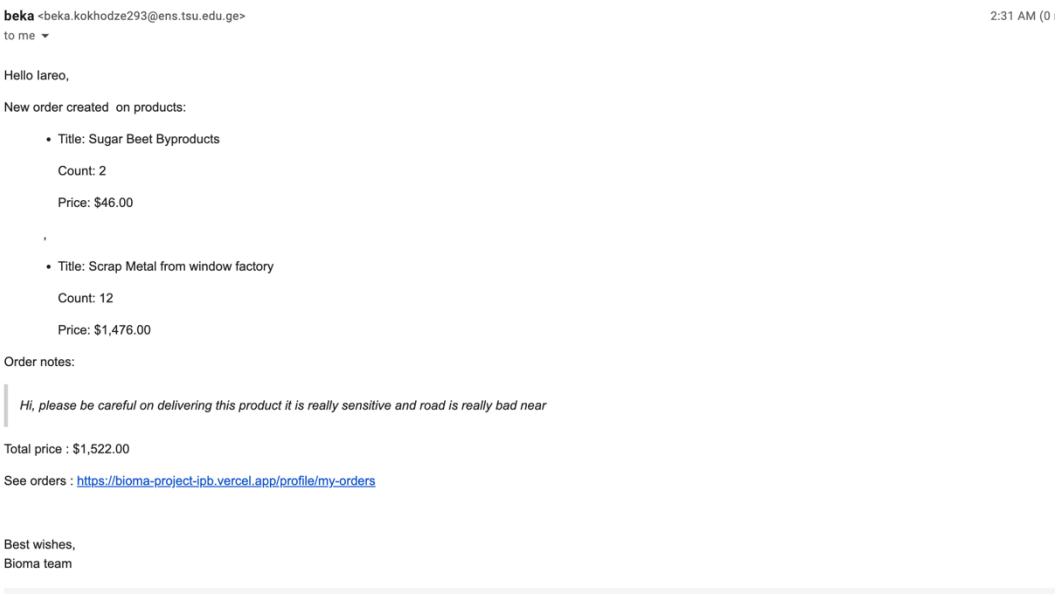
Figure 38:Checkout page, last step of the main flow.

The last page of the main flow where users need to finalise necessary details. User is free in selecting location, but they can add new address, or just choose one from their addresses that can be added from prefile section.

In additional info section user can write product specific comments which can be read by any person from another side, or preferred delivery information that can be passed to carrier, useability of this is input is enormous.

Users need to check checkbox of terms and conditions so that it will be valid process from user standpoint, user can market checkbox where they will receive promotions and News on email.

After clicking “complete order” process of creating order will start and after some seconds user will receive confirmation on website and via email too. So that now it is sellers' duty to send product to its location.



beka <beka.kokhodze293@ens.tsu.edu.ge> 2:31 AM (0

to me

Hello Iareo,

New order created on products:

- Title: Sugar Beet Byproducts
  Count: 2
  Price: $46.00
- Title: Scrap Metal from window factory
  Count: 12
  Price: $1,476.00

Order notes:

> *Hi, please be careful on delivering this product it is really sensitive and road is really bad near*

Total price : $1,522.00

See orders : https://bioma-project-ipb.vercel.app/profile/my-orders

Best wishes,
Bioma team

Figure 39:Created order email confirmation example.

4.8 Order type - *buy*

As it was said user can place *buy* or *sell* type of announcement, if type is *sell* flow is same as it is shown above, but flow changes little bit when we have *buy* type, user can’t go on checkout because there is nothing to do checkout on. Because of that user can send offer to announcer about his offer and announcer can review that order.

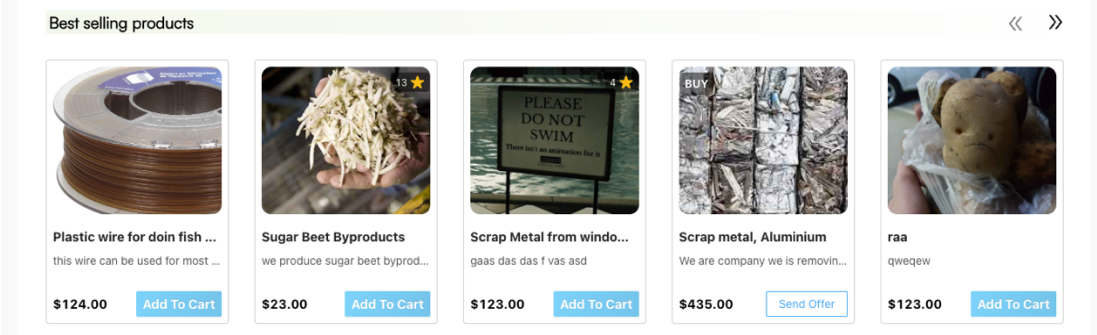


Figure 40:Home page section when product deal type is *sell.*

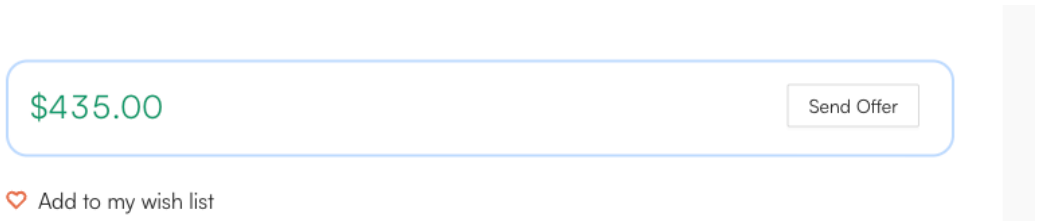


Figure 41:Product info page section when product deal type is *sell.*

As you can see instead of adding to cart, we have different button and on click they will see the form like it is shown in Figure 43 to fill. It is required to detailly fill description input so that receiver will have full idea about your offer.

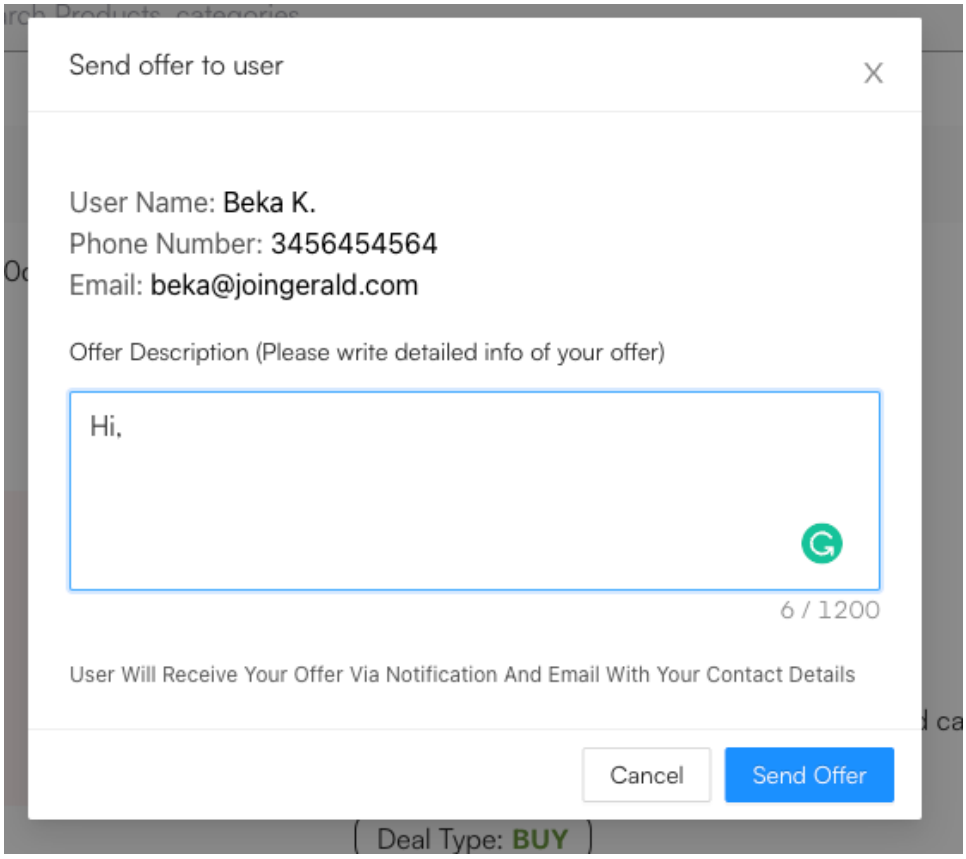


Figure 42:Send offer form for users.

After clicking "Send Offer" they both, sender and receiver, will receive email about the offer. After that if user will be interested, they can continue communicating with each other.

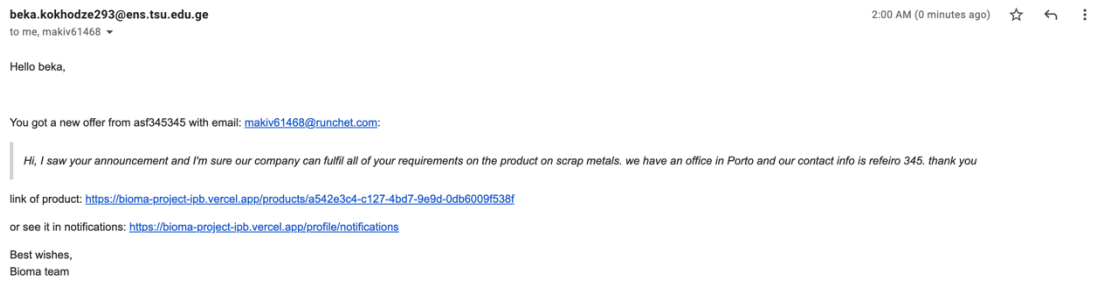


Figure 43:Received email about proposed offer.

4.9 All products with filters

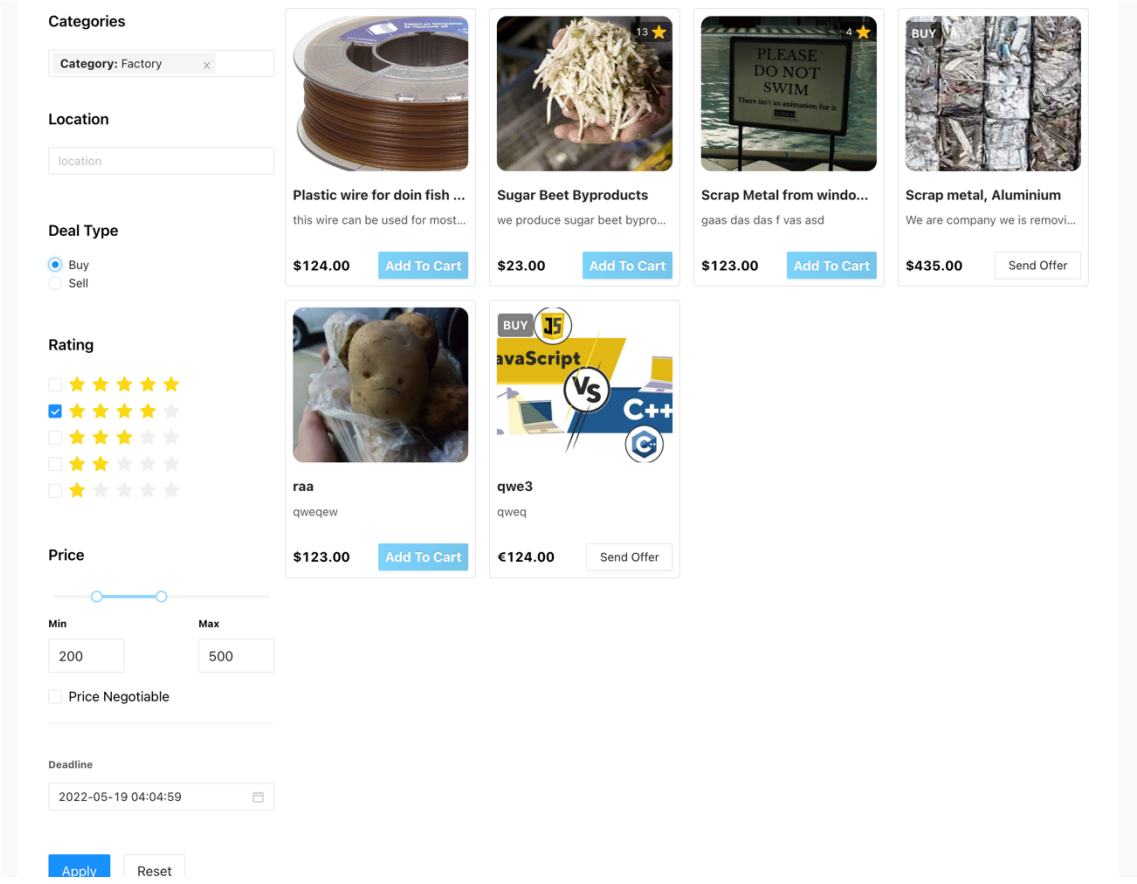


Figure 44:All products page with filters.

It is necessary to have ability to filter products and show all products that are on platform. User can get on this page so usability of this page will be high, here performance is important, so we need to try to maximise it. User can use the link of filters and left side filter options that will change URL of the website too so that it will be easy to share or track navigation history change.

User has ability to filter products according to categories, they can select as many categories as they want. Most wanted and important filter is location filter, it is super easy for user to search product in the region where you can reach, because some products can't be transported on long distances. With this location field user can select city, country and platform will show products inside that city, country.

On platform there are two types of products, so it is necessary to filter according to it too. Filtering according rating and price aren't something new and it is somehow mandatory to have it. Last but not least – deadline, every product has time until when it will be valid and will be placed on our platform, so it is important to see only products that fits in users' schedule.

4.10 Admin Panel

Every big and successful platform need controller panel where privileged people will be able to manipulate on platform which includes choosing what to show on home page of the website, which is first page where user will have interaction; change product information and remove it from there if necessary; change statuses and information of the already created order etc.

4.10.1. Home Page

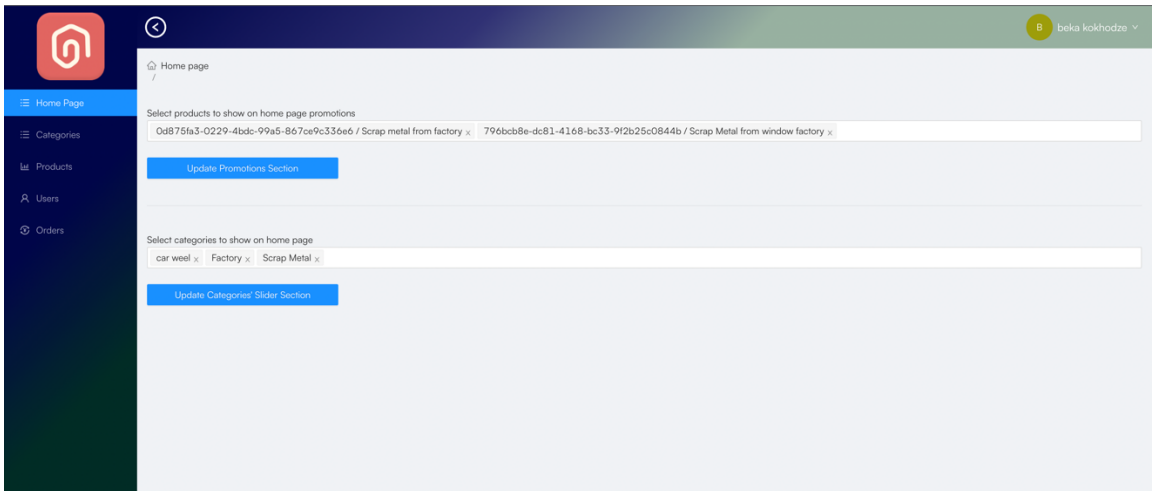


Figure 45:Home page controller in admin panel.

It has only two fields, where Administrator/Moderator can flexibly choose promotion section products which must be 2 or nothing other case validation will throw error, it is highly suggested not to left empty these fields.

The second field is used for creating sliders like “Top Categories”, “Most trended” or anything else, moderators can add as many categories as they want. After it, all users will be able to see products of these categories as a carousel.

4.10.2. Categories

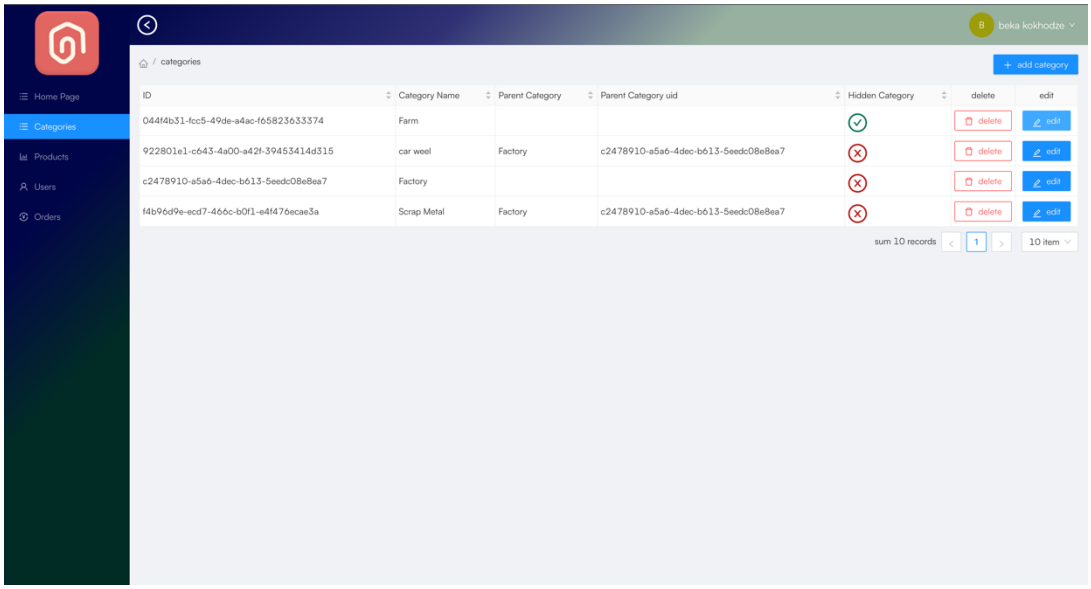


Figure 46:Categories controller and creation page.

In the admin panel, we have a page for categories, the only place to create them is here. All necessary details are shown that could be necessary for future manipulations. Moderators and administrators are able to edit and even delete category. Categories working logic isn’t as default as you may face on other platforms; we use two level category system which means that one category can have only one parent category or zero parent category, if the first one is the case I’m calling that *child category*, if second then *root category*. In Figure 48 there is shown creation modal where parent category field is optional. Which means that if user leaves it empty it means that that category is *root category*, if user choose parent category from the list that will show only *root categories* (we need to filter it so that we will get rid of more than 2 level hierarchy of categories), it will be *child category*.

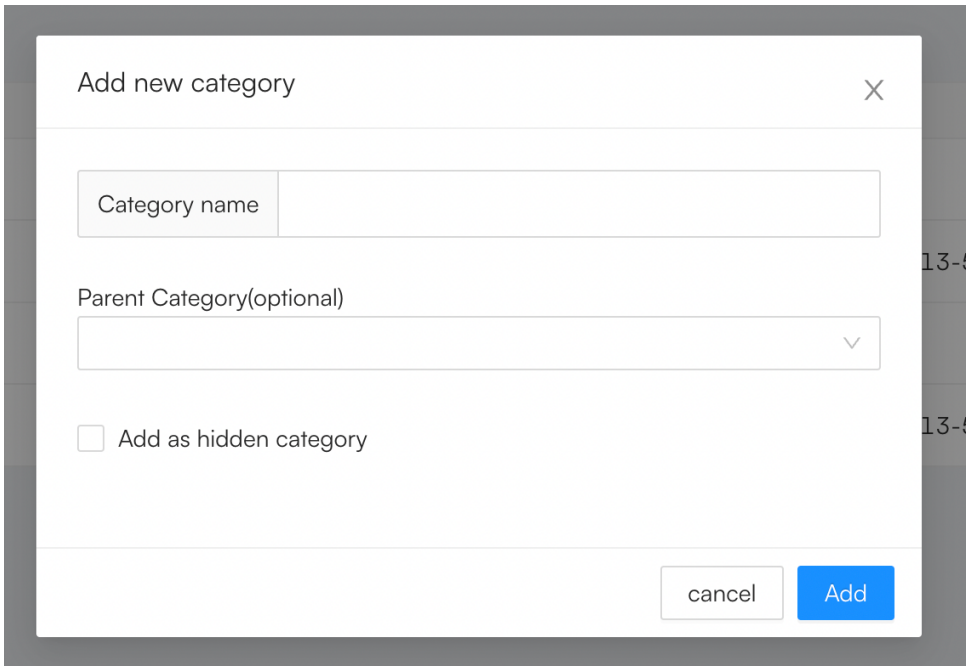


Figure 47:Modal for adding category.

During adding or editing category, user can mark category as hidden, which means that it needs to be used for home page so it will not be visible for default users. These kinds of categories will be used on home page sections.

4.10.3. Products

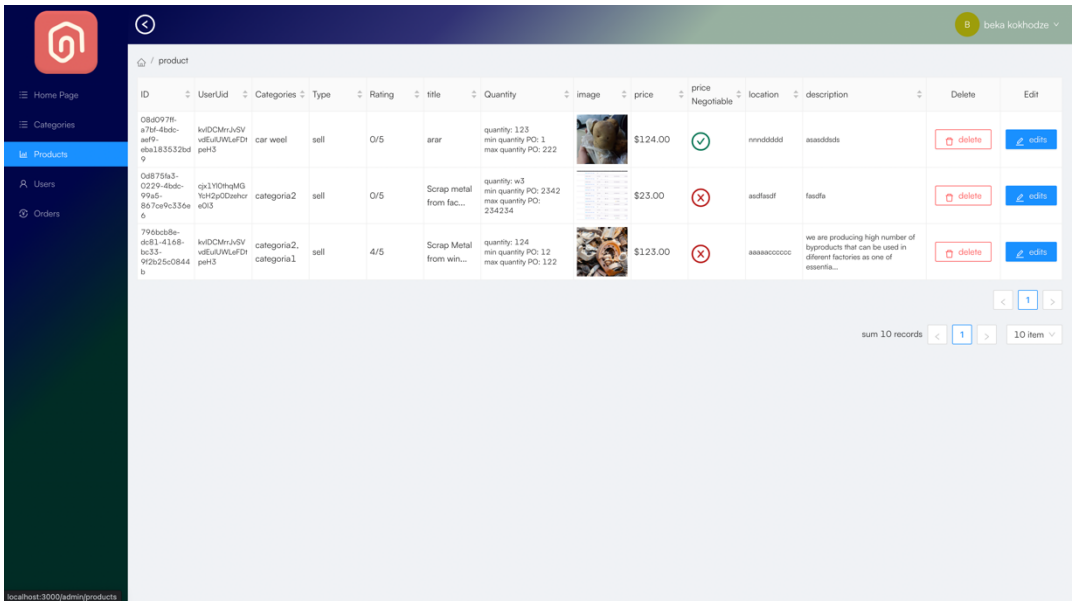


Figure 48:Page for managing all products.

The most important page for this platform is all products page, I'm showing all the necessary details and tools for properly purifying and maintaining the product. There

is a way to edit and delete product if the product doesn't follow the standards of product placement (most probably spam products).

4.10.4. Orders

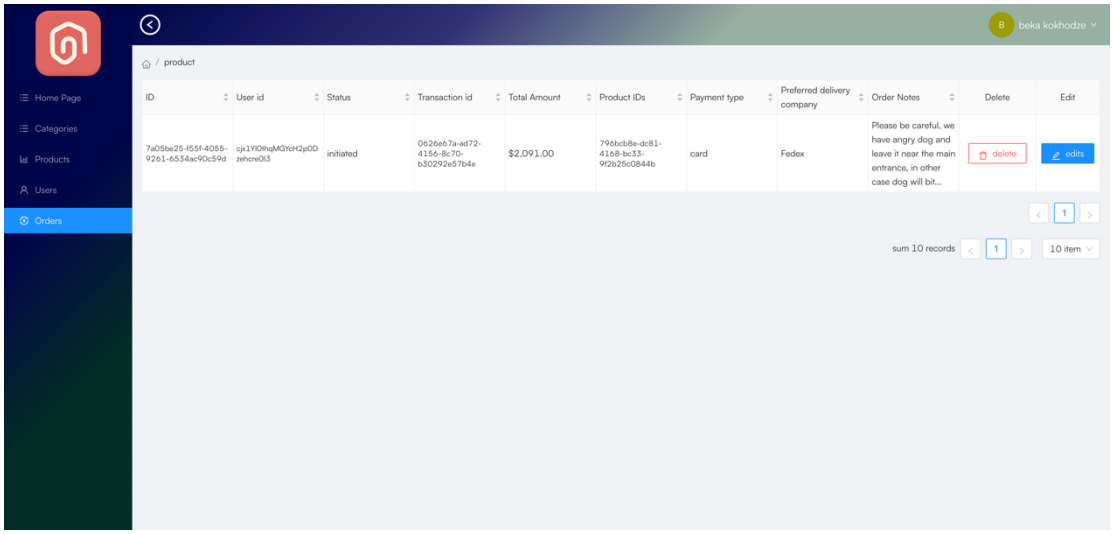


Figure 49:Page for orders management.

Sometimes orders are getting stuck or users want to change something. Because of that, those moderators and administrators need to be able to edit or delete any order, most of the time it will be the case for changing order notes.

4.11 Deployment

The main website's UI is deployed on Vercel using by Github's CI/CD pipelines that is automatically supported by Vercel and which gives us ability to preview every change in production with only pushing codes to the Git.

Chapter 5 Conclusions

Marketplace for Bioma was created in IPB to help and improve recyclability and support of the nature. With that in mind prototype of e-commerce platform was created which needs attaching it to real world environment and in initial step, researching for factories, manufacturers or people who will post products on our platform.

The methodology above adopted was divided into three parts: (1) modelling the system, (2) web platform architecture definition, (3) development of Marketplace for Circular Bioeconomy's web application and (4) system deployment in the production environment.

System modelling was the initial and most important step for creating the entire ecosystem of the platform, there is described use case diagrams, and class diagrams and I described 4 personas and dozens of functional requirements which were implemented accordingly.

With base information, general use case diagram was designed where was listed the personas' that played a key role in the platform and so after it, I was able to create a class diagram which helped me to construct the project from a codebase perspective and make the development process easier.

So, finally web app architecture is ready and we can implement it. Pattern for creating platform that was chosen is "one web server, one database", the simplest one indeed and most reliable pattern at the same time. Architecture is Server-side generation (SSG) and Server-side render (SSR) in combination with Single page application (SPA) which gives us great speed and google search index score. For authentication we use firebase SDK with browser cookies and JSON Web Token (JWT). JWT has only user id in it so it will be possible to search in database. It needs to be sent on every request to the server so that we can secure our database and we can identify the user.

For development process management Trello was used where sprints had duration of one week and mostly Kanban board was used to manage statuses of the task, it was integrated in git so it had possibility to have commit for every task on git.

As FrontEnd deployment server Vercel as used, which was super handy for debugging and checking production versions with its ability to deploy it on Versel's server and create preview links for Pull Request (PR) and every commit on git and give me ability to be more productive.

In the end we received functional product prototype which will help people and companies to contribute in helping our bioeconomy to get less prolusion as possible and reuse products maximally. It is fully featured e-commerce website where users can post their products or place a demand on a product so that both sides can be winner – seller and buyer and as a huge plus we are getting much better ecosystem.

There are some undone tasks that can be done in future works which is more about giving more flexibility to the all-user types. To add real payment and delivery companies and connect them to real delivery companies if seller or buyer wants it.

Final product can be access online from this link (https://bioma-project-ipb.vercel.app/), where everyone can register, place product, make orders and last but not least help circular bioeconomy and make environment healthy.

Bibliography

Continued Fashion: https://continued.fashion/

AgriMax project: https://agrimax.iris-eng.com

Wikipedia, AngularJS: https://en.wikipedia.org/wiki/AngularJS

Dayley, B. (2014): Node. js, MongoDB, and AngularJS web development. Addison-Wesley Professional.

Kumar, S. (2017): Differences Between Angular 1.x & Angular v2. https://www.c-sharpcorner.com/blogs/differences-between-angular-1x-angular-2

Nextjs: https://nextjs.org/

Youtube, Vue.js: The Documentary: https://youtube.com/watch?v=OrxmtDw4pVI/

Laravel Documentation: https://laravel.com/docs/9.x

Wikipedia, Firebase: https://en.wikipedia.org/wiki/Firebase

Firebase: https://firebase.google.com/

GitHub: https://github.com/

Vercel: https://vercel.com/

Trello: https://trello.com

EmailJS: https://www.emailjs.com

Google Map: https://www.google.com

Stefan Krause, 2021: https://github.com/krausest/js-framework-benchmark