

**Czech University of Life Sciences Prague**  
**Faculty of Economics and Management**  
**Department of Information Engineering**



**Diploma Thesis**

**Comparison of building an online store  
of digital technology using React and WordPress**

**Vasilenko Natalia**

**© 2022 CULS Prague**

# CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

## DIPLOMA THESIS ASSIGNMENT

Natalia Vasilenko

Systems Engineering and Informatics  
Informatics

Thesis title

**Comparison of building an online store of digital technology using React and WordPress**

---

### Objectives of thesis

The main objectives of the diploma are building an online store of digital technologies using React and WordPress and comparison of two build structures using React and WordPress. The differences in the principles of construction and the advantages and disadvantages of these two tools will be revealed. There will be also identified differences in the functioning and usability of the site in development of a backend and a user-friendly interface.

### Methodology

There will be followed standards of the software engineering such as UML and ISO/IEC 9126 software quality standard. An online store of digital technology and two information systems using React and WordPress will be created and evaluated from the focus on the usability, functioning, and maintainance.

**The proposed extent of the thesis**

80 – 140 pages

**Keywords**

React; WordPress; web IS; on-line store

---

**Recommended information sources**

EISENMAN, Bonnie. Learning react native: building native mobile apps with JavaScript. Second edition. Boston: O'Reilly, 2018. ISBN 1491989149

LEBENSOLD, Jonathan. React native cookbook: bringing the web to native platforms. Sebastopol: O'Reilly, 2018. ISBN 978-1-491-99384-2

WELLING, Luke & Laura THOMPSON. PHP and MySQL Web Development, Sams Publishing 2003, ISBN 80-86497-60-7



---

**Expected date of thesis defence**

2021/22 WS – FEM

**The Diploma Thesis Supervisor**

doc. Ing. Vojtěch Merunka, Ph.D.

**Supervising department**

Department of Information Engineering

Electronic approval: 19. 11. 2020

**Ing. Martin Pelikán, Ph.D.**

Head of department

Electronic approval: 19. 11. 2020

**Ing. Martin Pelikán, Ph.D.**

Dean

Prague on 31. 03. 2022

## **Declaration**

I declare that I have worked on my diploma thesis titled "Comparison of building an online store of digital technology using React and WordPress" by myself and I have used only the sources mentioned at the end of the thesis. As the author of the diploma thesis, I declare that the thesis does not break copyrights of any their person.

In Prague on: 30.03.2022

---

Vasilenko Natalia

## **Acknowledgement**

I would like to thank my supervisor Mr. Vojtěch Merunka for his advice, answers to all my questions, his experience, support and the time he gave me.

Nevertheless, many big thanks go to all my family and my friends for support during the whole time of the studies during and this difficult time.

# Comparison of building an online store of digital technology using React and WordPress

## Abstract

This thesis discusses a comparison between the open source content management system WordPress and React user interface tool.

This comparison is based on the implementation of two online stores of digital technology using React and WordPress. The six characteristics describing software quality are analyzed. The calculation of the COCOMO model for two stores is carried out, which makes it possible to calculate the cost of software development. UML diagrams necessary for high-quality development and modeling of business processes are presented. Conclusions are formed based on all the data.

The literature review discusses what are React and WordPress, describes the structure and architecture, describes the scheme of work, types of UML diagrams. The question of "What is MockAPI and Why is it needed" is revealed. The plan of stage-by-stage calculation of COCOMO and the calculation of Function point necessary for calculation of the parameter "Kilo of lines of code (KLOC)" involved in the COCOMO model are given. The ISO/IEC 9126 software quality standards and Database are described.

**Keywords:** React; WordPress; web IS; online store, COCOMO, Function point, UML, SQL, MySQL, Virtual DOM, JSX, HTML, CSS, MockAPI, ISO/IEC 9126, store on WordPress, store on React.

# Porovnání budování internetového obchodu digitální technologie pomocí React a WordPress

## Abstraktní

Tato práce pojednává o srovnání open source redakčního systému WordPress a nástroje uživatelského rozhraní React.

Toto srovnání je založeno na implementaci dvou internetových obchodů s digitální technologií pomocí React a WordPress. Je analyzováno šest charakteristik popisujících kvalitu softwaru. Provádí se výpočet modelu COCOMO pro dvě prodejny, který umožňuje spočítat náklady na vývoj softwaru. Prezentovány jsou UML diagramy nezbytné pro kvalitní vývoj a modelování podnikových procesů. Na základě všech údajů se tvoří závěry.

Literární přehled pojednává o tom, co jsou React a WordPress, popisuje strukturu a architekturu, popisuje schéma práce, typy UML diagramů. Otázka „Co je MockAPI a proč je potřeba“ je odhalena. Je uveden plán postupného výpočtu COCOMO a výpočet funkčního bodu nutného pro výpočet parametru "Kilo řádků kódu (KLOC)" zahrnutého v modelu COCOMO. Jsou popsány standardy kvality softwaru ISO/IEC 9126 a databáze.

**Klíčová slova:** Reagovat; WordPress; webový IS; internetový obchod, COCOMO, Function point, UML, SQL, MySQL, Virtual DOM, JSX, HTML, CSS, MockAPI, ISO/IEC 9126, obchod na WordPress, obchod na React.

# Table of contents

<b>1. Introduction</b> .....	10
<b>2. Objectives and Methodology</b> .....	11
<b>2.1 Objectives</b> .....	11
<b>2.2 Methodology</b> .....	11
<b>3. Literature review</b> .....	12
<b>3.1 What is WordPress</b> .....	12
<b>3.1.1 History</b> .....	12
<b>3.1.2 The scheme of work of WordPress</b> .....	13
<b>3.1.3 Difference between WordPress.com and WordPress.org</b> .....	15
<b>3.1.4 Wordpress file structure:</b> .....	15
<b>3.1.5 Instruments</b> .....	17
<b>3.2 What is React.js</b> .....	19
<b>3.2.1 History</b> .....	19
<b>3.2.2 Architecture</b> .....	20
<b>3.3 UML</b> .....	24
<b>3.3.1 History</b> .....	24
<b>3.3.2 Chart types</b> .....	25
<b>3.3.3 Structural Diagrams</b> .....	26
<b>3.3.4 Behavior diagram</b> .....	30
<b>3.3.5 Interaction diagram</b> .....	32
<b>3.4 MockAPI</b> .....	34
<b>3.5 Database</b> .....	34
<b>3.6 COCOMO</b> .....	37
<b>3.7 Function point (FP)</b> .....	39
<b>3.8 ISO/IEC 9126 software quality standards</b> .....	41
<b>3.8.1 Determination of integrated quality indicators</b> .....	42
<b>4. Practical Part</b> .....	46
<b>4.1 Store on React</b> .....	46
<b>4.1.1 Class diagram</b> .....	46
<b>4.1.2 Use case diagram</b> .....	47
<b>4.1.3 Sequence diagram</b> .....	48
<b>4.1.4 Activity diagram</b> .....	49
<b>4.2 Store on WordPress</b> .....	50
<b>4.2.1 Use case diagram</b> .....	50
<b>4.2.2 Activity diagram</b> .....	51



4.2.3 Sequence diagram.....	52
4.2.4 Database for store on WordPress.....	54
4.3 MockAPI for store on react.....	59
4.4 The process of creating an online store on React.....	63
4.4.1 Store description.....	68
4.5 The process of creating an online store on WordPress.....	71
4.5.1 Store description.....	73
4.6 Calculation of function point for store on wordpress.....	79
4.7 Calculation COCOMO.....	83
4.8 ISO/IEC 9126 software quality standards.....	84
5. Results and Discussion.....	88
6. Conclusion.....	89
7. References.....	90
8. Appendix.....	91

## **1. Introduction**

The information technology industry is currently experiencing rapid growth. It affects various areas of life. The world of development is expanding more and more every day. There are new libraries, programming languages, new website builders and various tools to help implement projects of various sizes. Some of these helpers are React and WordPress. WordPress and React are gaining worldwide recognition for their functionality and efficiency for application development. Developers have various controversies when thinking about which tool is best for implementing their project. This topic will be covered in this thesis.

Every day more and more online stores appear that allow you to make purchases day and night without leaving your home. In this regard, this diploma thesis will consider a comparison of React and WordPress based on the creation of two stores of digital technology.

## **2. Objectives and Methodology**

### **2.1 Objectives**

The main objectives of the diploma are building an online store of digital technologies using React and WordPress and comparison of two build structures using React and WordPress. The differences in the principles of construction and the advantages and disadvantages of these two tools will be revealed.

There will be also identified differences in the functioning and usability of the site in the development of a backend and a user-friendly interface.

### **2.2 Methodology**

There will be followed standards of software engineering such as UML and ISO/IEC 9126 software quality standards. An online store of digital technology and two information systems using React and WordPress will be created and evaluated from the focus on usability, functioning, and maintenance.

## **3. Literature review**

### **3.1 What is WordPress**

WordPress is an open-source Content management system (CMS) that allows you to create different types of websites, change their design and functionality, add and edit content without knowing the code.

#### **3.1.1 History**

In 2003, developers Mike Little and Matt Mullenweg decided to create a convenient blog hosting platform based on b2 / cafelog software. Matt's friend Christina Tremulet recommended that the platform be named WordPress. This is how the first version of WordPress 0.7 appeared on May 27, 2003.

WordPress 1.0 was released in January 2004: otherwise known as the "Davis" version. Mullenweg has a soft spot for great jazz musicians. He names all the updates after the great jazz musicians of the past and today.

In 2005, Matt Mullenweg created the Akismet comment spam filter for WordPress. This plugin remains one of the most popular anti-spam comments for blogs to this day.

A new Kubrick theme has been released.

Duke 2.0 was released in December 2005. This included a major overhaul of the dashboard and an increase in the speed and efficiency of posting and uploading images.

In 2008, the WordPress theme directory was launched.

June 2009 saw the next major update to WP: WordPress 2.8 Baker.

This update helped admins with features like automatic theme installation and the addition of a CodePress editor to help web developers code their sites was also important.

WordPress also won the Packt Best Open Source CMS Awards that same year.

The WordPress 3.1 Reinhardt update came out in February 2011.

One of the notable additions is the admin panel, which you can use to access the back end of each page while on your site.

Since December 2013, there has been a major change with the WordPress 3.8 Parker update.

For the first time since the Davis update, the core team has changed the look of the dashboard.

A responsive WordPress admin interface was also made.

This means that the platform can run on a mobile device or tablet as well as on a computer.

An update to WordPress 4.0, or Benny, has been released to polish the writing and management experience.

Uploading media files just got easier as they were all presented in a nice grid.

Improved search, new metrics, and a more visual experience have also been implemented to see exactly what types of plugins appear after a search.

WordPress 4.5 or Coleman has streamlined the workflow with add-ons like inline links.

Several other formatting shortcuts were provided, such as the ability to instantly add horizontal lines.

One of the most impressive updates was the preview button.

Since websites need to look great on all devices, it made sense to make the preview mobile responsive.

Custom logos came along with the update, and smart image resizing allowed images to load much faster without any changes to page speed.

The latest version of WordPress 4.9, Tipton, marks a significant step towards a more user-centric way of customizing and managing websites with big improvements in the Customizer, additional widget changes, a powerful WordPress text editor for code editing, and more.

All previous versions of WordPress did not have highlighting or code syntax checking.

WordPress 4.9 introduces CodeMirror text editor integration.

The last Tatum update was released on July 20, 2021. We have implemented widget management using the block editor, displaying posts with new blocks and patterns, editing templates for posts, a new file in theme.json templates, support for Internet Explorer 11 has ended, support for the WebP image format has been added, additional support for blocks has been added.

The project is constantly being improved and supplemented. It is supported by open-source - a community where participants from all over the world have united. Every two to three months, updated versions of the engine are released, the functionality is improved, and the security of the platform is increased.

### **3.1.2 The scheme of work of WordPress**

WordPress uses PHP and a SQL database to store all of its data. Each WordPress CMS uses one database. All information that is added to the site is stored in the WordPress database.

PHP is a server-side scripting language designed specifically for the Web. Within an HTML page, you can embed PHP code that will be executed each time the page is visited. Your PHP code is interpreted at the Web server and generates HTML or other output that the visitor will see.

PHP was conceived in 1994 and was originally the work of one man, Rasmus Lerdorf. It was adopted by other talented people and has gone through three major rewrites to bring us the broad, mature product we see today. As of January 2001, it was in use on nearly five million domains worldwide, and this number is growing rapidly. (Luke & Laura Thompson, 2003)

This includes:

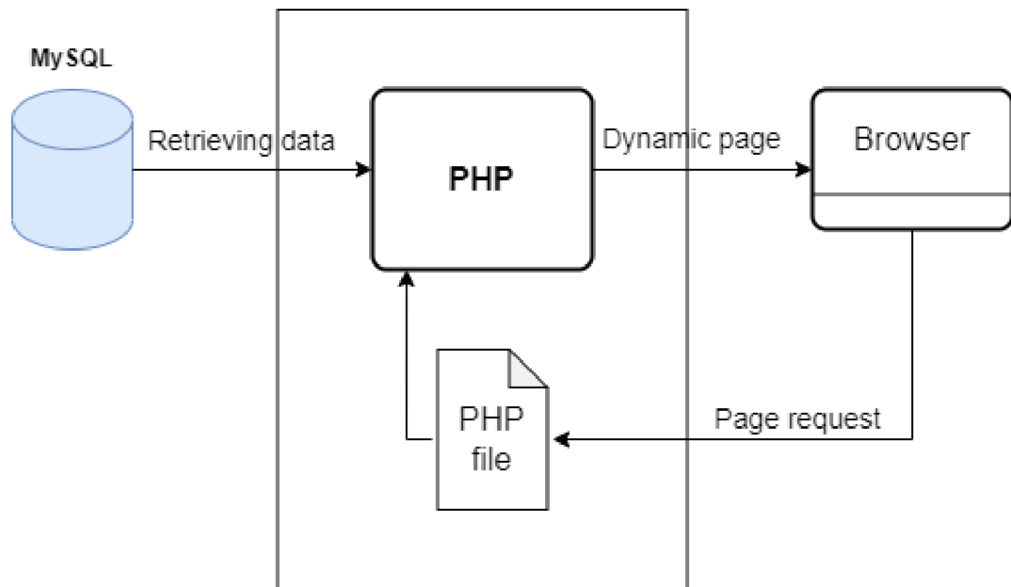
- User login, password (encrypted with MD5) email address, etc;
- All posts, pages, tags, categories, and links between them;
- Custom post types;
- Revisions, drafts, and records that have been deleted;
- Approved comments and pending moderation, spam;
- Theme customization options;
- Plugin data, etc.

Images, documents, and other uploads are not stored in the WordPress database. They are located in the "wp\_content" folder.

When a web page is requested, the following actions occur::

- The visitor's browser requests a web page.
- The WordPress core (can be thought of as the brain of WordPress) invokes the required PHP scripts starting at index.php.
- The WP core then connects to its database and retrieves data (posts, pages, comments, and other information).
- It then combines the extracted data, data from the currently active plugins and the currently active theme, and generates HTML code on the fly, that is, dynamically.
- It then serves this dynamically generated HTML to the visitor's browser.

Likewise, a post is published or saved, a comment is posted, a search is performed, WordPress core performs the necessary internal operations and stores them in its database for future use, and notifies the WordPress admin. The administrator sees in the control panel new comments awaiting moderation or a pool of comments that are immediately marked as spam.



*Figure 1 The scheme of work of WordPress*

### **3.1.3 Difference between WordPress.com and WordPress.org**

WordPress.com is a site owned by WordPress creator Matt Mullenweg. This site is a free service that allows you to create and host your WordPress site without thinking about paying for hosting, domain, etc. You can also purchase a separate domain name and link it to your site.

But there are also disadvantages:

- You have to pay to use your domain;
- Limited choice of topics and charging fees for changing them;
- Limited choice of plugins.

WordPress.org is a site where you can download the latest version of WordPress for installation on your hosting. This version is completely free and is customizable at the discretion of the client.

But there are also disadvantages:

- Need to pay for the domain name and hosting;
- Need to keep track of site updates and security;
- Need to regularly back up the site.

### **3.1.4 Wordpress file structure:**

The root directory contains the following folders and folder files:

wp-config.php - this php file contains the database name and password, encoding, table prefix, language, cache size, many other parameters can be added to the file.

The .htaccess file is an additional configuration file for the Apache web server and similar servers. Allows you to set a large number of additional parameters and permissions for the web server in separate directories.

wp-includes - core wordpress. The folder is overwritten with each update.

wp-admin - CSS, JavaScript and PHP files that power the admin console. The folder is overwritten with each update.

wp-content - contains custom folders and consists of folders:

- languages - contains engine translation files in .mo and .po formats
- plugins - installed plugins
- themes - installed templates, at least one template must be installed. May contain the following folders and files:
  - index.php - template for the main page of the site, also loads the sidebar file. Required file, at the root of the template folder
  - style.css - required file, responsible for css-styles of the template, in the root of the template folder
  - header.php - file responsible for outputting data in the <head> section and top menu
  - sidebar.php - the file is responsible for generating the side (additional) columns. Basically, there are headings, tags, banners.
  - footer.php - the file is responsible for displaying the footer, footer menu, copyrights and html tags
  - single.php - responsible for displaying individual posts.
  - page.php - responsible for displaying individual pages (for example, "Contacts", "About us", etc.)
  - archive.php - responsible for displaying the archive page records
  - category.php - generates a page that displays publications by category
  - tag.php - template of the page that displays the list of posts by tags
  - comments.php - the file describes the display of comments
  - functions.php - an additional file with PHP code, thanks to which you can enable or disable, add or remove certain functionality. Custom code is often added to this file if you need to improve something.
  - / css / - this folder may contain additional css files
  - / js / - folder with JavaScript files
  - / images / - the folder contains images embedded in the template



- / languages / - the folder contains the theme translation files
- uploads - media files: images, music, documents, etc.

Template tags in WordPress:

Template tags are PHP functions in WordPress to display information or customize your blog, for example

`wp_list_pages ()` - displays the list of pages as links.

It is more convenient to read the documentation on the official WordPress site using template tags.

In WordPress itself, template tags are described in the following files:

`wp-includes / author-template.php` - template tags associated with the author

`wp-includes / bookmark-template.php` - template tags associated with bookmarks

`wp-includes / category-template.php` - template tags about all terms and taxonomy, including categories and tags

`wp-includes / comment-template.php` - file for template tags of the comment department

`wp-includes / link-template.php` - template tags for links (permalinks, attachment links, archive links, etc.)

`wp-includes / nav-menu-template.php` - template tags for the navigation menu

`wp-includes / post-template.php` - template tags associated with posts

`wp-includes / post-thumbnail-template.php` - file for template tags associated with post thumbnails

`wp-includes / general-template.php` - file for other template tags that can be used anywhere

### **3.1.5 Instruments**

Content. These are text, pictures, article titles, tags, categories, article descriptions, and various metadata. All this is stored in the database.

Theme. Apart from the content, there is a theme - this is similar to a pre-designed mini program that is responsible for displaying content.

Widgets

They allow you to display some blocks of information in places specially created for this. For example, in the sidebar (sidebar on the page. Can be on the left or right) or to the footer.

This can be a list of recent posts, a contact form, headline, and text, banner, gallery or video.

The key thing in widgets is that they can be displayed only in specially designated places.

In each theme, these places can differ both visually and quantitatively. For example, in some topics, only one block is displayed in the basement. In the other, there will be 3 of them.

Plugins. These are mini programs that are responsible for some special behavior: for example, comments on the site, pop-ups with a subscription to a newsletter, caching or displaying quotes of great people on the home page. A separate class of plugins is content editors: they allow you to typeset complex pages with special effects, beautiful buttons, animations, and multi-column content.

### Required plugins

Mandatory plugins solve various technical problems, without which a good website is impossible.

#### Yoast SEO

This plugin helps to customize the operation of the web resource according to the requirements of Google. Helps to create a sitemap (it contains links to all pages of the site).

Enables or disables breadcrumbs (additional navigation on the page. Mandatory for online stores).

Helps to write a keyword, title, and description for each page. In addition, customize the appearance in search results, social networks Facebook and Twitter (if users shared the link).

#### Autooptimize

This plugin is an accelerator. Its task is to compress HTML, CSS, and javascript codes. Move some scripts to the bottom of the page so that they do not interfere with the initial loading.

It compresses images on pages, makes them lighter.

### Plugins for specific tasks

Depending on the type of project, there may be a need for these plugins:

#### Woo commerce

The plugin allows you to deploy a full-fledged online store on WordPress. With its help, we manage goods, their descriptions, leftovers, display or hide goods, define them in a category, set a cost.

#### Wpbakery

This plugin is a visual page builder. Thanks to him, you can arrange the elements on the page as needed.

Elementor

Another page builder, but with a nicer and cleaner interface.

BuddyPress

Turns the site into a social network, where users can send each other a request and become friends, send private messages, write on a shared wall.

Masterstudy

LMS plugin. What does learn management system mean. Helps to build a fully-fledged online school. Organize content into courses, lessons. There is a function with tests, homework. You can set access to a single course, a set of courses, or access by a membership fee (monthly payment with access to all content).

Loco translate

The plugin allows you to translate a topic into any language.

WPML

Helps to translate into multiple languages.

## **3.2 What is React.js**

The authors (Alex Banks, 2017) say: “React is a popular library used to create user interfaces. It was built at Facebook to address some of the challenges associated with large-scale, data-driven websites”

### **3.2.1 History**

React was invented by one of the Facebook developers, namely Jordan Valke, it was he who created React and first used it in the Facebook news feed in 2011, a little later this technology was implemented in the Instagram feed

At the very beginning, it was developed as a closed technology, only for Facebook developers. However, at one of the conferences in 2013, his code became completely open, and React became available to every programmer.

Two years later, the React Native platform was already created, which allows you to create native mobile applications, on the already familiar React.

Two years later, Facebook announced the creation of React Fiber, which should be a more improved and optimized version of the familiar React, but today the React Fiber platform exists and makes life easier for developers.

### 3.2.2 Architecture

Components are reusable code. In the components are transferred repeating blocks. In HTML, we edit all blocks, even if they are repetitive. In React, we transfer repeating blocks to a component and only edit the component. The place we edited changes throughout the site where this component is located. The components allow you to save time, there is no confusion in the code, the code writing becomes clear. The code is being optimized and shortened.

Each React component will live in its own file. These components are usually presentational components, meaning that they can be used without any knowledge of an external dependency. React applications assume that it is the component's responsibility to declare what it needs from its consumer. A simple component could just be a single JavaScript file in the components/ folder. (Jonathan Lebensold, 2018)

Component life cycle:

1. First, a React.js template is defined to create elements from a component.
2. It is indicated where it will be used. For example, inside a call to the render function of another component or using "ReactDOM.render".
3. React creates an instance of the element and passes it a set of properties (props), which will be accessible through "this.props". These properties are what we passed in the second step.
4. Since this is JavaScript, the class constructor method (if defined) will be called. This is the first of the methods, which are called component lifecycle methods.
5. React handles the result of calling the render function.
6. Then React will mount the component: interacting with the browser through the DOM API, React will render.
7. Next, React calls another lifecycle method called "componentDidMount". This method can be used to do something in the document tree. The entire DOM that we have been working with previously was virtual.
8. Dismantling. The life cycle of some components ends already at this stage. Components can be removed from a document for a variety of reasons. However, before that React calls another method - "componentWillUnmount".

9. A component may need to be re-rendered if its state is updated or if its parent changes its properties.
10. If properties have changed, React.js will call the "componentWillReceiveProps" lifecycle method.
11. If the object or its properties have changed, React.js calls another method, "shouldComponentUpdate", which is essentially a question. So, if there is a need to customize the rendering process yourself, you can answer this question by returning "true or false".
12. If "shouldComponentUpdate" is not declared, React will call the unconditional "componentWillUpdate" and calculate the differences between the current component and its new look, taking into account the changes.
13. If no changes are committed, React.js will do nothing.
14. If there is a difference, the framework will render the component.
15. Since the update process has taken place anyway, React will call the "componentDidUpdate" method.

#### Component states:

Props represents data that comes from outside.

State describes the internal state of the component. Also, a significant difference between state and props is that the values in "State" can and should be changed.

An important point is that the values from "State" must be used when rendering. If some property of the state object is not used in the render, there is no point in storing it in "State". The `setState ()` method is used to update the state.

Components can be functional or class-based.

Class components are written based on ES6 JavaScript classes.

The functional component is named this way because we are declaring a function that returns JSX markup.

JSX markup is the same HTML only in it you can still write JavaScript logic. JSX represents a combination of JavaScript and XML and provides a simple and intuitive way to describe the structure of an interface.

One of the main advantages of JSX is native error notifications and warnings, which help you quickly learn about situations that are worth paying attention to.

Every element in the JSX markup for React is an object.

React takes an array of objects and converts them to Dom elements.

**Virtual Dom** is a collection of object sets (object tree).

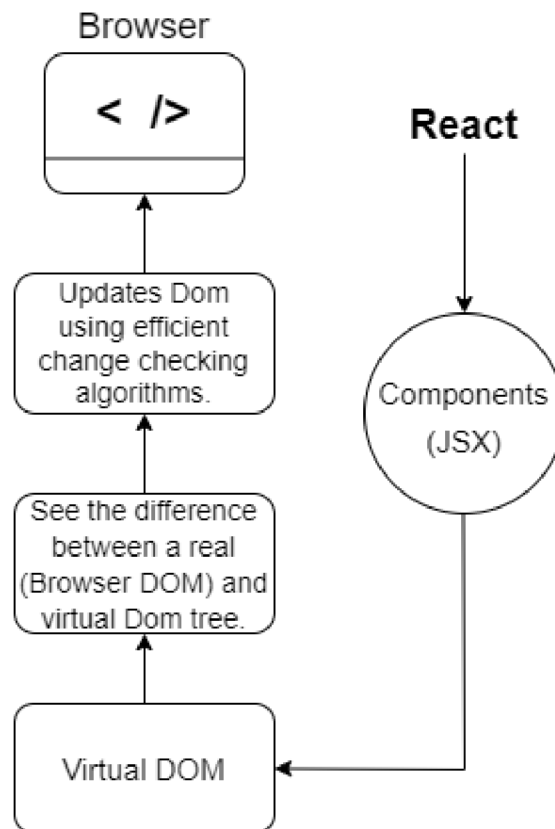
In React, the Virtual DOM acts as a layer between the developer's description of how things ought to look, and the work done to actually render your application onto the page. To render interactive user interfaces in a browser, developers must edit the browser's DOM, or Document Object Model. This is an expensive step, and excessive writes to the DOM have a significant impact on performance. Rather than directly render changes on the page, React computes the necessary changes by using an in-memory version of the DOM, and rerenders the minimal amount necessary. (Bonnie Eisenman, 2018)

If elements of a web page need to be changed, the changes are first made to the Virtual DOM. Then the new state of the Virtual DOM is compared with the current state. And if these states are different, then React finds the minimum amount of manipulation that is necessary before updating the real DOM to a new state and performing them.

As a result, this scheme of interaction with elements of a web page works much faster and more efficiently than if we worked directly from JavaScript with the DOM.

For example, after a user submits data in a form to the server, React detects a state change and updates the interface's appearance.

Thus, the Virtual DOM allows you to update the real DOM and change the state of components on the screen without delay. The user does not see blinking and other visual artifacts. It allows reducing the load on displaying some data on the page.



*Figure 2 The scheme work React*

Hook is a javascript function that allows you to create/access the state and lifecycles of React.

Before hooks were introduced, functional components did not have the ability to specify a local state. The situation changed with the introduction of `useState ()`.

Since the ultimate goal of hooks is to simplify the current logic, React provides only a reduced set of them, while flexibly reacting to various situations in the application life cycle and allowing you to create your own.

React provides three main hooks that allow you to solve the main tasks of implementing a lifecycle in a class component:

UseState state hook:

This hook returns the value with the saved state and the function needed to update it:

```
const [count, setCount] = useState(0)
```

The initial state is the parameter passed to `useState`, in this case 0, and this will be the state available during initial rendering and before calling `setCount` with the new value. In this case, for example, `setCount (count + 1)` will increment count by one, which will become 1 on the next render. You

can also use the previous state value in the set state function itself, so the above can also be written as `setCount (count => count + 1)`.

UseEffect hook:

UseEffect: designed to intercept various kinds of changes in components that cannot be processed inside components.

This hook allows us to add additional effects to a given functional component, that is, it allows us to make changes to our logic after rendering, similar to the `componentDidMount`, `componentDidUpdate` and `componentWillUnmount` lifecycle methods in class components.

By default, effects run after every redraw. But you can make them run only after changing the values of specific variables by passing them as the second optional parameter as an array.

UseContext context hook:

Context in React is a way to pass data between different components without having to manually cascade props. This is useful, for example, when we want to create themes or localizations that need to be global to the entire component tree.

In the case of class components, the context is passed through a provider that spans the component tree.

## **3.3 UML**

In the book (Sinan Si Alhir, 2003), there is information that UML is a visual language for modeling and communicating about systems through the use of diagrams and supporting text.

### **3.3.1 History**

In the not too distant 80s, there were many different modeling methodologies. Each of them had its own advantages and disadvantages, as well as its own notation. That troubled time was called the "war of methods". The problem is that different people used different notations, and in order to understand what a particular diagram describes, a "translator" was often required. The same symbol could mean completely different things in different notations! In the figure below, you can see only a small part of the variety of methods that existed at that time and to some extent influenced the UML.

In addition, around the same time (the early 1980s), the "object-oriented era" kicked off. It all started with the emergence of the SmallTalk family of programming languages that used some of the concepts of the Simula-67 language used in the 60s. The emergence of the object-oriented



approach was primarily due to the increase in the complexity of tasks. The object-oriented approach has made quite radical changes in the very principles of creating and operating programs, but, at the same time, it has made it possible to significantly increase the productivity of programmers, take a different look at problems and methods of solving them, and make programs more compact and easily extensible. As a result, languages originally oriented towards the traditional approach to programming have received a number of object-oriented extensions. One of the first, in the mid-80s, was Apple with its Object Pascal project. In addition, the object-oriented approach has spawned a powerful wave of completely new software technologies, culminating in such widely recognized platforms as the Microsoft .NET Framework and Sun Java.

But most importantly, the emergence of OOP required a convenient tool for modeling, a unified notation for describing complex software systems. And so the "three amigos", three leading specialists, three authors of the most popular methods decided to combine their developments. In 1991, each of the "three amigos" began by writing a book in which he outlined his OOAP method. Each methodology was good in its own way, but each also had its drawbacks. So, Booch's method was good at design, but weak at analysis. Rumbach's OMT was, on the other hand, a great analysis tool, but bad at design. Finally, Jacobson's Objectory was really good in terms of user experience, which neither Booch's method nor the OMT paid much attention to. The main idea behind Objectory was that analysis should start with use cases, not with a diagram of the classes that should be derived from them.

By 1994, there were 72 methods or private methods. Many of them "overlapped", that is, they used similar ideas, notations, etc. was created in the field of modeling.

In 1995, the main role in organizing the UML development process was transferred to the OMG (Object Management Group) consortium. The OMG development group, which also included Booch, Rambeau, and Jacobson (the "three amigos"), released the UML specifications 0.9 and 0.91 in June and October 1996.

### **3.3.2 Chart types**

There are two main types of diagrams used in the UML:

- Structural diagrams
- Behaviour diagrams

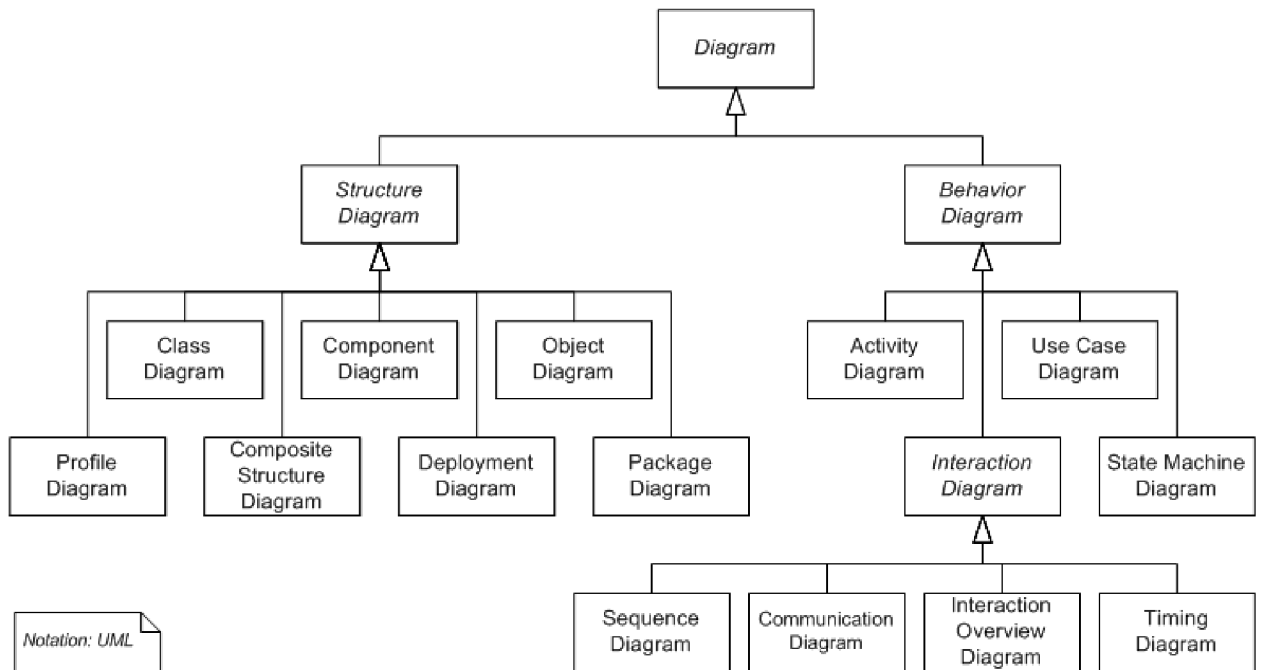


Figure 3 Structural diagrams and Behaviour diagrams

### 3.3.3 Structural Diagrams

#### Class Diagram

The authors (Martin Fowler, 2003) say: " A class diagram describes the types of objects in the system and the various kinds of static relationships that exist among them. Class diagrams also show the properties and operations of a class and the constraints that apply to the way objects are connected. The UML uses the term feature as a general term that covers properties and operations of a class."

#### **Chart elements**

The class in the diagram is depicted as a rectangle divided by horizontal lines into three parts. The first part contains the name of the class. Typically, a class name consists of one, maximum two words. The second part contains a list of class attributes that characterize this or that object of this class in the domain model. The third part contains a list of operations that reflect its behavior in the domain model.

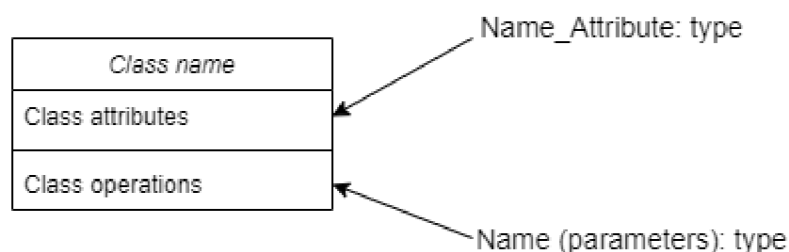


Figure 4 Chart elements

## Visibility

+ public - open access

- private - only from operations of the same class

# protected - only from operations of the same class and classes created on its basis

If an attribute or operation is described with the private modifier, then it can only be accessed from an operation defined in the same class. If an attribute or operation is described with the public visibility modifier, then it can be accessed from any part of the program. The protected modifier allows access only from operations of the same class and classes created on its basis.

## Relationships

**Addiction.** Dependency occurs when the implementation of a class of one object depends on the specification of the operations of the class of another object. And if the specification of the operations of this class changes, you need to make changes to the dependent class as well.

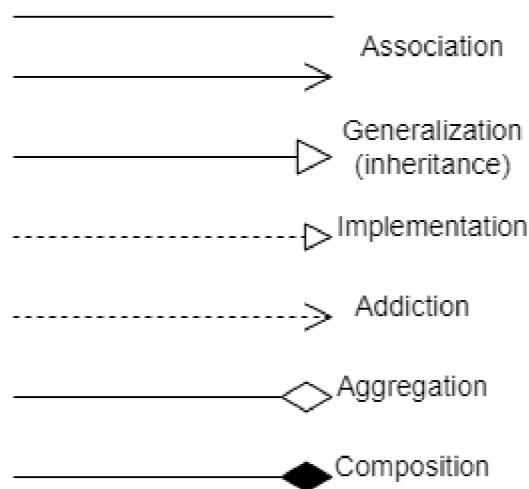
**Association.** This is the connection between objects, along which you can move between them. An association can have a name that indicates the nature of the relationship between objects, and the name can indicate the direction of reading the relationship using a triangular marker. An arrow can represent a unidirectional association.

**Aggregation** is a part-to-whole relationship. In this case, one class has a higher status (whole) and consists of lower status classes (parts). At the same time, simple and composite aggregation are distinguished. Simple aggregation assumes that parts separated from the whole can continue to exist independently of it. **Composite** aggregation is understood as a situation when the whole owns its parts and their lifetime corresponds to the lifetime of the whole, i.e., parts cannot exist independently of the whole.

**Generalization** is the relationship between a more general entity called a superclass and a specific incarnation called a subclass. Sometimes a generalization is called a relationship of the "is" type, meaning that some entities (for example, a circle, a square, a triangle) are the embodiment of a more general entity (for example, the class "geometric figure"). Moreover, all the attributes and operations of the superclass, regardless of the visibility modifiers, are part of the subclass. Generalization (or, as is often said, inheritance) in diagrams is denoted very simply - an open triangular arrow pointing at the superclass.

In UML modelling, a **realization** relationship is a relationship between two model elements, in which one model element (the client) realizes (implements or executes) the behavior that the other model element (the supplier) specifies. A provider is usually an abstract class or an interface class.

The UML graphical representation of a Realization is a hollow triangle shape on the interface end of the dashed line (or tree of lines) that connects it to one or more implementers. A plain arrow head is used on the interface end of the dashed line that connects it to its users. In component diagrams, the ball-and-socket graphic convention is used (implementors expose a ball or lollipop, whereas users show a socket). Realizations can only be shown on class or component diagrams.(Wikipedia, 2009)



*Figure 5 Notation relationships*

## Multiplicity

The multiplicity of a property is an indication of how many objects may fill the property. The most common multiplicities you will see are

- 1 (An order must have exactly one customer.)
- 0..1 (A corporate customer may or may not have a single sales rep.)
- \* (A customer need not place an Order and there is no upper limit to the number of Orders a Customer may place—zero or more orders.)

More generally, multiplicities are defined with a lower bound and an upper bound, such as 2..4 for players of a game of canasta. The lower bound may be any positive number or zero; the upper is any positive number or \* (for unlimited). If the lower and upper bounds are the same, you can use one number; hence, 1 is equivalent to 1..1. Because it's a common case, \* is short for 0..\*.

## Object diagram

An object is an instance of a class.

An object diagram shows many objects - instances of classes (depicted in a class diagram) and the relationship between them at some point in time. That is, an object diagram is a kind of snapshot of the state of the system at a certain point in time, showing a set of objects, their states and the relationship between them at a given moment.

Thus, object diagrams represent a static view of the system in terms of design and processes, being the basis for the scenarios described by interaction diagrams. In other words, an object diagram is used to explain and drill down on interaction diagrams such as sequence diagrams.

## Deployment diagram

The deployment diagram shows the topology of the system and the distribution of system components among its nodes, as well as the connections - information transfer routes between hardware nodes. This is the only diagram that uses "three-dimensional" notation: the nodes of the system are indicated by cubes.

## Component diagram

A component diagram describes the features of the physical representation of a system. A component diagram allows you to determine the architecture of the system under development by establishing dependencies between software components, which can be source, binary and executable code. In many development environments, a module or component corresponds to a file. Dotted arrows connecting modules show interdependent relationships.

The main graphic elements of the component diagram are components, interfaces and dependencies between them.

A component diagram is designed for the following purposes:

- visualization of the general structure of the source code of the software system;
- specification of the executable version of the software system;
- ensuring the reuse of separate fragments of the program code;
- presentation of conceptual and physical database schemas.

## Package diagram

A package diagram is a diagram that contains packages of classes and the dependencies between them.

A package is a UML construct for organizing UML models and also for grouping classes.

Packages communicate with each other by a special relationship - dependence. This is a directed relationship, and it goes from the package that depends to the package that is needed by the first dependent. This means that the package being used contains a description of the constructs that the dependent package imports, rather than implements itself. Dependency is not limited to package diagrams, but can be used to link other UML constructs, for example, can link two use cases: one can depend on the other.

#### Composite structure diagram

A diagram that depicts the internal structure of a classifier such as a class, component, or cooperation, including the points of interaction of the classifier with other parts of the system.

Internal structure - the structure of interacting model elements that are created in an instance of the classifier containing them.

A property is a set of instances that are the property of the containing classifier instance.

### **3.3.4 Behavior diagram**

#### Use case

A use case diagram is a type of UML behavioral diagram that is often used to analyze various systems. They allow you to visualize the different types of roles in the system and how these roles interact with the system.

#### Use Case Diagram Objects

The corpus diagrams used are composed of 4 objects.

- Actor
- Use case
- System
- Package

#### Actor

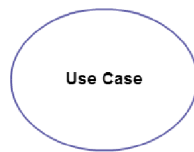
An actor uses a use case diagram - any entity that plays a role in one given system. It can be a person, an organization, or an external system and is usually drawn like the skeleton shown below.



*Figure 6 Actor*

### Use case

A use case represents a function or action within a system. It is drawn as an oval and named function.



*Figure 7 Use case*

### System

The system is used to define the scope and is drawn as a rectangle. This is an optional element, but useful when rendering large systems. For example, you can create all use cases and then use the system object to define the scope of your project. Or you can even use it to show different areas covered in different releases.



*Figure 8 System*

### Package

The package is another add on that is extremely useful in complex diagrams. Similar to class diagrams, packages are used to group use cases. They are drawn as shown in the picture below.



*Figure 9 Package name*

Communication Association

Shown as a solid line from an actor to use case indicates that the actor is applying that use case.

### **3.3.5 Interaction diagram**

An interaction diagram is a diagram that represents an interaction that consists of many objects and the relationships between them, including the messages that they exchange. This term applies to types of diagrams with an emphasis on the interaction of objects (cooperation, sequence, interaction overview and timing diagrams).

#### Sequence diagram

A sequence diagram shows the interaction of objects over time.

The sequence diagram displays the temporal features of the transmission and reception of messages by objects, the life cycle of the object and the interaction of actors (actors) of the information system within the use case.

#### Designations

Objects are denoted by rectangles with underlined names (to distinguish them from classes), messages (method calls) by lines with arrows, and results returned by dashed lines with arrows. The rectangles on the vertical lines below each of the objects show the "lifetime" (focus) of the objects, but quite often they are not depicted in the diagram.



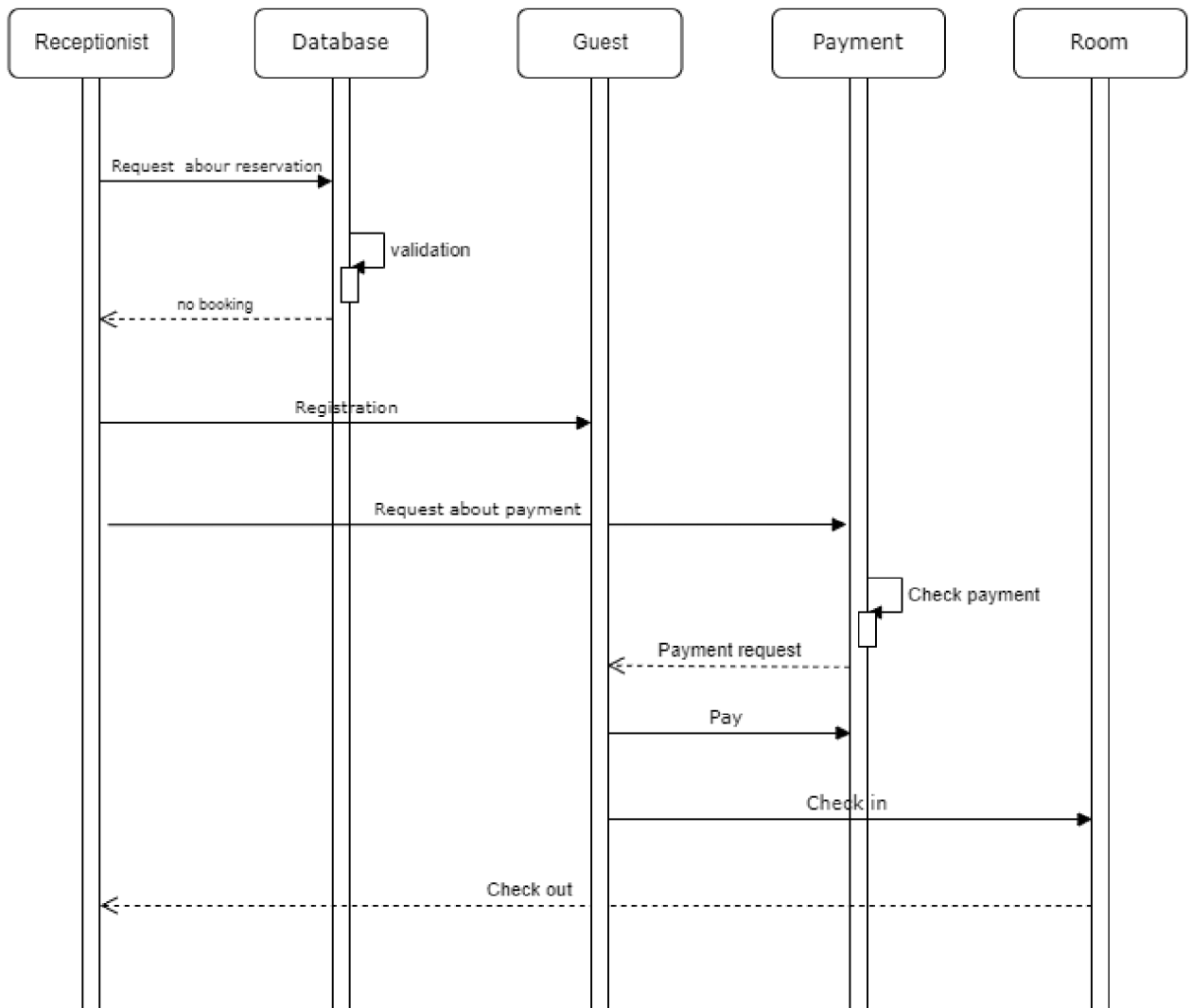


Figure 10 Example diagram

Cooperation / Communication diagram is one of the subtypes of interaction diagram, in which the main emphasis is on the structural organization of objects that send and receive messages. This is an analog of a sequence diagram, which also shows the exchange of messages between objects, but focuses on the roles that objects play in interaction.

### State diagram

State diagrams are used to explain how complex objects work.

A state (state) is a situation in the life cycle of an object during which it satisfies a certain condition, performs a certain activity, or waits for an event. The state of an object is determined by the values of some of its attributes and the presence or absence of links with other objects.

A state diagram shows how an object transitions from one state to another.

A state diagram is useful in modeling the life cycle of an object.

Designations

The rounded rectangles represent the states that an object passes through during its life cycle. The arrows show the transitions between states that are caused by the execution of the methods of the object described by the diagram. There are also two types of pseudo states: the initial, in which the object is located immediately after its creation (indicated by a solid circle), and the final, which the object cannot leave if passed into it (indicated by a circle encircled by a circle).

### Activity diagram

Activity Diagrams are a representation of the algorithms of certain actions (activities) running in the system.

Activity diagrams allow you to model the complex life cycle of an object, with transitions from one state (activity) to another.

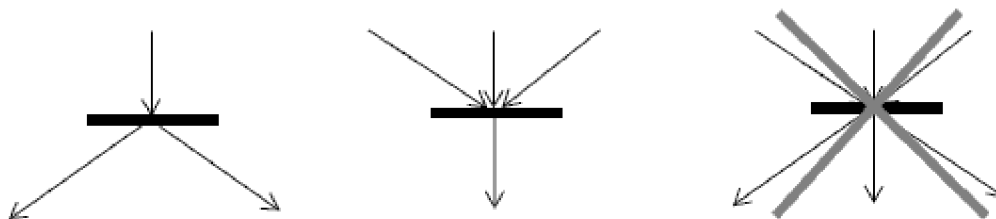
### Designations

Actions are shown with rounded rectangles.

Rhombus symbol of decision making with designations of conditions near the transitions.

The beginning is a filled circle, and the end is shown as a filled circle within a circle.

Parallelization symbol.



*Figure 11 Designations*

## 3.4 MockAPI

MockAPI is a simple tool that lets you easily mock up APIs, generate custom data, and perform operations on it using RESTful interface. MockAPI is meant to be used as prototyping/testing/learning tool. (MockAPI, 2022)

A MockAPI is used instead of a backend.

It can be said that this is a stub to a real API, which will hypothetically be implemented soon.

The MockAPI returns the "fake" data necessary for front-end developers to write the front-end of the application while the back-end developers are developing the back-end.

## 3.5 Database

A database is a set of data that shows the state of objects and the relationship between them.

A database management system (DBMS) is a system for creating databases and for querying, sorting, and searching data. Databases are divided into a large number of varieties.

WordPress uses an SQL database to store all of its data.

Wordpress uses this type of database as relational. For this period, there are several variants of DBMS such as Oracle Database, MySQL, Access, SQL Server, Fox Pro and others.

MySQL (pronounced My-Ess-Que-Ell) is a very fast, robust, relational database management system (RDBMS). A database enables you to efficiently store, search, sort, and retrieve data. The MySQL server controls access to your data to ensure that multiple users can work with it concurrently, to provide fast access to it, and ensure that only authorized users can obtain access. Hence, MySQL is a multi-user, multi-threaded server. It uses SQL (Structured Query Language), the standard database query language worldwide. MySQL has been publicly available since 1996, but has a development history going back to 1979. It has now won the Linux Journal Readers' Choice Award three years running.

MySQL is now available under an Open Source license, but commercial licenses are also available if required. (Luke & Laura Thompson, 2003)

It is very easy to develop and easy to learn. Basic operations performed in MySQL:

- Creation of a database;
- Creation of tables;
- Adding, editing and deleting records;
- Sorting;
- Creation and execution of requests;
- Building relationships between tables.

When creating tables, you must specify the data type.

Supported data types:

TINYINT: represents integers from -128 to 127, occupies 1 byte.

SMALLINT: represents integers from -32768 to 32767, occupies 2 bytes.

MEDIUMINT: represents integers from -8388608 to 8388607, occupies 3 bytes.

BIGINT: represents integers from -9 223 372 036 854 775 808 to 9 223 372 036 854 775 807, occupies 8 bytes.

FLOAT: Single precision signed floating point number.

DOUBLE, DOUBLE PRECISION, REAL: signed double precision floating point number.

DECIMAL, NUMERIC: signed floating point number.

DATE: date.

DATETIME: combination of date and time.

TIMESTAMP: timestamp.

TIME: time.

YEAR: Year in YY or YYYY format.

CHAR: A fixed-length string, right-padded with spaces to the maximum length.

VARCHAR: Variable length string.

TINYTEXT: A string with a maximum length of 255 characters.

BLOB, TEXT: stores binary data as a string up to 65 KB in length.

MEDIUMBLOB, MEDIUMTEXT: Stores binary data as a string up to 16MB in length.

ENUM: enumeration.

SET: sets.

INT or INTEGER: represents integers from -2147483648 to 2147483647, occupies 4 bytes. Used when creating PRIMARY and FOREIGN KEY. Each component in the tables has its own Id, for which such a data type as INT is indicated. With the help of Id, a connection is made with other tables, if necessary.

The data type is specified after the column name. Also, NULL is specified for each column. There are two conditions:

- NULL means that this column can store empty values.
- NOT NULL - means that this column cannot store null values.

In addition, operators are used to perform various operations when writing code. As you know, each of the operators is responsible for a specific function and must be written in capital letters.

They are divided into:

Data Definition Language (DDL) operators:

CREATE creates a database object (database, table, view, user, and so on),

ALTER changes an object,

DROP deletes an object;

Data manipulation operators (Data Manipulation Language, DML):

SELECT selects data that satisfies the given conditions,

INSERT adds new data,

UPDATE changes existing data,

DELETE deletes data;

Data access definition statements (Data Control Language, DCL):

GRANT grants a user (group) permissions to perform certain operations on an object,

REVOKE revokes previously issued permissions,

DENY sets a ban that takes precedence over a permit;

Transaction Control Language (TCL) statements:

COMMIT applies a transaction

ROLLBACK rolls back all changes made in the context of the current transaction,

SAVEPOINT divides the transaction into smaller sections.

### 3.6 COCOMO

COCOMO is a procedural software cost estimation model proposed by Barry W. Boehm in 1981. This cost estimation model is extensively used in predicting the effort, development time, average team size and effort required to develop a software project. The distinctiveness of this strategy is that COCOMO estimates the cost based on the number of lines of source code and sets of subjective assessment (cost drivers) of product, hardware, personnel and project attributes. This provides transparency to the model which allows software managers to understand why the model gives the estimates it does. Moreover, the baseline COCOMO originally underlies a waterfall model lifecycle. (Warakorn Jetlohasiri, 2020)

In the COCOMO model, projects are divided into three categories:

1. Organic - This category is typical for small teams with average work experience.
2. Semi-detached - This category is typical for teams with mixed backgrounds, projects are relatively less familiar and difficult to develop than organic projects.
3. Embedded - This category is typical for projects with a high level of complexity and large teams with extensive experience.

Model types:

Basic COCOMO model is based on an estimate of the time and effort involved depending on the estimated size of the software being created, expressed in lines of source code. It is suitable for a quick cost estimate and has some inaccuracies due to some factors that cannot be taken into account.

Formula:

$$\text{Effort} = a * (\text{KLOC})^b$$

$$\text{Time} = c * (\text{Effort})^d$$

$$\text{Personrequired} = \text{Effort} / \text{Time}$$

Intermediate COCOMO II computes software development effort as function of program size and a set of "cost drivers" that include subjective assessment of product, hardware, personnel and project attributes. This extension considers a set of four "cost drivers", each with a number of subsidiary attributes:

Product attributes:

- Required software reliability extent

- Size of application database
- Complexity of the product

Hardware attributes:

- Run-time performance constraints
- Memory constraints
- Volatility of the virtual machine environment
- Required turnabout time

Personnel attributes:

- Analyst capability
- Software engineering capability
- Applications experience
- Virtual machine experience
- Programming language experience

Project attributes:

- Use of software tools
- Application of software engineering methods
- Required development schedule. ( Wikipedia, 2015)

Formula:

$$E = a * (KLoC)^b (EAF)$$

“Detailed COCOMO incorporates all characteristics of the intermediate version with an assessment of the cost driver’s impact on each step of the software engineering process. The detailed model uses different effort multipliers for each cost driver attribute. In detailed cocomo, the whole software is divided into different modules and then we apply COCOMO in different modules to estimate effort and then sum the effort.

The Six phases of detailed COCOMO are:

1. Planning and requirements
2. System design
3. Detailed design
4. Module code and test

5. Integration and test

6. Cost Constructive model

The effort is calculated as a function of program size and a set of cost drivers are given according to each phase of the software lifecycle.” (GeeksforGeeks, 2022)

### **3.7 Function point (FP)**

The function point method was first proposed in 1983 by Allan Albrecht and is currently the main technology for estimating the functional size of both ready-made and software tools under design.

Calculation Function point (FP):

The value adjustment factor (VAF) is based on 14 general system characteristics (GSC's) that rate the general functionality of the application being counted. Each characteristic has associated descriptions to determine the degrees of influence.

The degrees of influence range on a scale of zero to five, from no influence to strong influence. (Pratt, 2022)

Formula:

$$\text{VAF} = (\text{TDI} \times 0.01) + 0.65$$

Fourteen general system characteristics:

General System Characteristic		Brief Description
1	Data communications	How many communication facilities are there to aid in the transfer or exchange of information with the application or system?
2	Distributed data processing	How are distributed data and processing functions handled?
3	Performance	Did the user require response time or throughput?
4	Heavily used configuration	How heavily used is the current hardware platform where the application will be executed?
5	Transaction rate	How frequently are transactions executed daily, weekly, monthly, etc.?
6	On-Line data entry	What percentage of the information is entered On-Line?
7	End-user efficiency	Was the application designed for end-user efficiency?
8	On-Line update	How many ILF's are updated by On-Line transaction?
9	Complex processing	Does the application have extensive logical or mathematical processing?
10	Reusability	Was the application developed to meet one or many user's needs?
11	Installation ease	How difficult is conversion and installation?
12	Operational ease	How effective and/or automated are start-up, back up, and recovery procedures?
13	Multiple sites	Was the application specifically designed, developed, and supported to be installed at multiple sites for multiple organizations?
14	Facilitate change	Was the application specifically designed, developed, and supported to facilitate change?

*Figure 12 Fourteen general system characteristics*

Total Degree of Influence (TDI) is calculated by analyzing 14 general system characteristics.

#### Unadjusted function points (UFP)

Function points are a measure of functionality, that is, the usefulness of software tools from the user's point of view. Overall functionality is defined and measured by:

- analysis of logical groups of data that are used and maintained by the PS and characterize the functionality of the data;
- analysis of the information entered and output by the user, that is, the functionality of the transactions made.

Function point analysis considers two types of data groups:

Internal Logical File (ILF) - a logically related group of data, defined by the user and located within the boundaries of the software;

External Interface File (EIF) - a logically related group of data that provides software with information, but lies outside it and is supported by other software.



In the analysis based on the method of function points, three types of transactions are distinguished:

External Input (EI) - the process of data input coming to the input of PS, used to maintain the internal logical file. Typically, processes of the EI type are used to add, change, or delete information;

External Output (EO) - a process that generates data or control information that comes at the output of the software. Usually EO type process is the formation of various screens, reports, messages;

External Inquiry (EQ) - dialog input that results in an immediate response from the PS in the form of dialog output. In this case, the dialog input is not stored in the software itself, and the dialog output does not require calculations.

This article (Cost Estimation, 2022) says: “For each function identified above the function is further classified as *simple*, *average* or *complex* and a weight is given to each. The sum of the weights quantifies the size of information processing and is referred to as the Unadjusted Function points.”

Unadjusted Function points is calculated by multiplying the data and adding all the data ILF, EIF, EI, EO, EQ.

Formula for calculate FP:

$$FP = UFP * VAF$$

### **3.8 ISO/IEC 9126 software quality standards**

The standard defines six characteristics. These characteristics form the basis for further clarification and description of the quality of the software.

#### Functionality

A set of attributes related to the essence of a set of functions and their specific properties. Functions are those that fulfill stated or implied needs:

#### Notes

1 This set of attributes characterizes what the software does to satisfy needs, while the other sets mainly describe when and how it is done.

NOTE 2 In this characteristic, the quality note (see 3.6) is taken into account for the stated and implied needs.

#### 4.2 Reliability

A set of attributes related to the ability of software to maintain its level of performance under specified conditions over a specified period of time.

#### Notes

1 No deterioration or aging of software occurs. Reliability limitations arise from errors in requirements, design, and implementation. Failures due to these errors depend on the way you use the software and the previously selected software versions.

NOTE 2 In ISO 8402, reliability is the ability of an item to perform a required function. In this International Standard, functionality is only one of the characteristics of software quality. Therefore, the definition of reliability has been extended to “maintain its level of performance” instead of “performing the required function” (see also 3.4).

#### 4.3 Usability

A set of attributes related to the scope of work required for use and the individual assessment of such use by a specific or intended set of users.

##### Notes

1 "Users" can be interpreted as the majority of direct users of interactive software. The range of users may include operators, end-users and indirect users who are affected by the software or who depend on its use.

Practicality should be considered across the variety of user operating conditions that may affect the software, including preparation for use and evaluation of results.

NOTE 2 Practicality, defined in this standard as a specific set of attributes of a software product, differs from the ergonomic definition, which considers other characteristics such as efficiency and inefficiency as part of usability.

#### 4.4 Efficiencies

A set of attributes related to the relationship between the level of software performance and the amount of resources used under specified conditions.

NOTE Resources can include other software products, hardware, materials (eg printing paper, floppy disks), and the services of an operating, maintenance or service personnel.

4.5 Maintainability A set of attributes related to the scope of work required to implement specific changes (modifications).

NOTE A change may include fixes, enhancements, or software adaptations to changes in the environment, requirements and operating conditions.

#### 4.6 Portability

A set of attributes related to the ability of software to be ported from one environment to another.

NOTE The environment can include an organizational, technical or software environment.

### **3.8.1 Determination of integrated quality indicators**

#### **A.2.1 Functionality**

##### A.2.1.1 Suitability

A software attribute related to the presence and relevance of a feature set for a specific task.

NOTE Examples of correspondence are the composition of task-oriented functions, of its sub-functions and the volumes of tables.

#### A.2.1.2 Accuracy

Software attributes related to ensuring the correctness or consistency of results or effects.

NOTE For example, it includes the required degree of precision for the calculated values.

#### A.2.1.3 Interoperability

Attributes of software related to its ability to interact with specific systems.

#### A.2.1.4 Compliance

Software attributes that force a program to adhere to appropriate standards or conventions, or legal provisions, or similar guidelines.

#### A.2.1.5 Security

Attributes of software related to its ability to prevent unauthorized access, accidental or deliberate, to programs and data.

### A.2.2 **Reliability**

#### A.2.2.1 Maturity

Software attributes related to the failure rate of software errors.

#### A.2.2.2 Fault tolerance

Attributes of software related to its ability to maintain a certain level of performance in the event of software errors or disruption to a specific interface.

NOTE A defined performance level includes fail-safe capability.

#### A.2.2.3 Recoverability

Attributes of software related to its ability to recover a level of performance and recover data directly damaged in the event of a failure, and the time and effort required to do so.

### A.2.3 **Usability**

#### A.2.3.1 Understandability

Software attributes related to the user's effort to understand a general logical concept and its applicability.

#### A.2.3.2 Learnability

Attributes of the software related to the user's efforts to train in its application.

#### A.2.3.3 Operability

Software attributes related to user effort in operation and operational management.

### A.2.4 **Efficiency**

#### A.2.4.1 Time behavior

Software attributes related to response and processing times and speed of execution of its functions.

#### A.2.4.2 Resource utilization

Software attributes related to the amount of resources used and the duration of such use when performing a function.

### A.2.5 **Maintainability**

#### A.2.5.1 Analysability

Software attributes related to the effort required to diagnose deficiencies or failures, or identify parts for upgrades.

#### A.2.5.2 changeability

Software attributes related to the effort required to modify, correct a failure, or change operating conditions.

#### A.2.5.3 Stability

Software attributes related to the risk of unintended effects of the modification.

#### A.2.5.4 Testability

Software attributes related to the effort required to validate modified software.

### A.2.6 **Portability**

#### A.2.6.1 Adaptability

Attributes of the software related to the convenience of its adaptation to various specific operating conditions, without the use of other actions or methods, other than those intended for this in the software in question.

#### A.2.6.2 Installability

Software attributes related to the effort required to implement software in a specific environment.

#### A.2.6.3 Portability compliance

Software attributes that force a program to comply with standards or conventions related to portability.

#### A.2.6.4 Replaceability

Attributes of software related to the ease and complexity of using it instead of another specific software tool in the environment of that tool.

#### Notes

NOTE 1 Interchangeability is used instead of interoperability in order to avoid possible confusion with interoperability.

NOTE 2 Interchangeability with a particular software tool does not imply that the tool is interchangeable with the software in question.

NOTE 3 Interchangeability can include attributes of ease of implementation and adaptability.

## 4. Practical Part

### 4.1 Store on React

#### 4.1.1 Class diagram

The following picture shows an overview of the class diagram. The class diagram shows the general structure of the hierarchy of system classes, their cooperation, attributes, methods/operation-, interfaces, and the relationships among them.

The diagram shows such classes as Items, Favorites, Cart, Orders.

Each class has its own attributes, functions, and relationships between classes are established through structural association.

"Items" is considered as the main class. Composition is established between the classes "Items", "Favorites", "Cart". This is based on the fact that composition has a hard dependency on the lifetime of container class instances and contained class instances. If the container is destroyed, then all of its contents will also be destroyed.

In this case, classes cannot exist without class "Items". Destroying the "Items" class should result in the destruction of the "Favorites", "Cart", "Orders" classes.

"Favorites", "Cart", "Orders" have an association with each other. An association shows that objects of one entity (class) are related to objects of another entity in such a way that you can move from objects of one class to another.

After adding the product to favorites, the customer can add the product to the cart and after adding the product to the cart, he can checkout. These three classes are interconnected and move from objects of one class to another.

The store does not have authorization, registration and personal account, so the class "Person" is not created.

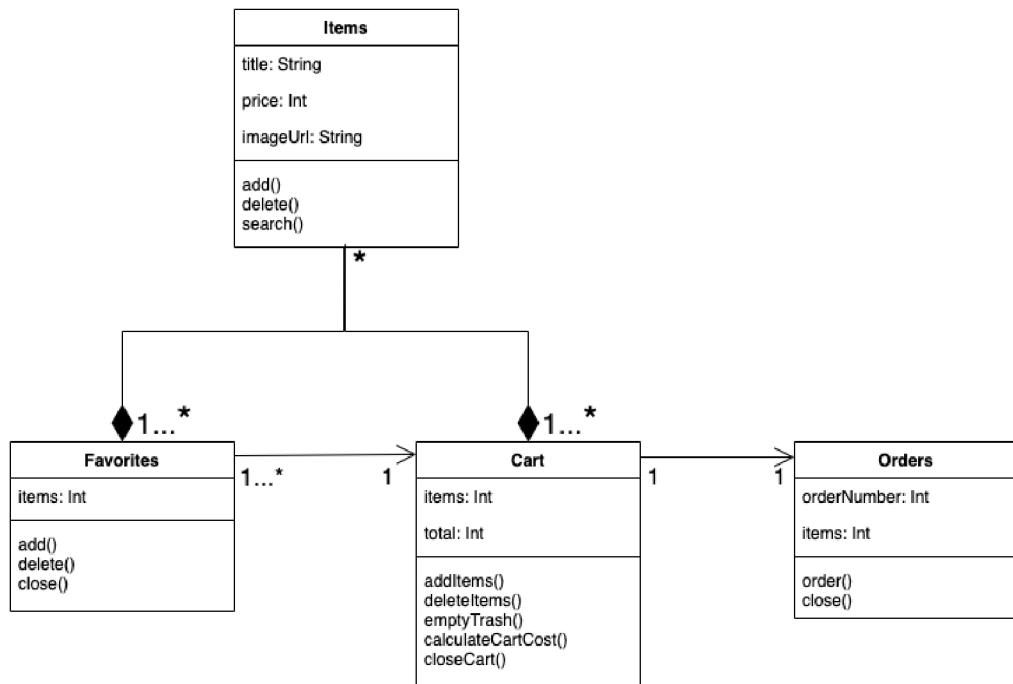


Figure 13 Class Diagram for store on react

#### 4.1.2 Use case diagram

Use case model diagram shows how the actors interact with the system. The diagram shows the user group "Customer", which is depicted as a person.

The first action "Customer" can take is to "Open the site", as shown in the ellipse. The use case is indicated on the diagram by an ellipse, inside of which there is a description indicating the execution of the action. There are also different types of relationships. On the diagram three types of relationships are shown. A solid line without an arrow indicates association relationships. Example "View product catalog" or "Add product to favorites".

The inclusion relation indicates that the specified behavior (Checkout) necessarily includes adding product to the cart. The extends relationship shows the possibility of expanding the base case, in this case it indicates the possible addition product from catalog to cart or the addition product from favorites to cart.

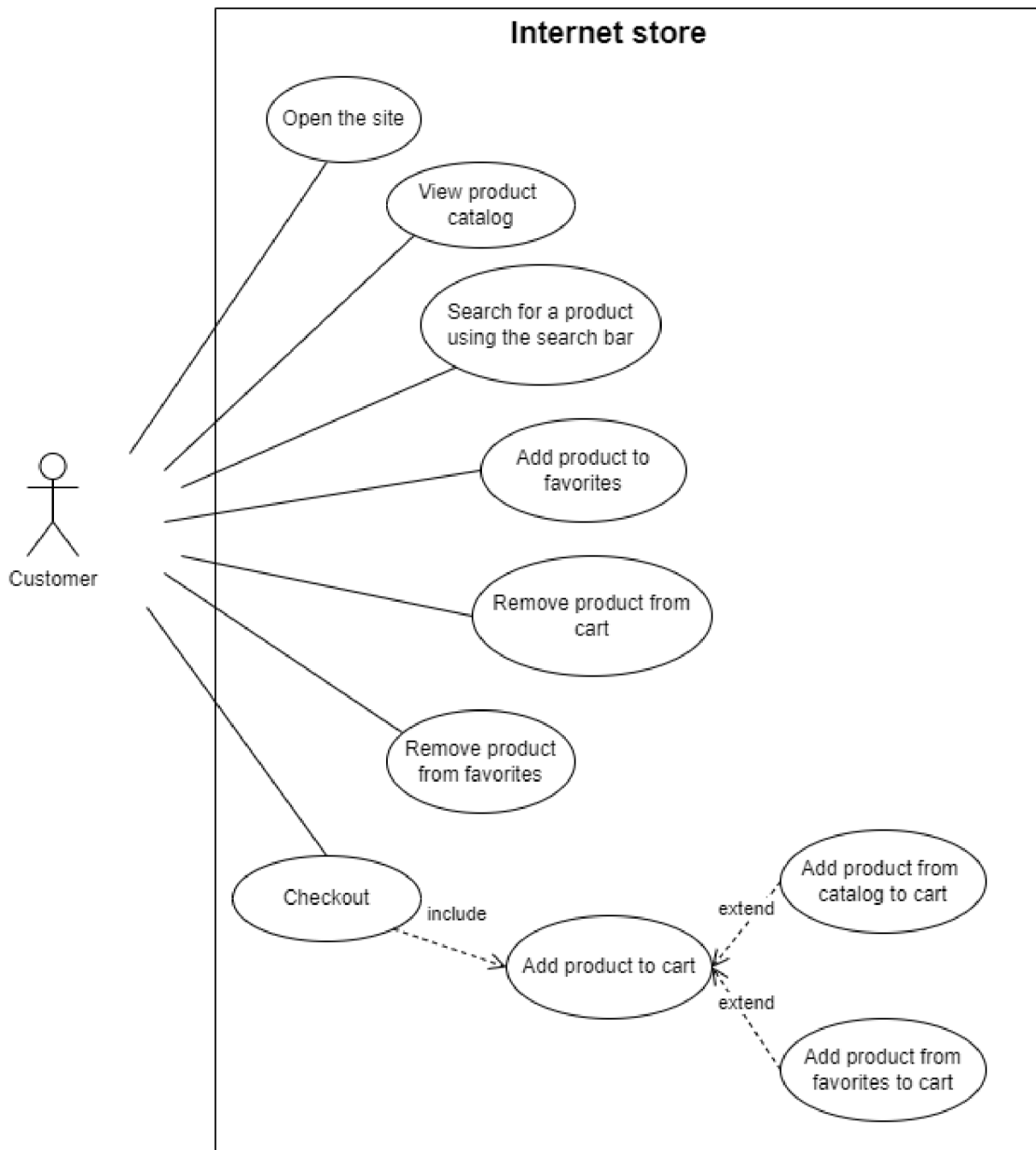


Figure 14 Use case for store on React

### 4.1.3 Sequence diagram

The sequence diagram (picture) reflects the flow of events occurring within the use case. It is used to refine precedent diagrams, a more detailed description of the logic of use cases. In this diagram, objects such as Customer, Store Interface , MockApi interact with each other, the messages they exchange, and the returned results associated with the messages are presented. The process begins when the “Customer” opens the site, at which point a request is sent to the MockApi to receive products. When “Customer” adds a product to the cart, the MockApi



also receives a request to save the data and create the cart. The diagram also shows other interactions. All interactions between objects occur sequentially.

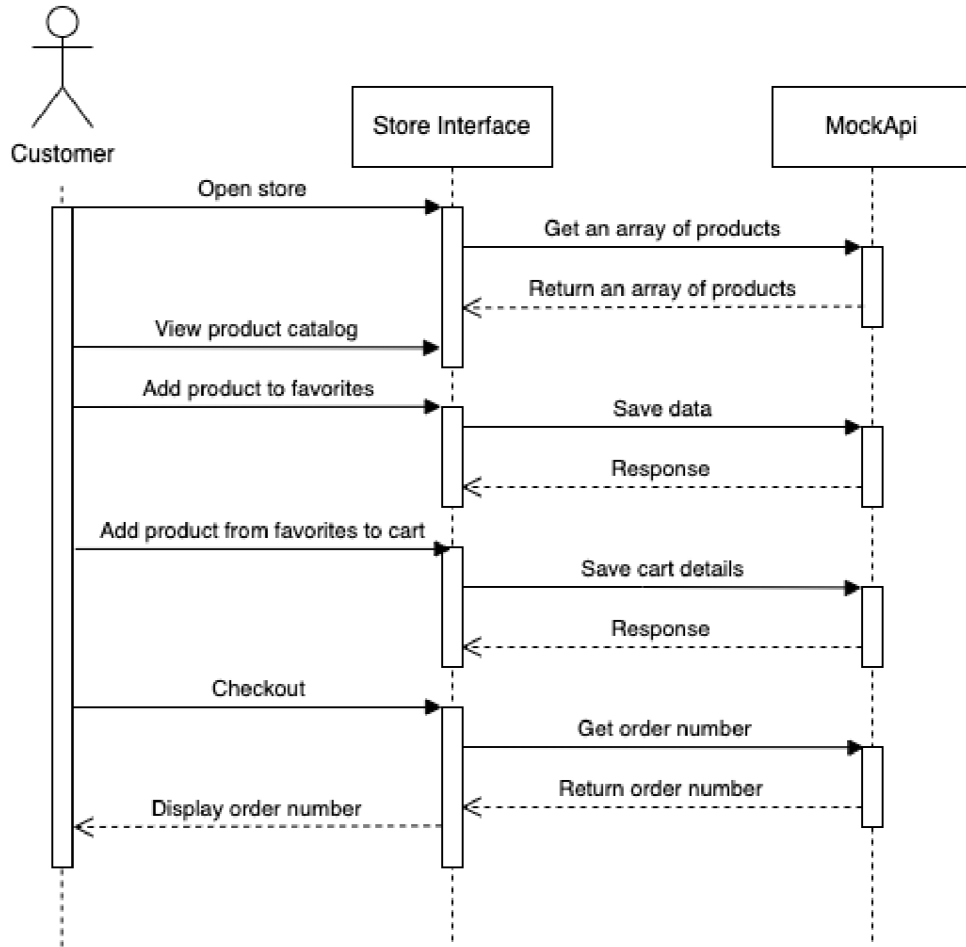


Figure 15 Sequence diagram for store on React

#### 4.1.4 Activity diagram

In the activity diagram (figure), we can see the entire flow of activity through the system during the interaction of the customer with the store and the final action checkout. This diagram includes two conditions and a branch, which is used to represent a stream that can branch into two parallel streams. The first condition is related to the choice of actions for favorites and the basket. The second condition is related to the choice of adding a product from favorites to the cart or performing an action.

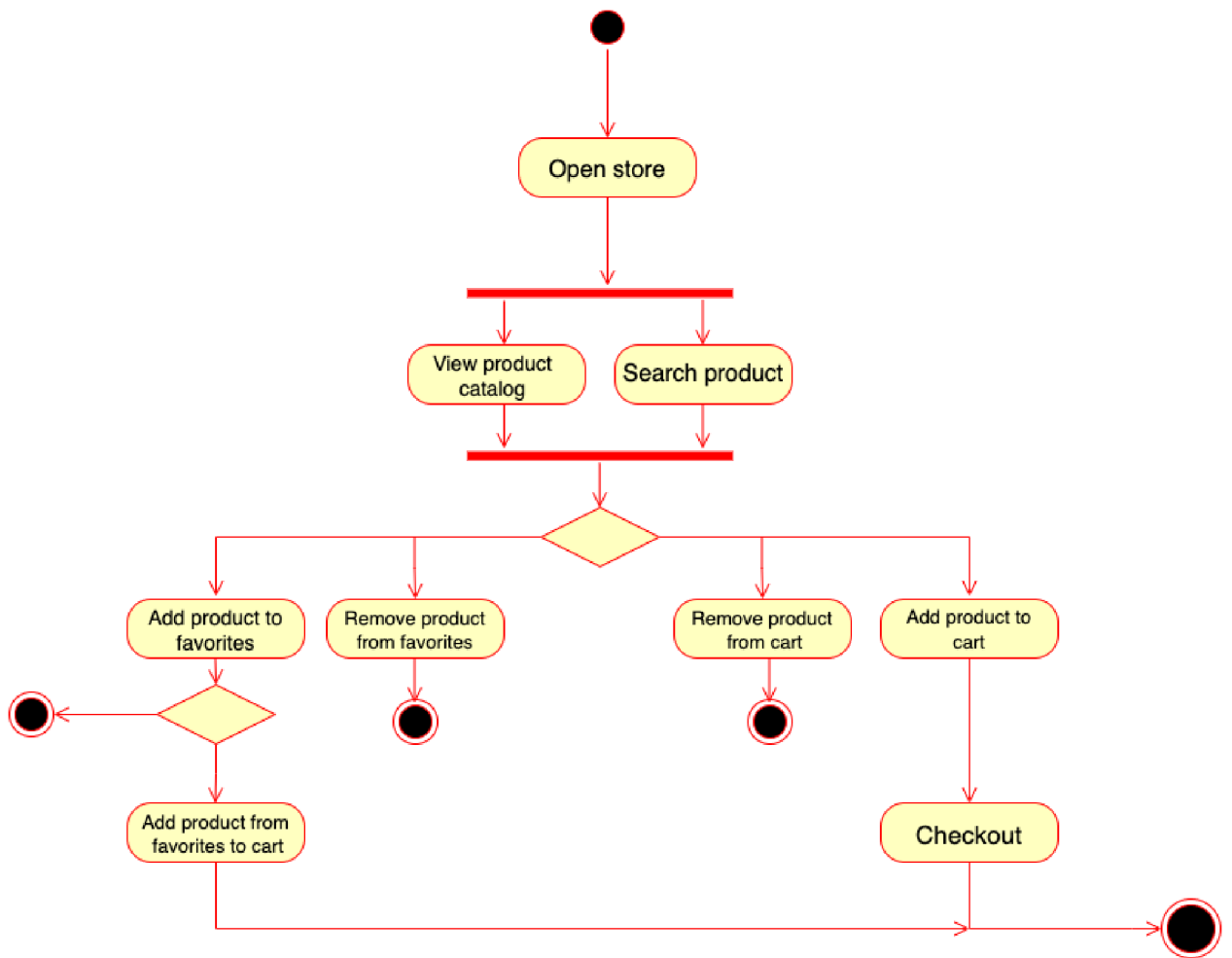


Figure 16 Activity diagram for store on React

## 4.2 Store on WordPress

### 4.2.1 Use case diagram

Use case model diagram (figure) for an online store shows a group of users "Client" and "Administrator" who are depicted as a person.

There are three types of relationships in the diagram. Association, inclusion and extension relations. An example of an inclusion relationship is "Register" and "Login", these actions are necessarily included in account management. Another example is "Leave feedback". When leaving a feedback, you must write a comment and set a rating about product.

An example of an extension relationship showing the possibility of extending the base option is the possible addition of a product to favorites or removal of a product from favorites when managing favorites.

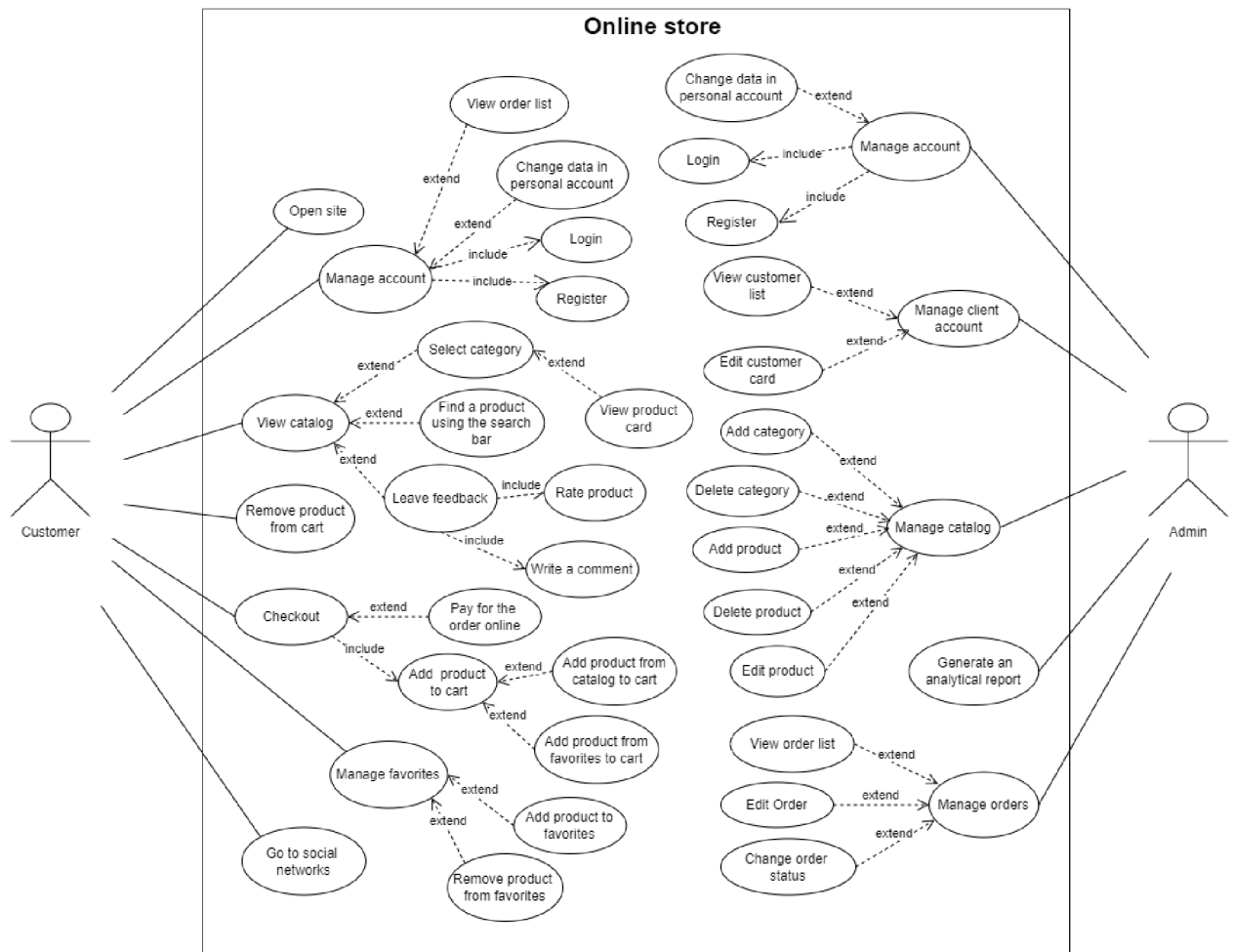


Figure 17 Use case diagram for store on WordPress

#### 4.2.2 Activity diagram

Store on wordpress has more features. The activity diagram implies that the person has already been registered and will log in.

This diagram includes five conditions and a branch that is used to represent the flow. The first condition is related to the opening of the product card. The customer can go to the product card, add the product to the cart or add to favorites or do it from the mini card in the catalog.

The second condition is related to the choice of actions for favorites and the basket. The third condition is related to the choice of adding a product from favorites to the cart or completing the action.

The fourth and fifth conditions are related to authorization. The customer can place an order without going through authorization or go through authorization with checking the validity of the data entered by the customer.

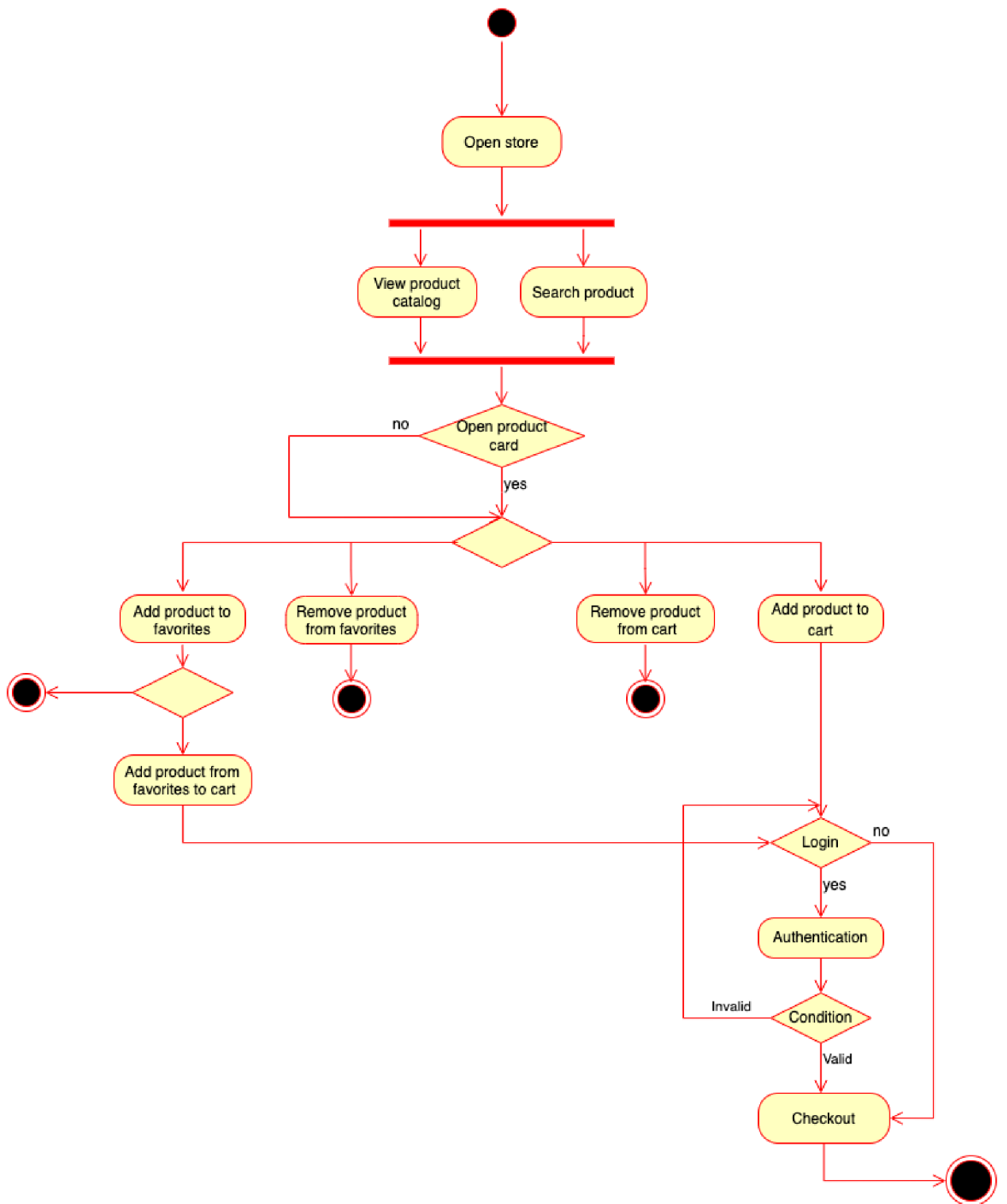


Figure 18 Activity diagram for store on WordPress

### 4.2.3 Sequence diagram

In this diagram, objects such as Customer, Store (WordPress), Admin, Admin panel, Database interact with each other, the messages they exchange, and the returned results associated with the messages are presented.

First, the interaction of the customer with the store is shown. The process begins when the "Customer" opens the store, goes to the catalog after which a request for products is sent to the Databases. When the "Customer" adds an product to the cart, the Database also receives a request to store the data and creating the cart.

Next is the moment of authorization. The sequence diagram assumes that the customer and admin have been previously registered and will log in.

The customer can log in at any time, it is not necessary to do this in the shopping cart. For authorization, the "Customer" enters data, the data is transferred to the database and checked, the result is returned. This process is placed in a loop. The loop is a fragment that can be executed several times, and the boundaries indicate the body of the iteration.

After authorization, the process of placing an order begins, all data is also transmitted and requested to the database.

After placing an order, a customer can cancel the order, ask the admin to edit the order.

The lower part shows the process of canceling an order by an admin in the admin panel. In order to enter the admin panel, the admin needs to log in. This process is similar to customer authorization and is also placed in a loop.

After authorization, the admin can change the status of the order to "Cancel" in the list of orders. All data is also transferred and queried to the database.

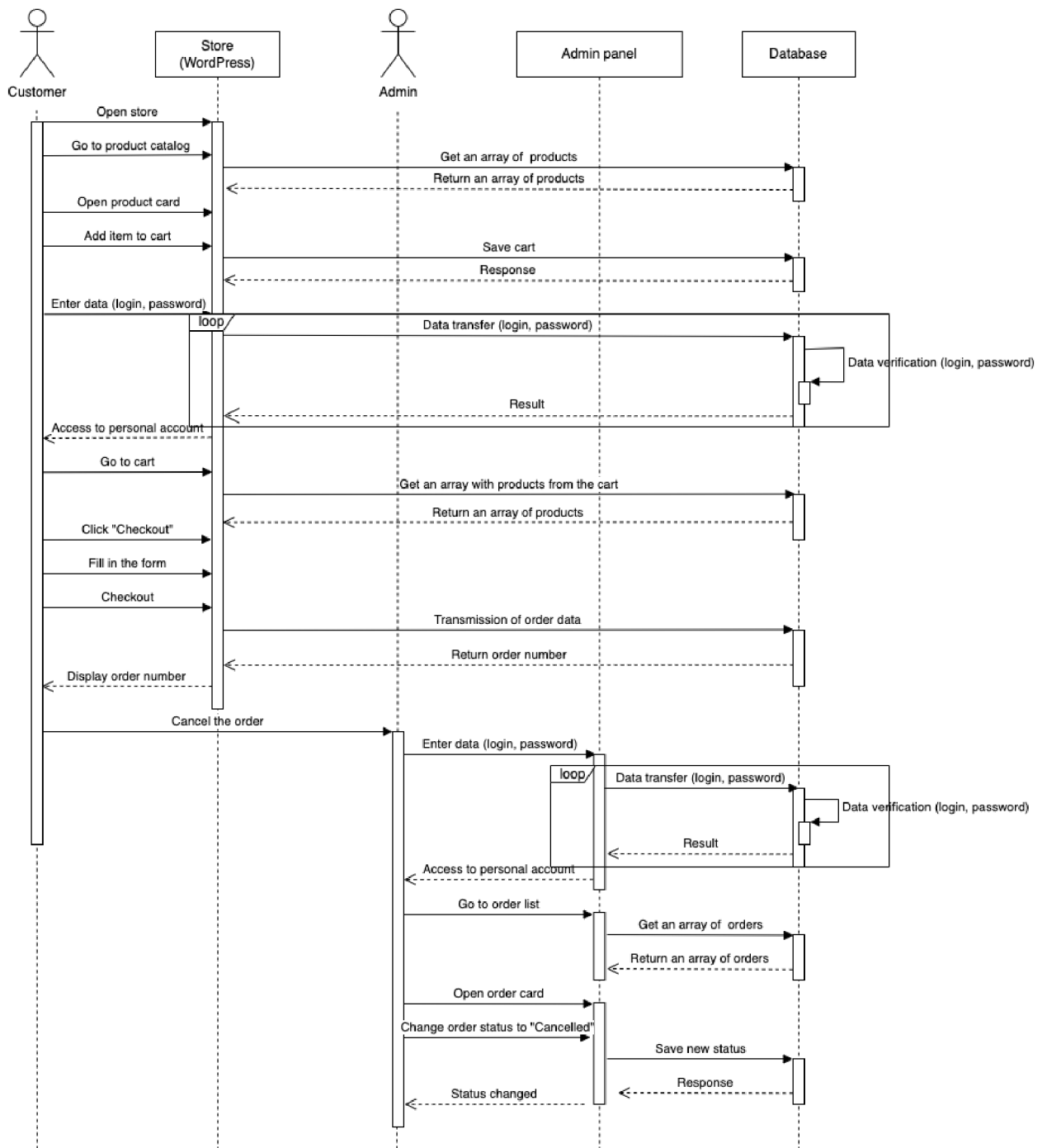


Figure 19 Sequence diagram for store on WordPress

#### 4.2.4 Database for store on WordPress

When WordPress renders any page, it connects to the database to display the content that the authors have added to the site.

The database for this store is named “wpfolder”. It consists of 58 tables.

The diagram below provides a visual overview of the WordPress database and table relationships created during a standard WordPress installation. (Codex, 2022)



3. wp\_links - The wp\_links table stores information related to links whose functionality was introduced as Links in WordPress.
4. wp\_options - Site Options and Settings (set in the admin panel on the settings page and in themes/plugins). This table is not related to other tables.
5. wp\_postmeta - Metadata for posts, pages, etc.
6. wp\_posts - Posts, pages, attachments, revisions, custom posts.
7. wp\_terms - The wp\_terms table stores categories for both posts and links, as well as tags for posts.
8. wp\_termmeta - Each term has information called metadata and it is stored in the wp\_termmeta table.
9. wp\_term\_relationships - Relationships between taxonomies and posts, pages
10. wp\_term\_taxonomy - Taxonomies (including categories and tags)
11. wp\_usermeta - Metadata for each user
12. wp\_users - List of users

The full list of tables in the database “wpfolder” is shown in the pictures below:

Table Name	Engine	Character Set	Collation	Row Count	Size
wp_actionscheduler_actions	InnoDB	utf8mb4_unicode_520_ci		58	175 K
wp_actionscheduler_claims	InnoDB	utf8mb4_unicode_520_ci		0	32 K
wp_actionscheduler_groups	InnoDB	utf8mb4_unicode_520_ci		5	32 K
wp_actionscheduler_logs	InnoDB	utf8mb4_unicode_520_ci		164	48 K
wp_cartflows_ca_cart_abandonment	InnoDB	utf8mb4_unicode_520_ci		2	32 K
wp_cartflows_ca_email_history	InnoDB	utf8mb4_unicode_520_ci		0	48 K
wp_cartflows_ca_email_templates	InnoDB	utf8mb4_unicode_520_ci		3	16 K
wp_cartflows_ca_email_templates_meta	InnoDB	utf8mb4_unicode_520_ci		15	32 K
wp_commentmeta	InnoDB	utf8mb4_unicode_520_ci		17	48 K
wp_comments	InnoDB	utf8mb4_unicode_520_ci		21	112 K
wp_lead_form	InnoDB	utf8mb4_unicode_520_ci		1	16 K
wp_lead_form_data	InnoDB	utf8mb4_unicode_520_ci		0	16 K
wp_lead_form_extension	InnoDB	utf8mb4_unicode_520_ci		0	16 K
wp_lead_form_options	InnoDB	utf8mb4_unicode_520_ci		0	16 K
wp_links	InnoDB	utf8mb4_unicode_520_ci		0	32 K
wp_options	InnoDB	utf8mb4_unicode_520_ci		574	7.1 M
wp_postmeta	InnoDB	utf8mb4_unicode_520_ci		1,867	496 K
wp_posts	InnoDB	utf8mb4_unicode_520_ci		175	192 K
wp_termmeta	InnoDB	utf8mb4_unicode_520_ci		79	48 K
wp_terms	InnoDB	utf8mb4_unicode_520_ci		55	48 K

Figure 21 Tables database “wpfolder”



Table Name	Engine	Character Set	Size (KB)
wp_term_relationships	InnoDB	utf8mb4_unicode_ci	32
wp_term_taxonomy	InnoDB	utf8mb4_unicode_ci	46
wp_th_popup	InnoDB	utf8mb4_unicode_ci	16
wp_usermeta	InnoDB	utf8mb4_unicode_ci	48
wp_users	InnoDB	utf8mb4_unicode_ci	64
wp_wc_admin_notes	InnoDB	utf8mb4_unicode_ci	64
wp_wc_admin_note_actions	InnoDB	utf8mb4_unicode_ci	32
wp_wc_category_lookup	InnoDB	utf8mb4_unicode_ci	16
wp_wc_customer_lookup	InnoDB	utf8mb4_unicode_ci	48
wp_wc_download_log	InnoDB	utf8mb4_unicode_ci	48
wp_wc_order_coupon_lookup	InnoDB	utf8mb4_unicode_ci	16
wp_wc_order_product_lookup	InnoDB	utf8mb4_unicode_ci	48
wp_wc_order_stats	InnoDB	utf8mb4_unicode_ci	64
wp_wc_order_tax_lookup	InnoDB	utf8mb4_unicode_ci	48
wp_wc_product_attribute_lookup	InnoDB	utf8mb4_unicode_ci	88
wp_wc_product_meta_lookup	InnoDB	utf8mb4_unicode_ci	112
wp_wc_rate_limits	InnoDB	utf8mb4_unicode_ci	32
wp_wc_reserved_stock	InnoDB	utf8mb4_unicode_ci	16
wp_wc_tax_rate_classes	InnoDB	utf8mb4_unicode_ci	32
wp_wc_webhooks	InnoDB	utf8mb4_unicode_ci	32

Figure 22 Tables database “wpfolder”

Table Name	Engine	Character Set	Size (KB)
wp_wc_reserved_stock	InnoDB	utf8mb4_unicode_ci	48
wp_wc_tax_rate_classes	InnoDB	utf8mb4_unicode_ci	32
wp_wc_webhooks	InnoDB	utf8mb4_unicode_ci	32
wp_woocommerce_api_keys	InnoDB	utf8mb4_unicode_ci	48
wp_woocommerce_attribute_taxonomies	InnoDB	utf8mb4_unicode_ci	2
wp_woocommerce_downloadable_product_permissions	InnoDB	utf8mb4_unicode_ci	88
wp_woocommerce_log	InnoDB	utf8mb4_unicode_ci	32
wp_woocommerce_order_itemmeta	InnoDB	utf8mb4_unicode_ci	48
wp_woocommerce_order_items	InnoDB	utf8mb4_unicode_ci	32
wp_woocommerce_payment_tokenmeta	InnoDB	utf8mb4_unicode_ci	48
wp_woocommerce_payment_tokens	InnoDB	utf8mb4_unicode_ci	32
wp_woocommerce_sessions	InnoDB	utf8mb4_unicode_ci	32
wp_woocommerce_shipping_zones	InnoDB	utf8mb4_unicode_ci	16
wp_woocommerce_shipping_zone_locations	InnoDB	utf8mb4_unicode_ci	48
wp_woocommerce_shipping_zone_methods	InnoDB	utf8mb4_unicode_ci	16
wp_woocommerce_tax_rates	InnoDB	utf8mb4_unicode_ci	88
wp_woocommerce_tax_rate_locations	InnoDB	utf8mb4_unicode_ci	48
wp_wpforms_tasks_meta	InnoDB	utf8mb4_unicode_ci	16
wp_wppb	InnoDB	utf8mb4_unicode_ci	16
wp_yith_wcwl	InnoDB	utf8_general_ci	32
yith_wcwl_lists	InnoDB	utf8_general_ci	48

Figure 23 Tables database “wpfolder”

List of tables and their values created during plugins installation.

### WooCommerce plugin tables:

1. actionscheduler\_actions - list of actions that the Action Scheduler will execute
2. actionscheduler\_claims - list of claims
3. actionscheduler\_groups - List of groups in which involved Action Scheduler
4. actionscheduler\_logs - actions performed by the action scheduler
5. woocommerce\_sessions - basically contains users carts
6. woocommerce\_api\_keys - API key store
7. woocommerce\_attribute\_taxonomies - categories, tags, etc.
8. woocommerce\_downloadable\_product\_permissions - downloads permissions
9. woocommerce\_order\_items - items related to orders

10. woocommerce\_order\_itemmeta - item metadata
11. woocommerce\_tax\_rates - list of tax rates set manually
12. woocommerce\_tax\_rate\_locations - location based tax rate data
13. woocommerce\_shipping\_zones - list of custom shipping zones
14. woocommerce\_shipping\_zone\_locations - location of shipping zones
15. woocommerce\_shipping\_zone\_methods - shipping methods for each zone
16. woocommerce\_payment\_tokens - payment tokens
17. woocommerce\_payment\_tokenmeta - payment token metadata
18. woocommerce\_log - event log
19. wc\_webhooks - webhooks
20. wc\_download\_log - download log
21. wc\_product\_meta\_lookup - lookup table that speeds up searching for products inside orders
22. wc\_tax\_rate\_classes - tax classes
23. wc\_reserved\_stock - stock that is reserved for availability at checkout
24. wp\_wc\_category\_lookup - category list
25. wp\_wc\_customer\_lookup - customer list
26. wp\_wc\_order\_product\_lookup - List of products from the order
27. wp\_wc\_order\_stats - Order status
28. wp\_wc\_order\_coupon\_lookup - list of coupons and conditions
29. wp\_wc\_order\_tax\_lookup - taxes on orders
30. wp\_wc\_product\_attributes\_lookup
31. wp\_wppb

#### **YITH WooCommerce Wishlist plugin tables:**

wp\_yith\_wcwl - Products added to favorites (wishlist)

wp\_yith\_wcwl\_lists - List wishlist

#### **WooCommerce Cart Abandonment Recovery plugin tables:**

wp\_cartflows\_ca\_cart\_abandonment - this table stores data on abandoned carts for which an order was not placed

wp\_cartflows\_ca\_email\_history - This table stores the email addresses of users on the checkout page if the user hasn't completed the purchase.

wp\_cartflows\_ca\_email\_templates - Templates for sending emails if they don't complete a purchase.

wp\_cartflows\_ca\_email\_templates\_meta - Metadata for users who did not complete the purchase.

#### **Wpforms plugin tables:**

wp\_lead\_form - List of contact forms

wp\_lead\_form\_data - data received from the form

wp\_lead\_form\_extension - List of connected extensions\

wp\_lead\_form\_options

wp\_wpforms\_tasks\_meta - - Metadata for form

#### **WP Popup Builder plugin tables:**

wp\_th\_popup - this table contains data on pop-up windows

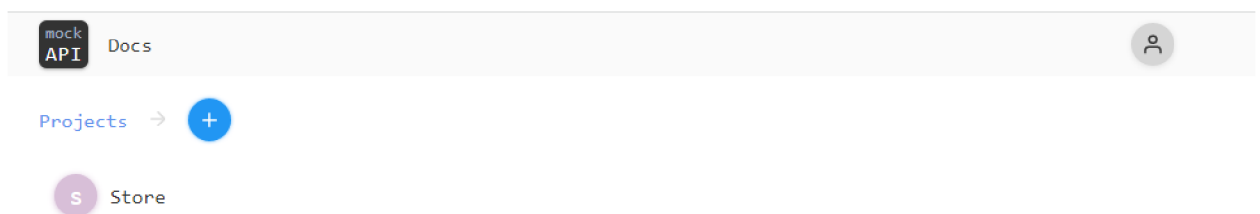
wp\_wc\_admin\_notes

wp\_wc\_admin\_note\_actions

### **4.3 MockAPI for store on react**

As previously written in the theoretical part, MockAPI allows "generate custom data, and preform operations on it using RESTful interface." ( MockAPI, 2022)

In MockAPI was created project "Store". In the project was created four resources.



*Figure 24 The figure shows the project "Store"*

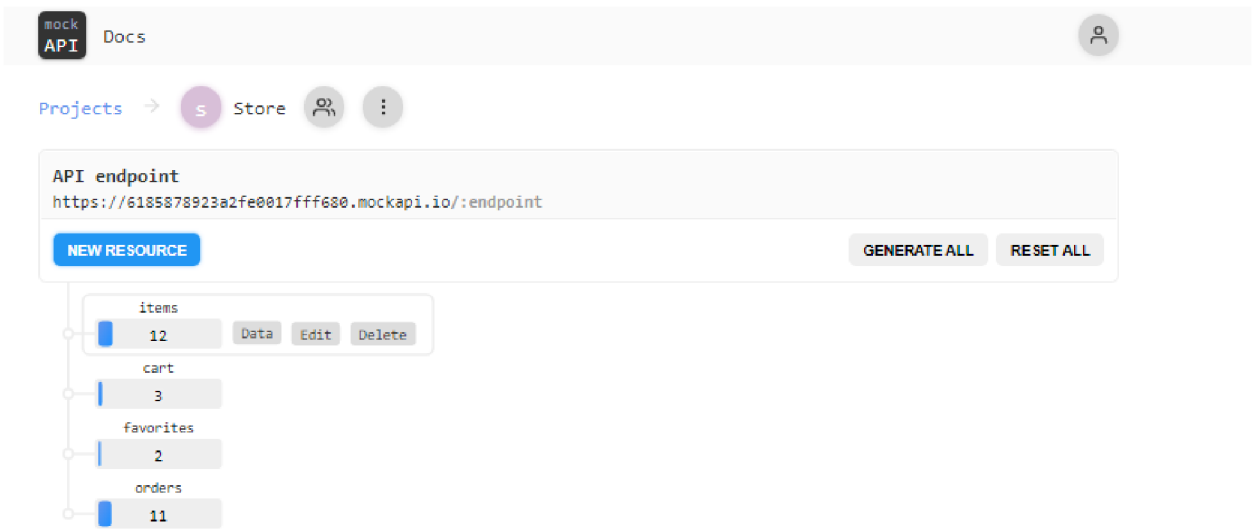


Figure 25 The figure shows the resources of the "Store" project

The first resource is "Items". All product data was added manually in JSON format. To display data on the site, the MockAPI leaves a request with method "GET" to receive an array of products. The pictures below show the entire list of products present in the store.

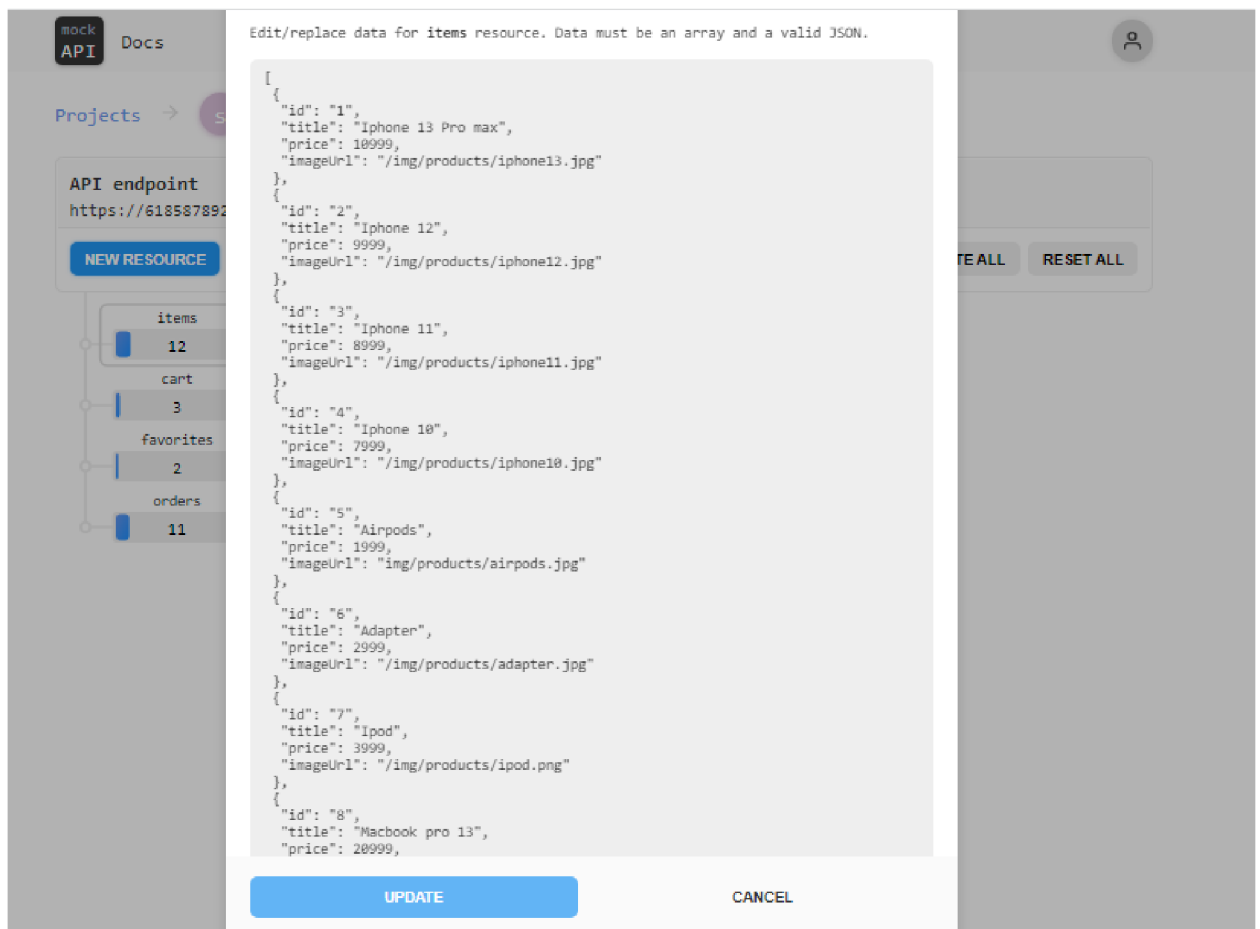
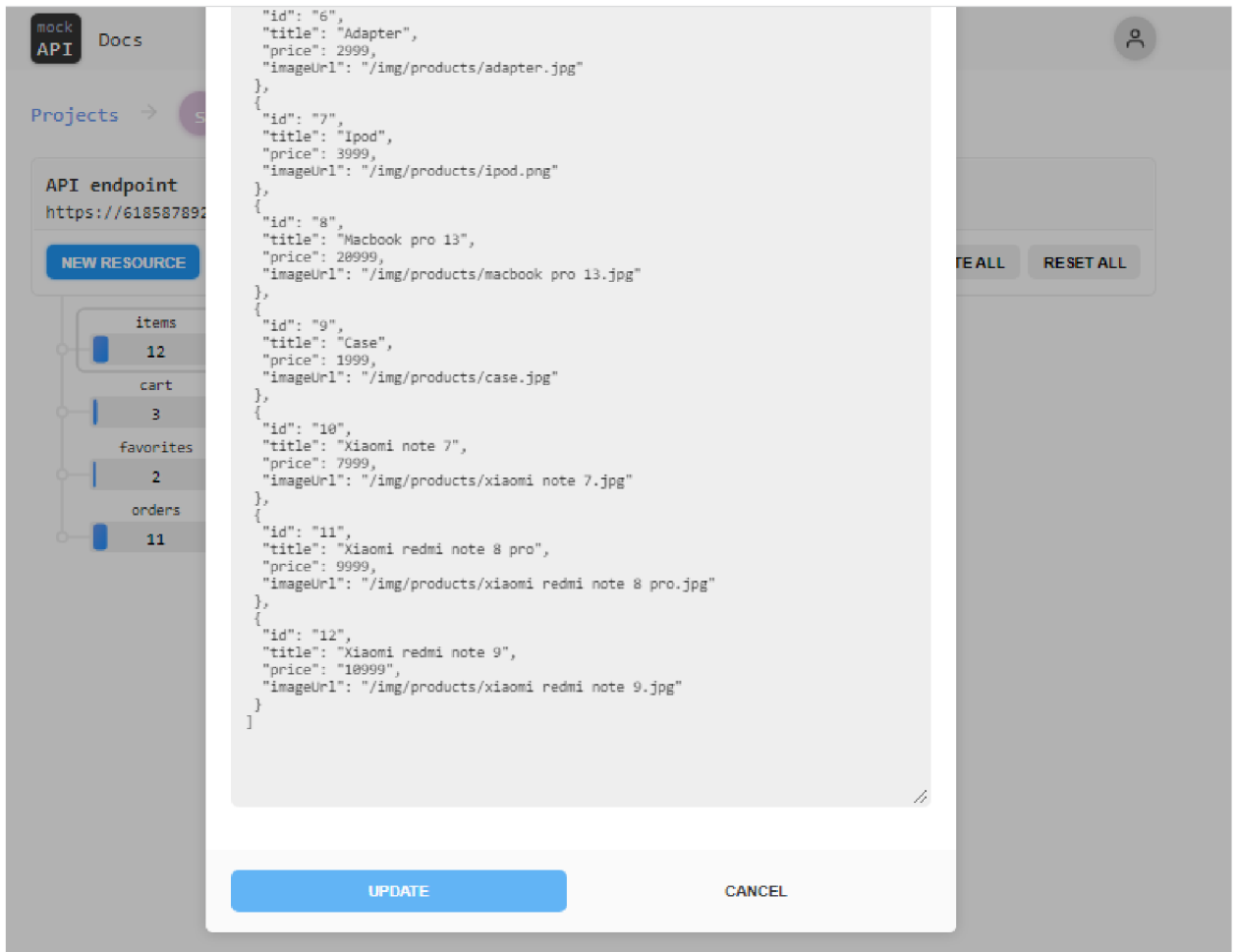


Figure 26 List of products



*Figure 27 List of products*

When adding a product to the cart, a request with the method “POST” is sent to MockAPI to the "Cart" resource to add products, where the added products are transmitted in the request body.

When a product is removed from the cart, a request with the method “DELETE” is sent to MockAPI to remove the product and the product is removed from the resource ”Cart”.

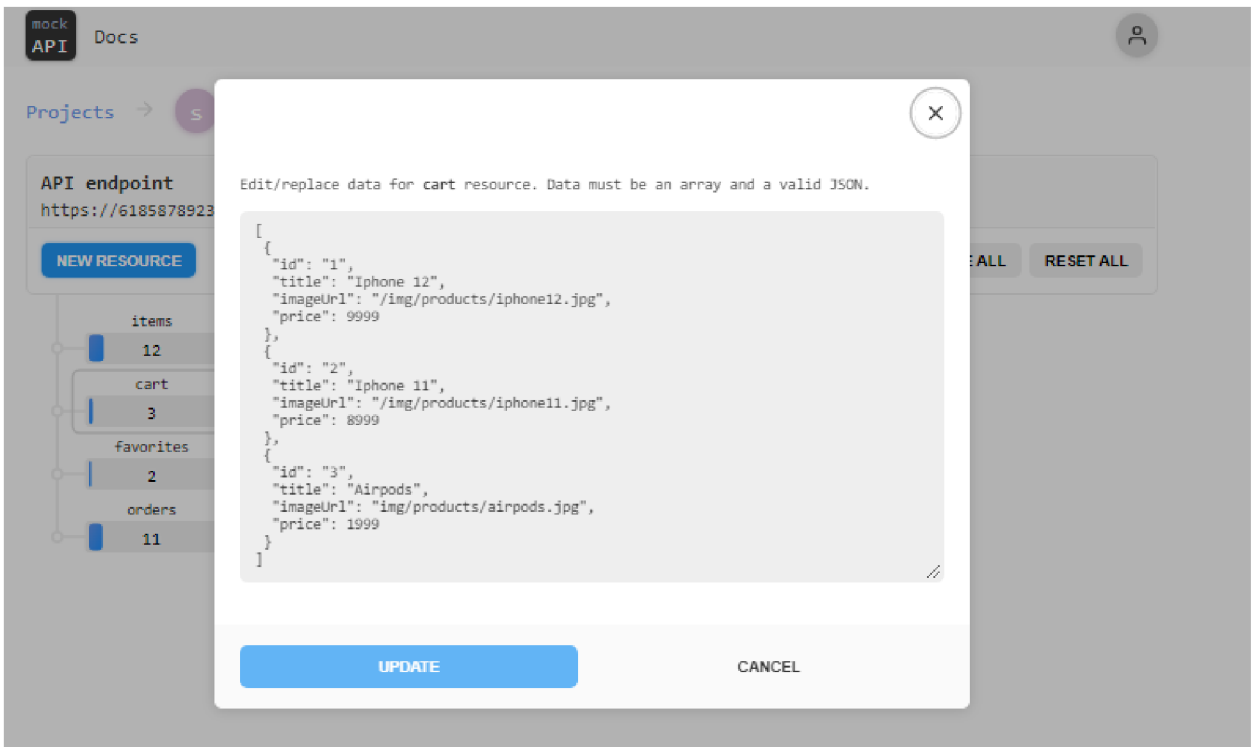


Figure 28 The picture shows the items added to the cart

When adding a product to the favorites, a request with the method “POST” is sent to MockAPI to the "favorites" resource to add products, where the added products are transmitted in the request body.

When a product is removed from the favorites, a request with the method “DELETE” is sent to MockAPI to remove the product and the product is removed from the resource ”favorites”.

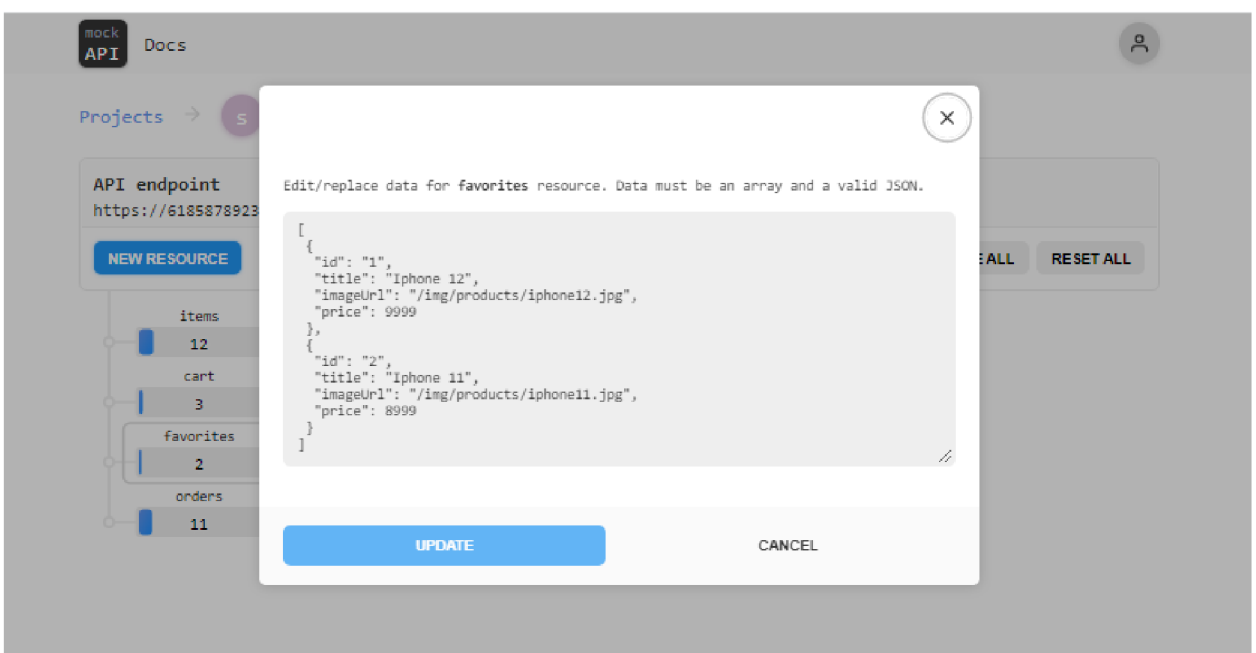


Figure 29 The picture shows products added to favorites

When placing an order in MockAPI, a request with method "POST" is sent to the "Orders" resource, where products from the order are transferred in the request body. In the response from MockAPI, the order number generated in MockAPI is returned.

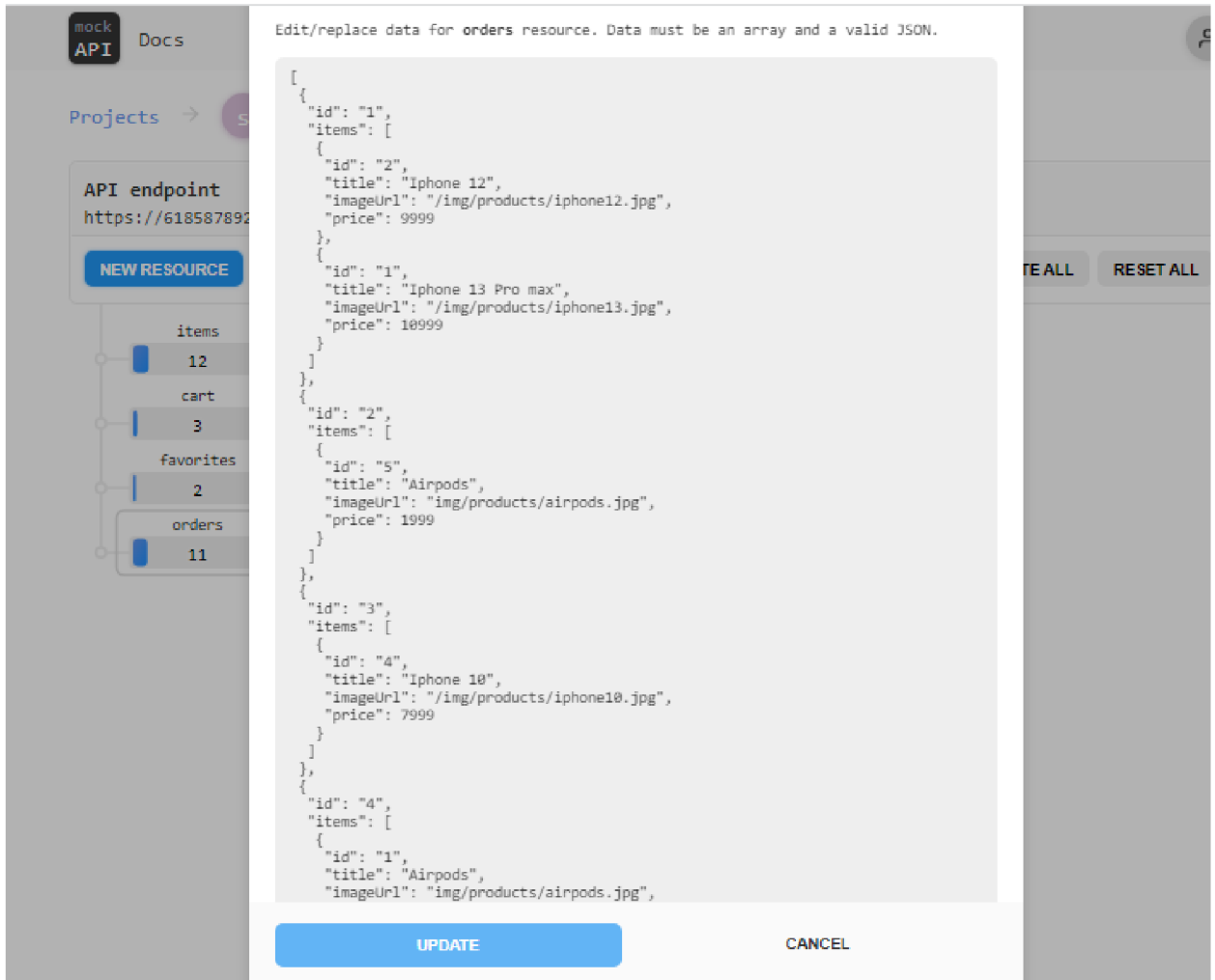


Figure 30 The figure shows the order numbers and order composition

#### 4.4 The process of creating an online store on React

To create a React store, you need to install the library react, node.js and a coding tool like Visual Studio.

After installing all the necessary tools and libraries, you can start creating store.

To create and run a react project, you need to use the commands:

```
npx create-react-app name-app
```

```
cd name-app
```

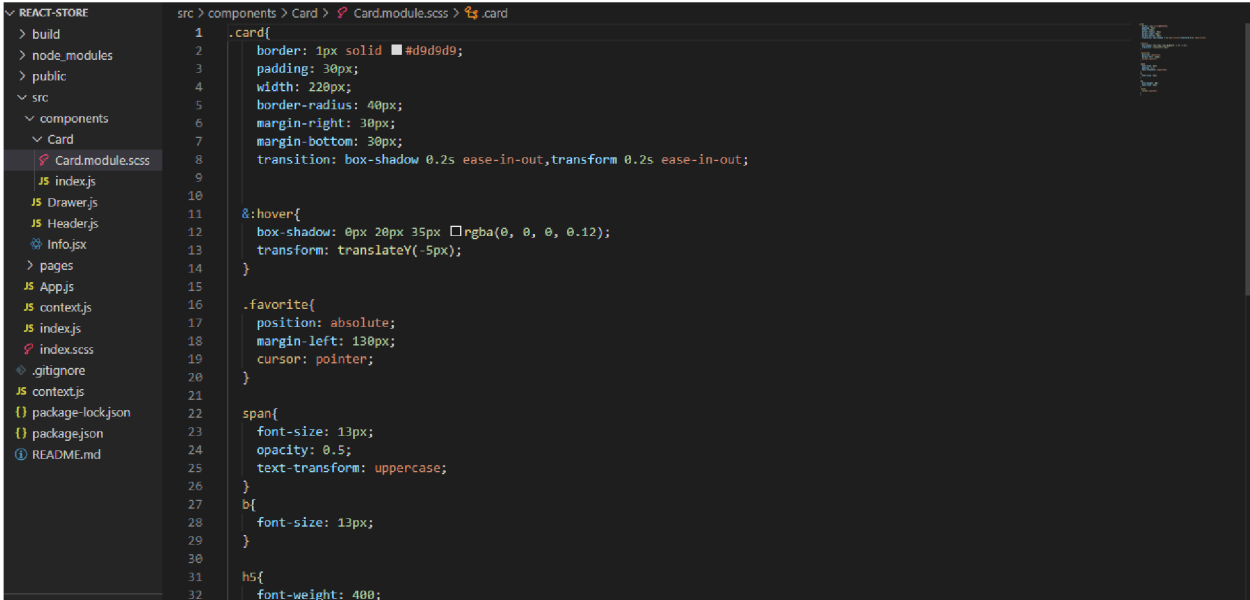
```
npx start
```

Steps to create a React store:

1. Create a visual part using SCSS for structuring. SCSS designed to simplify the creation of CSS code.
2. Create Home Pages
3. Create components in separate folders.
4. Create functionality to hide and open the basket using useState
5. We connect MockAPI and add all products by calling a request to the server. Using useEffect
6. We write adding to the cart (variable setCartItems)
7. Write the search (variable searchValue)
8. We write “Favorites” and install react router (we use it so that the transition to the bookmarks page is carried out immediately without reloading)
9. Render the products added to favorites on the favorites page
10. Make a button for placing an order, when clicked, a transition occurs to the checkout page and the basket is cleared.
11. Making the cart amount count
12. Add “try catch” to catch errors
13. Checkout and returning the order number

### Store interface description and structure

This file describes all store styles on SCSS.



```
1 .card{
2   border: 1px solid #d9d9d9;
3   padding: 30px;
4   width: 220px;
5   border-radius: 40px;
6   margin-right: 30px;
7   margin-bottom: 30px;
8   transition: box-shadow 0.2s ease-in-out,transform 0.2s ease-in-out;
9
10
11  &:hover{
12    box-shadow: 0px 20px 35px rgba(0, 0, 0, 0.12);
13    transform: translateY(-5px);
14  }
15
16  .favorite{
17    position: absolute;
18    margin-left: 130px;
19    cursor: pointer;
20  }
21
22  span{
23    font-size: 13px;
24    opacity: 0.5;
25    text-transform: uppercase;
26  }
27  b{
28    font-size: 13px;
29  }
30
31  h5{
32    font-weight: 400;
```

Figure 31 Scss

Drawer and Header files are components. Components are reusable code. They are repeating blocks. These components use a hook called useState and useContext. useState changes the state.



Example work hook "useState":

State "isLoading" is in the function "onClickOrder".

If "setIsLoading(true)" then an order request is sent to the MockAPI where in body of request contains "items" from the order.

If "setIsLoading=false" then an error is displayed 'Failed to create order' using "try catch"

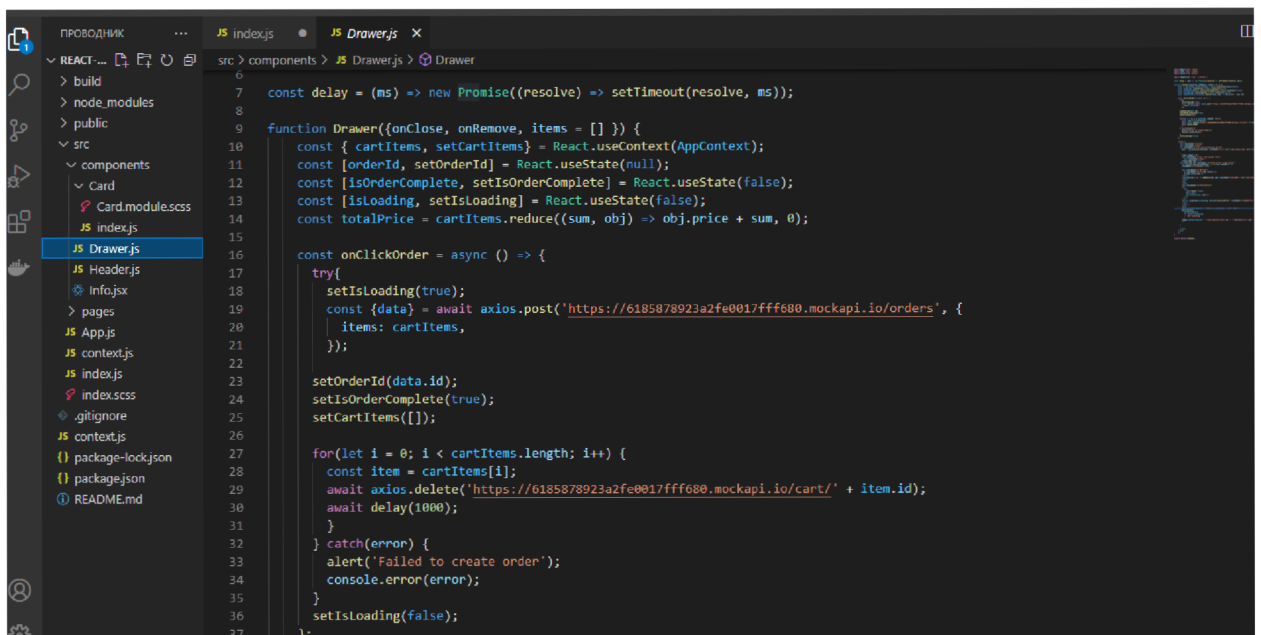
If state isOrderComplete = true (setIsOrderComplete(true);) then

in title shows "Order is processed" and description `Order № {orderId}`

where "orderId" is the order number

If state isOrderComplete = false, then

in title shows "Cart empty" and description 'Add something'



```
6
7 const delay = (ms) => new Promise((resolve) => setTimeout(resolve, ms));
8
9 function Drawer({onClose, onRemove, items = []}) {
10   const { cartItems, setCartItems } = React.useContext(AppContext);
11   const [orderId, setOrderId] = React.useState(null);
12   const [isOrderComplete, setIsOrderComplete] = React.useState(false);
13   const [isLoading, setIsLoading] = React.useState(false);
14   const totalPrice = cartItems.reduce((sum, obj) => obj.price + sum, 0);
15
16   const onClickOrder = async () => {
17     try {
18       setIsLoading(true);
19       const {data} = await axios.post('https://6185878923a2fe0017fff680.mockapi.io/orders', {
20         items: cartItems,
21       });
22
23       setOrderId(data.id);
24       setIsOrderComplete(true);
25       setCartItems([]);
26
27       for(let i = 0; i < cartItems.length; i++) {
28         const item = cartItems[i];
29         await axios.delete('https://6185878923a2fe0017fff680.mockapi.io/cart/' + item.id);
30         await delay(1000);
31       }
32     } catch(error) {
33       alert('Failed to create order');
34       console.error(error);
35     }
36     setIsLoading(false);
37   };

```

Figure 32 Drawer

```

src > components > JS Drawerjs > Drawer > onClickOrder
51 <div style={{backgroundImage: `url(${obj.imageUrl})`}}
52 className="cartItemImg"></div>
53
54 <div className="mr-20 flex">
55 <p className="mb-5">{obj.title}</p>
56 <b>{obj.price}</b>
57 </div>
58 <img onClick = {( () => onRemove(obj.id)) } className="removeBtn" src="/img/remove.svg" alt="remove" />
59 </div>
60 </div>
61 </div>
62 <div className="cartTotalBlock">
63 <ul>
64 <li>
65 <span>Total:</span>
66 <div></div>
67 <b>{ totalPrice } czk</b>
68 </li>
69 </ul>
70 <button disabled={isLoading} onClick={onClickOrder} className="orangeButton">Checkout</button>
71 </div>
72 :
73 <Info
74 title={isOrderComplete ? "Order is processed" : "Cart empty"}
75 description={
76 isOrderComplete
77 ? `Order# ${orderId}`
78 : 'Add something'
79 }
80 image={isOrderComplete ? "/img/complete-order.jpg" : "/img/empty-cart.jpg"}
81 />
82

```

Figure 33 Drawer

```

src > components > JS Headerjs > Header
1 import React from 'react';
2 import { Link } from 'react-router-dom';
3 import AppContext from '../context';
4
5 function Header(props) {
6   const { cartItems } = React.useContext(AppContext);
7   const totalPrice = cartItems.reduce((sum, obj) => obj.price + sum, 0);
8
9   return(
10 <header className="d-flex justify-between align-center p-40">
11 <Link to="/" />
12 <div className="d-flex align-center">
13 
14 <div>
15 <h3 className="text-uppercase">Digital Shop</h3>
16 <p className="opacity-5">Best digital store</p>
17 </div>
18 </div>
19 <Link to="/favorites" />
20 <ul className="d-flex">
21 <li onClick={props.onClickCart} className="mr-30 cu-p">
22 
23 <span>{ totalPrice } czk</span>
24 </li>
25 <li className="mr-20 cu-p">
26 <Link to="/favorites" />
27 
28 </li>
29 <li>
30 
31 </li>
32 </ul>

```

Figure 34 Header

The APP.js file contains logic for displaying data when opening a store, adding products to the cart, deleting a product, adding to favorites. This functionality is implemented using a hook: useEffect, useState and useContext. Requests are sent using the Axios library.

Example work useEffect:

By using hook `useEffect`, you are telling React that your component should do something after rendering.

That is, when only the first time the `APP.js` component is displayed, only then the code is called (three requests are sent and the received data is displayed) located inside `useEffect`.

This action is only performed on the first rendering.

This code:

```
19 React.useEffect(() => {
20   async function fetchData() {
21     try {
22       const cartResponse = await axios.get('https://6185878923a2fe0017fff680.mockapi.io/cart');
23       const favoritesResponse = await axios.get('https://6185878923a2fe0017fff680.mockapi.io/favorites');
24       const itemsResponse = await axios.get('https://6185878923a2fe0017fff680.mockapi.io/items');
25
26       setIsLoading(false);
27
28       setCartItems(cartResponse.data);
29       setFavorites(favoritesResponse.data);
30       setItems(itemsResponse.data);
31     } catch (error) { |
32       | alert('Error while requesting data');
33     }
34   }
35
36   fetchData();
37 }, []);
38
```

*Figure 35 useEffect*

Example work useContext:

Context provides a way to pass data through the component tree without having to pass props down manually at every level. (Context, 2022)

In other words:

We create one global object. In this object we will store all the necessary data and all the components we need will depend on this data that we have in the object.

This logic will notify every component that depends on it that the object has changed.

How it looks in code:

Create an empty object in `context.js`. Import it in `App.js`

```
const AppContext = React.createContext({});
```

Here we say that there is a `"AppContext.Provider"`, it has data and they will be available to the entire application. That is, the properties `{ items, cartItems, favorites, isItemAdded, onAddToFavorite, setCartOpened, setCartItems }` will be available in `Header, Home, Favorites`.

That is, it shortens the code.

This code:

```

1 import React from 'react';
2 import { Routes, Route } from 'react-router-dom';
3 import Home from './pages/Home';
4 import Drawer from './components/Drawer';
5 import Header from './components/Header';
6 import axios from 'axios';
7 import Favorites from './pages/Favorites';
8 import AppContext from './context';
9
10
11 function App() {
12   const [items, setItems] = React.useState([]);
13   const [cartItems, setCartItems] = React.useState([]);
14   const [favorites, setFavorites] = React.useState([]);
15   const [searchValue, setSearchValue] = React.useState('');
16   const [cartOpened, setCartOpened] = React.useState(false);
17   const [isLoading, setIsLoading] = React.useState(true);
18
19   React.useEffect(() => {
20     async function fetchData() {
21       try {
22         const cartResponse = await axios.get('https://6185878923a2fe0017fff680.mockapi.io/cart');
23         const favoritesResponse = await axios.get('https://6185878923a2fe0017fff680.mockapi.io/favorites');
24         const itemsResponse = await axios.get('https://6185878923a2fe0017fff680.mockapi.io/items');
25
26         setIsLoading(false);
27
28         setCartItems(cartResponse.data);
29         setFavorites(favoritesResponse.data);
30         setItems(itemsResponse.data);
31       } catch (error) {

```

Figure 36 App

```

1 import React from "react";
2
3
4 const AppContext = React.createContext({});
5
6 export default AppContext;

```

Figure 37 Context

```

90 return (
91   <AppContext.Provider value={{ items, cartItems, favorites, isItemAdded, onAddToFavorite, setCartOpened, setCartItems }}>
92     <div className="wrapper clear">
93       {cartOpened && <Drawer items={cartItems} onClose={() => setCartOpened(false)} onRemove={onRemoveItem}/>}
94       <Header onClickCart={() => setCartOpened(true)} />
95
96       <Routes>
97         <Route path="/" element={<Home items={items}
98           cartItems={cartItems}
99           searchValue={searchValue}
100           setSearchValue={setSearchValue}
101           onChangeSearchInput={onChangeSearchInput}
102           onAddToFavorite={onAddToFavorite}
103           onAddToCart={onAddToCart}
104           isLoading={isLoading}
105         />} />
106         <Route path="/favorites" element={<Favorites />} />
107       </Routes>
108     </div>
109   </AppContext.Provider>
110 );

```

Figure 38 Context

#### 4.4.1 Store description

When user open the site, the first time the site is opened (not reloaded), three requests are sent.

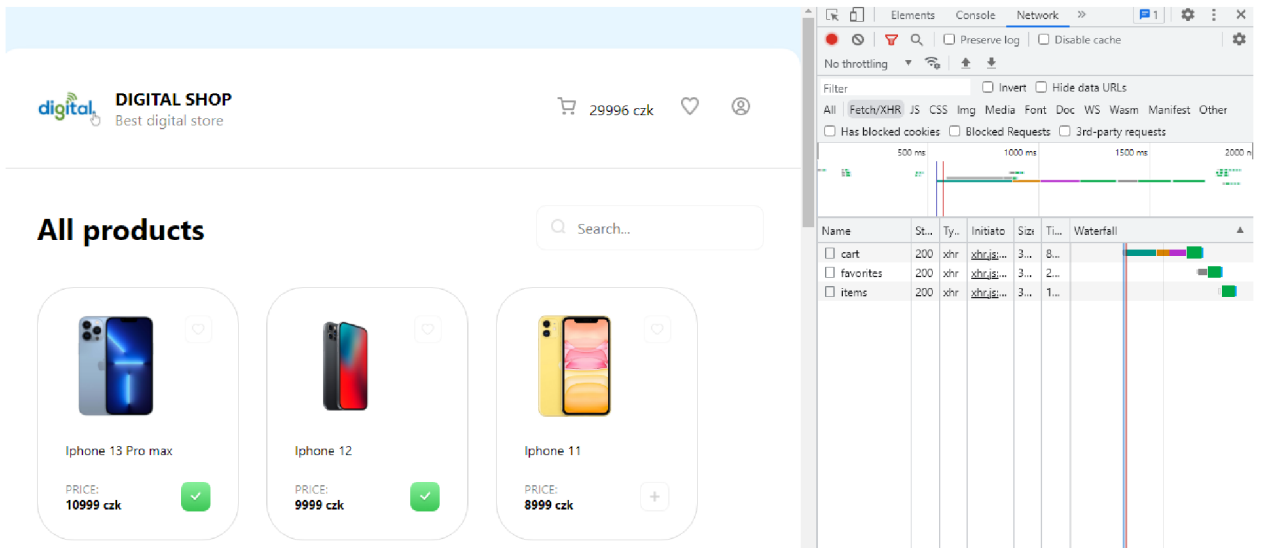


Figure 39 Open site

The user can also add an item to the shopping cart. When adding an item to the cart, a request is sent to add items to the MockAPI.

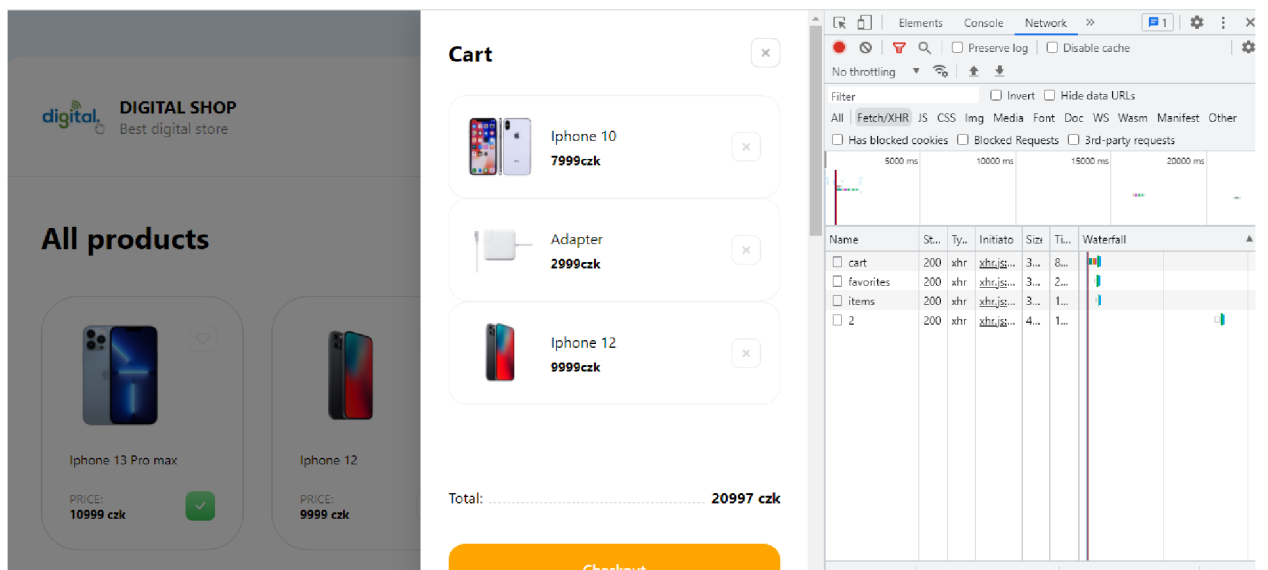


Figure 40 Card

The user can also add a product to favorites and place an order.

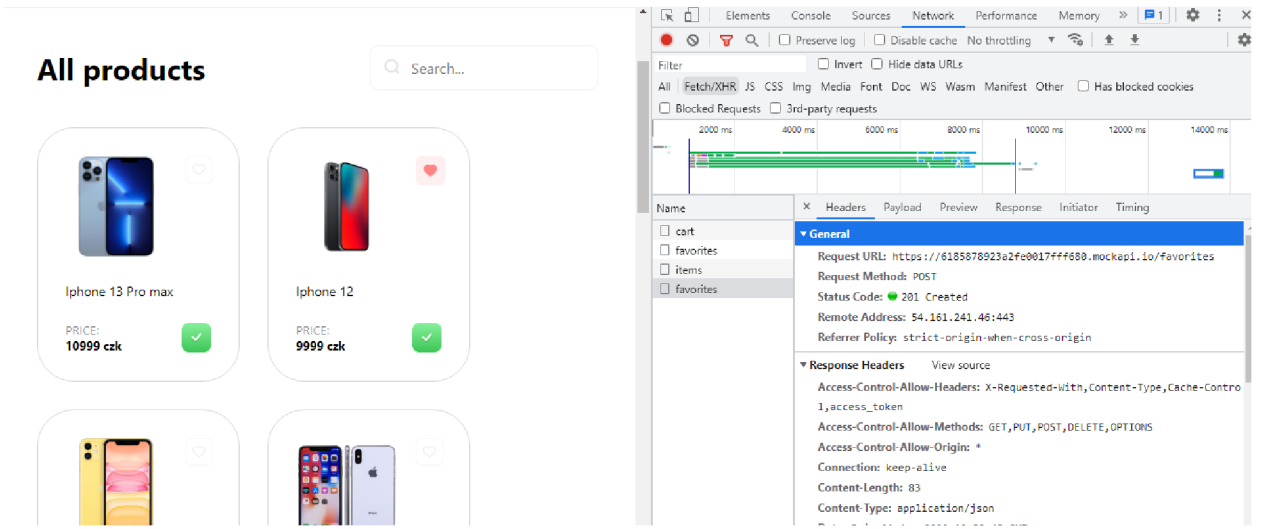


Figure 41 Favorites

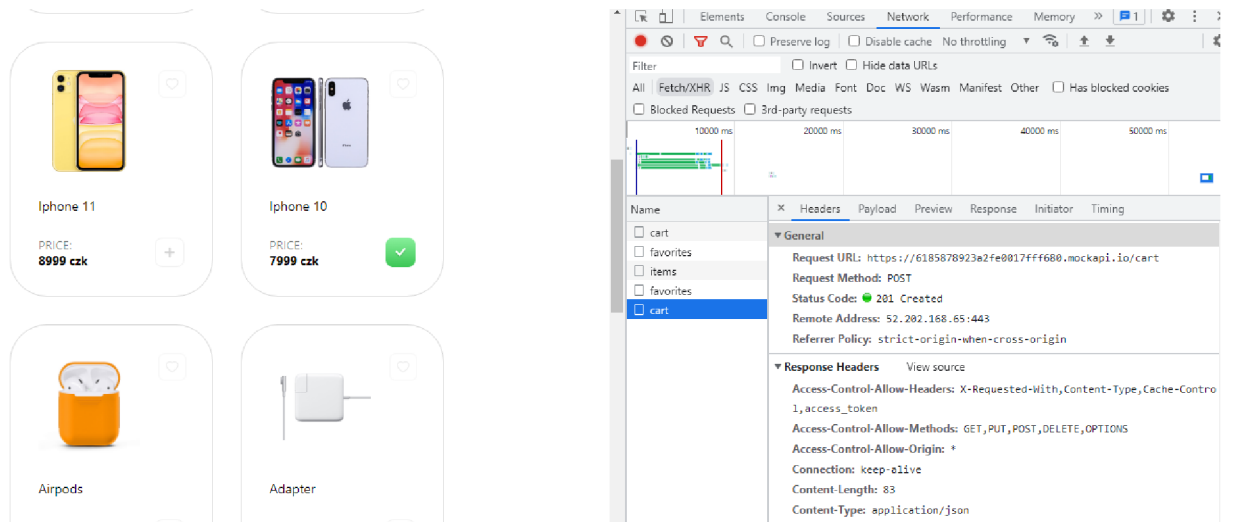
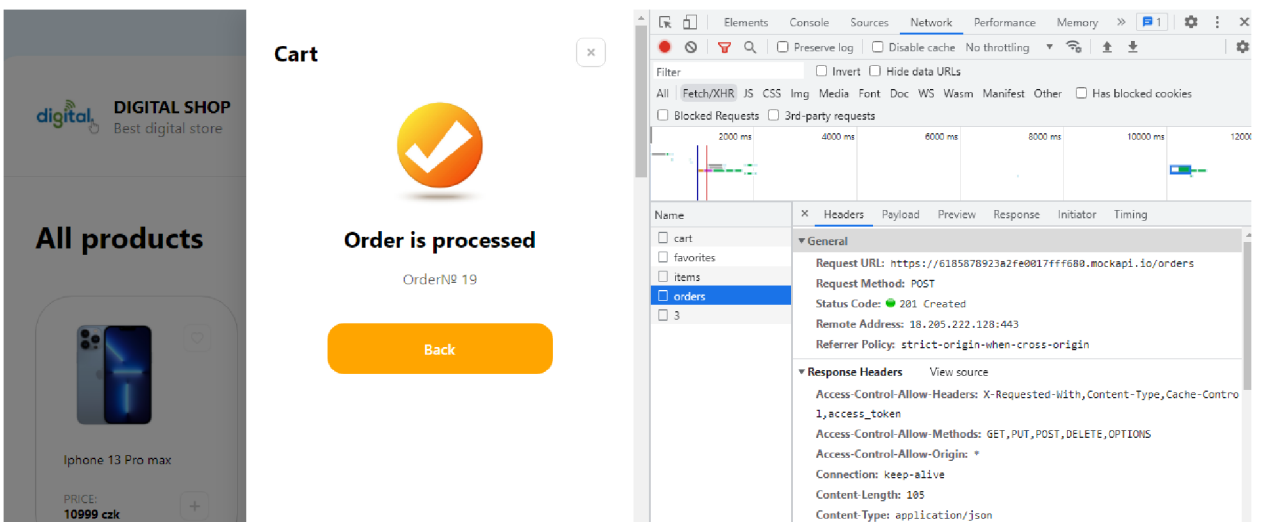


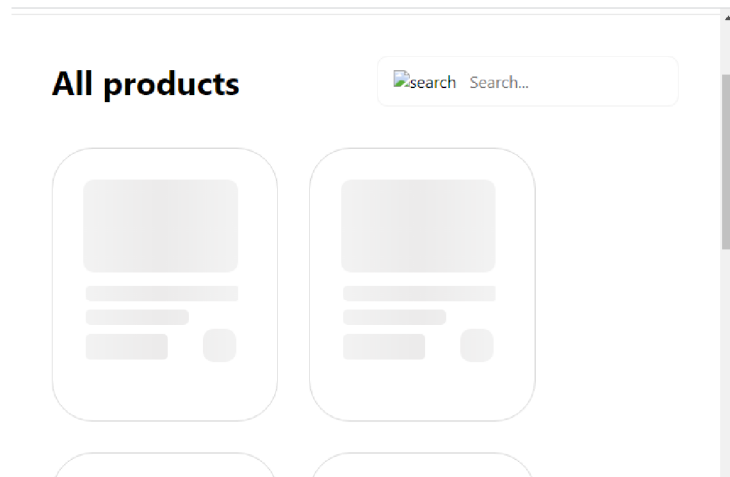
Figure 42 Cart

When placing an order, a request with order data is sent to the MockAPI and the order number is returned from the MockAPI.



*Figure 43 Checkout*

When loading the site, product cards are displayed like this:



*Figure 44 Products card*

## **4.5 The process of creating an online store on WordPress**

### Installation

To create a WordPress site, the first thing to do is to set up a local server. In my case it was OpenServer. After installing and configuring it, you need to install WordPress itself on your computer. After installation in the local server folder, you need to create a folder for the project. All WordPress files and folders must be moved to this folder.

### Store creation

After installation, you need to start the server and phpMyAdmin. In OpenServer, in the tabs "My sites", you need to run the project. In the WordPress interface, you need to create a database and fill in the information to create a user and site name.

The next step is to login to the admin panel.

To create a store, you need to install a special WooCommerce plugin and select a theme.

WordPress has various plugins. Additionally, I installed plugins for the full operation of the store:

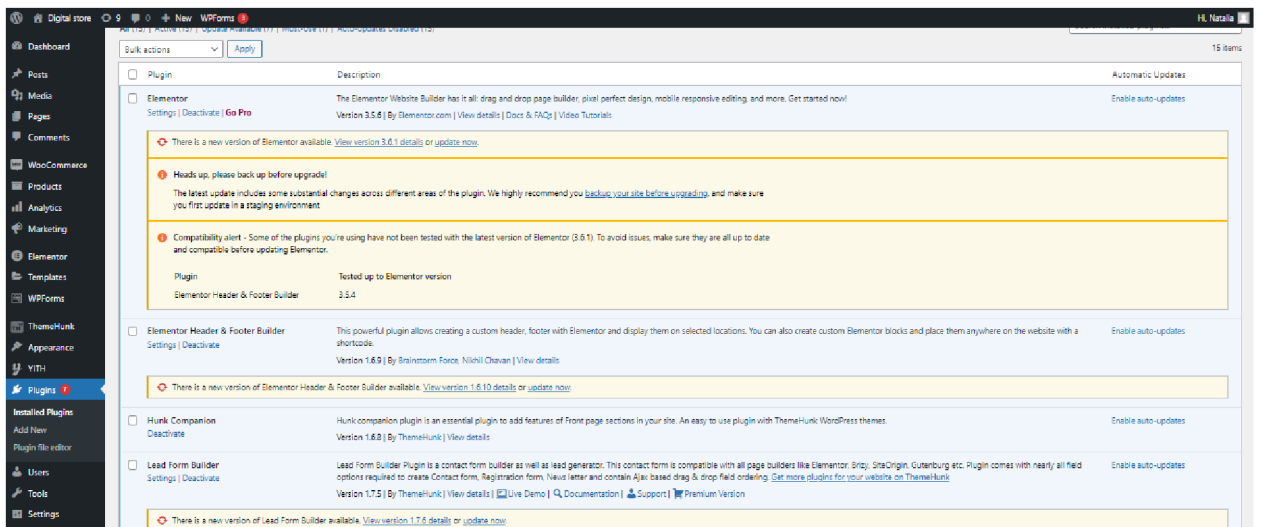


Figure 45 Plugins

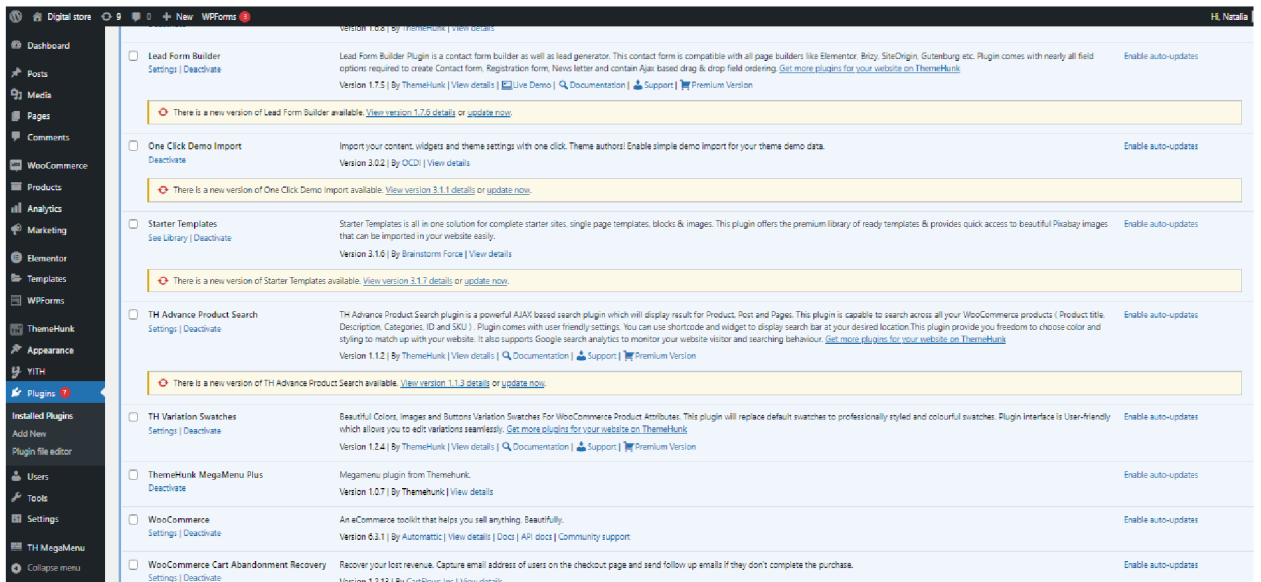


Figure 46 Plugins



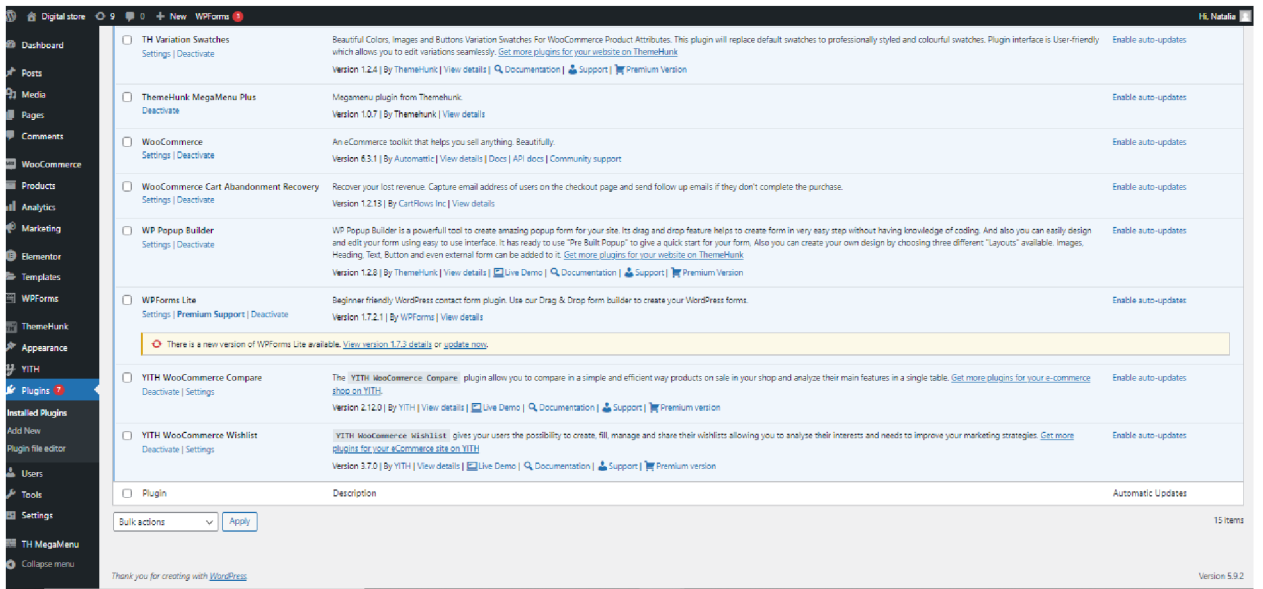


Figure 47 Plugins

### 4.5.1 Store description

The main tab in the admin panel where all the main data on the store is located is WooCommerce. The "Orders" category displays all orders. The administrator can change the order status or edit the order.

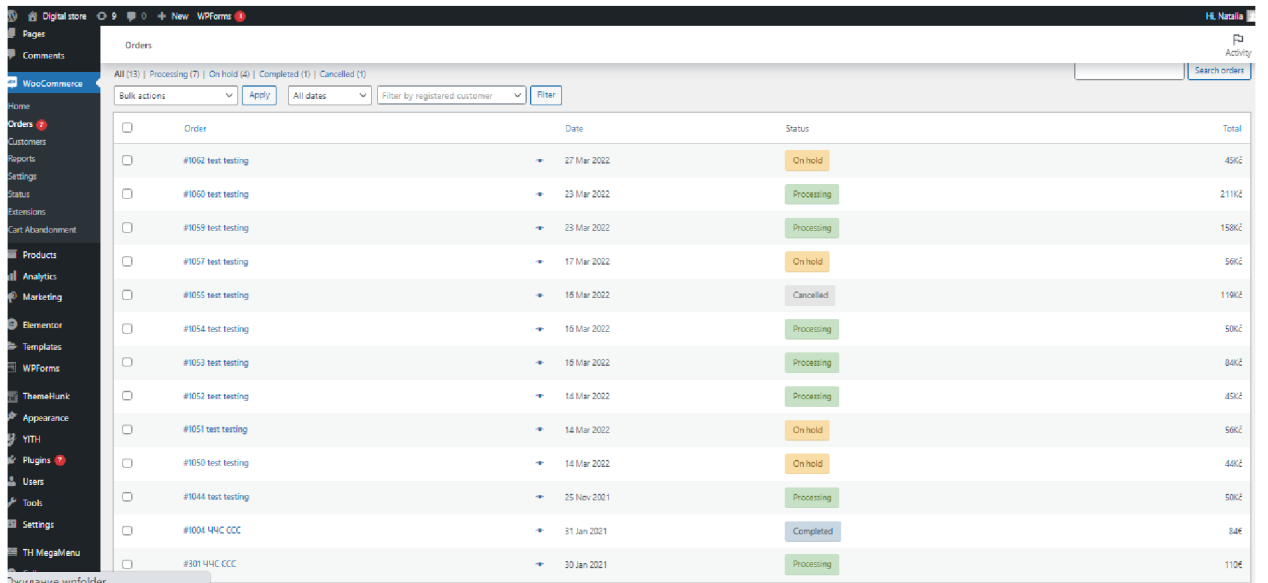


Figure 48 Orders

There is also a "Customers" tab. It displays a complete list of customers.

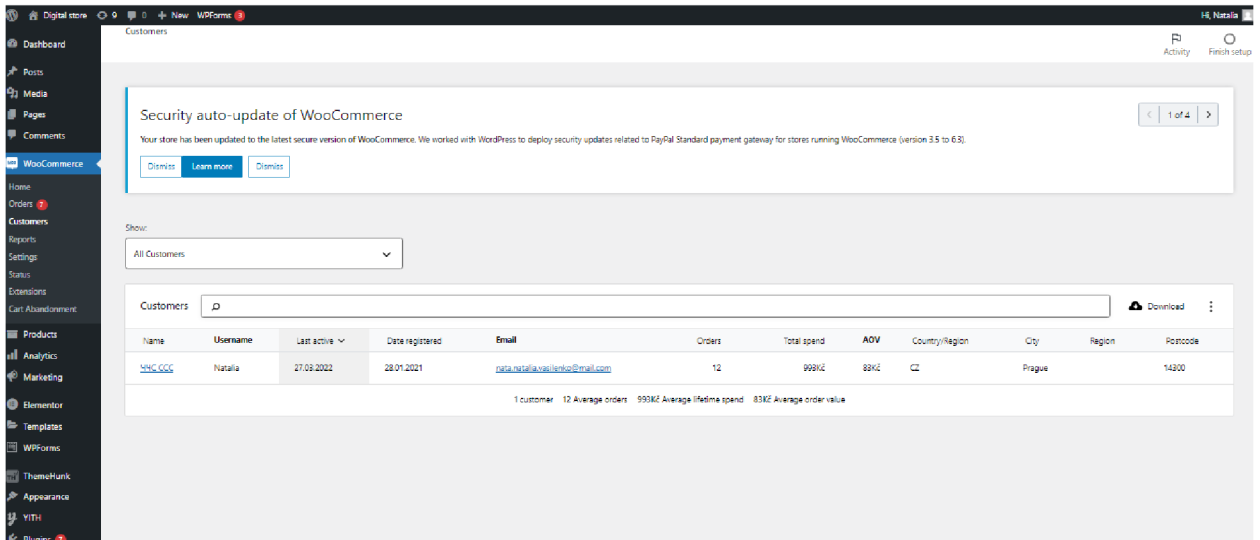


Figure 49 Customers

An example of the "Products" tab.

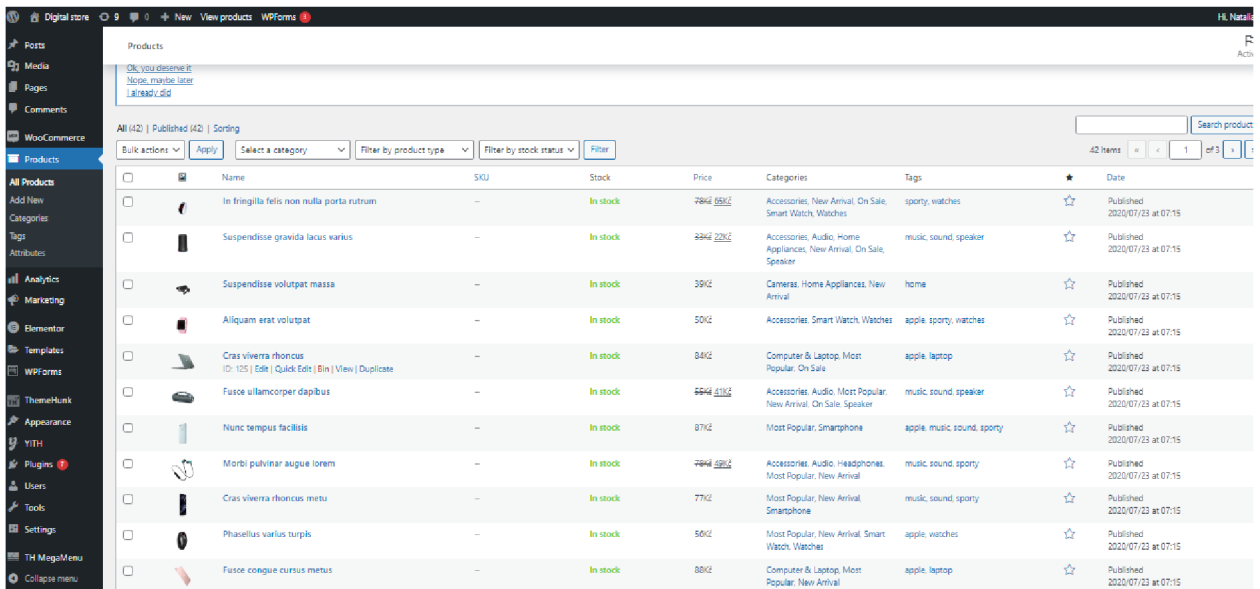


Figure 50 Products

The admin panel has many different functions. In the admin panel, you can make changes on the site, add and remove products, edit product descriptions, respond to reviews, make changes users, upload reports, set up analytics, and much more.

The main interface of the store.

The store has 6 categories. Each category has its own list of products. Products can be searched using the search bar.

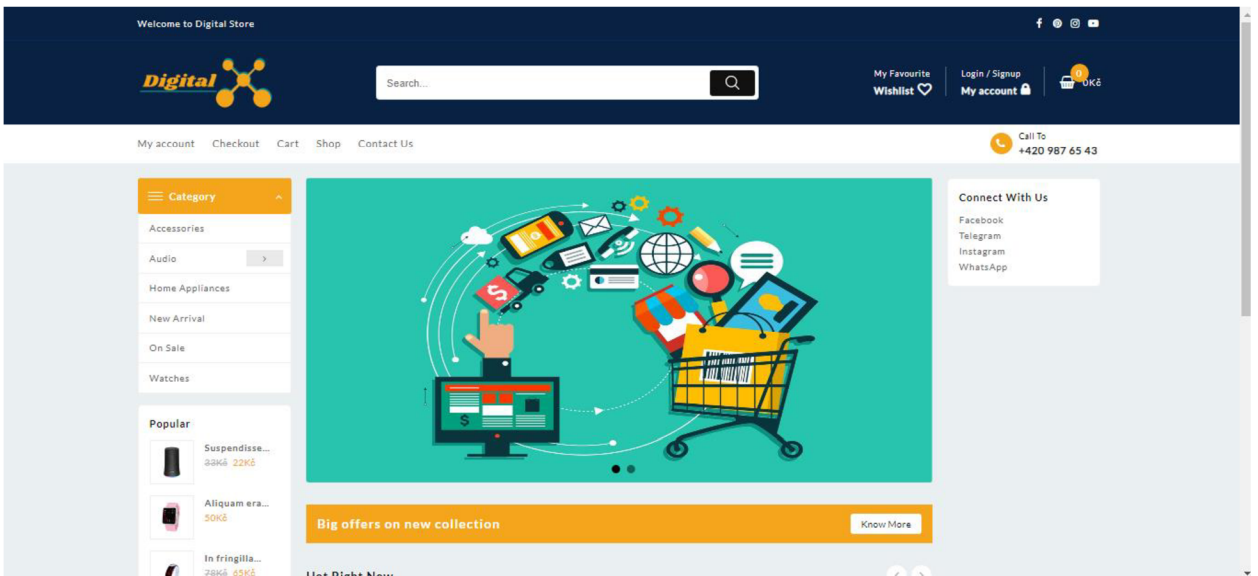


Figure 51 Main page

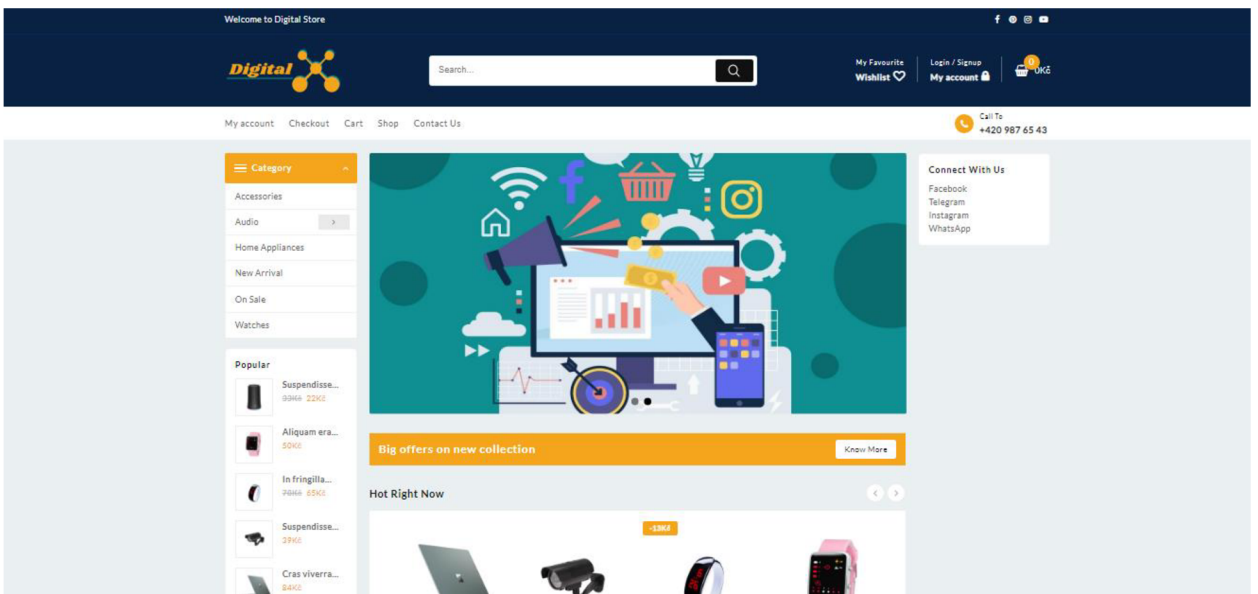


Figure 52 Main page

You can go to the product card through the catalog or open the mini product card.

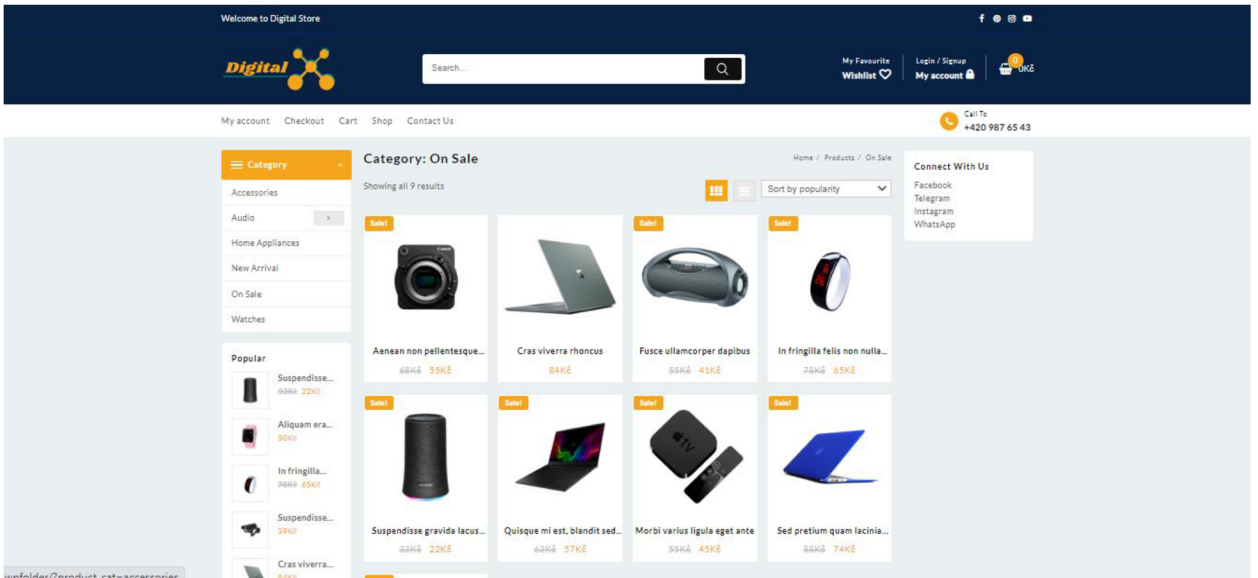


Figure 53 Catalog

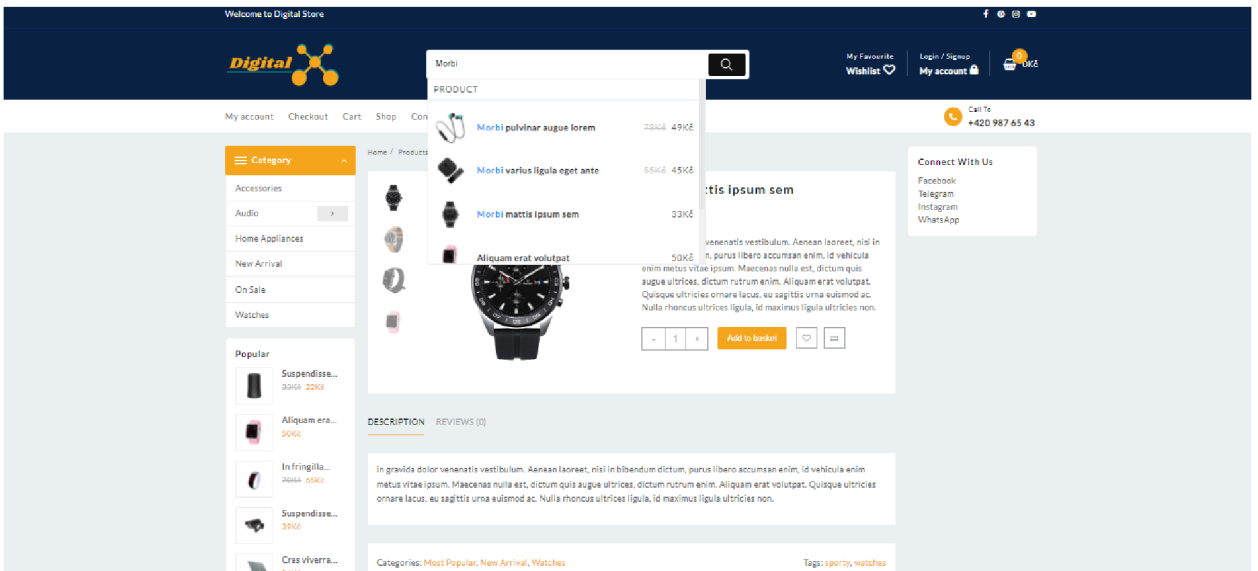


Figure 54 Card product and search example

To place an order, you must put the goods in the basket and click on the "Proceed to checkout" button in the basket. The buyer can also put the goods in favorites. From your favorites, you can add an item to your shopping cart.

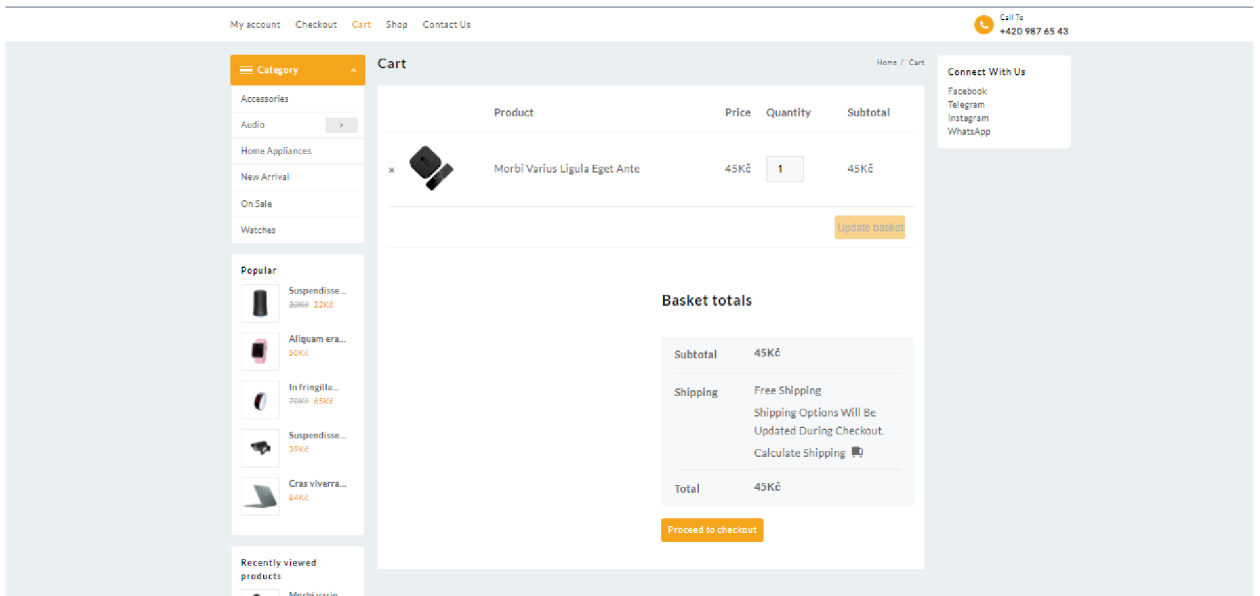


Figure 55 Cart

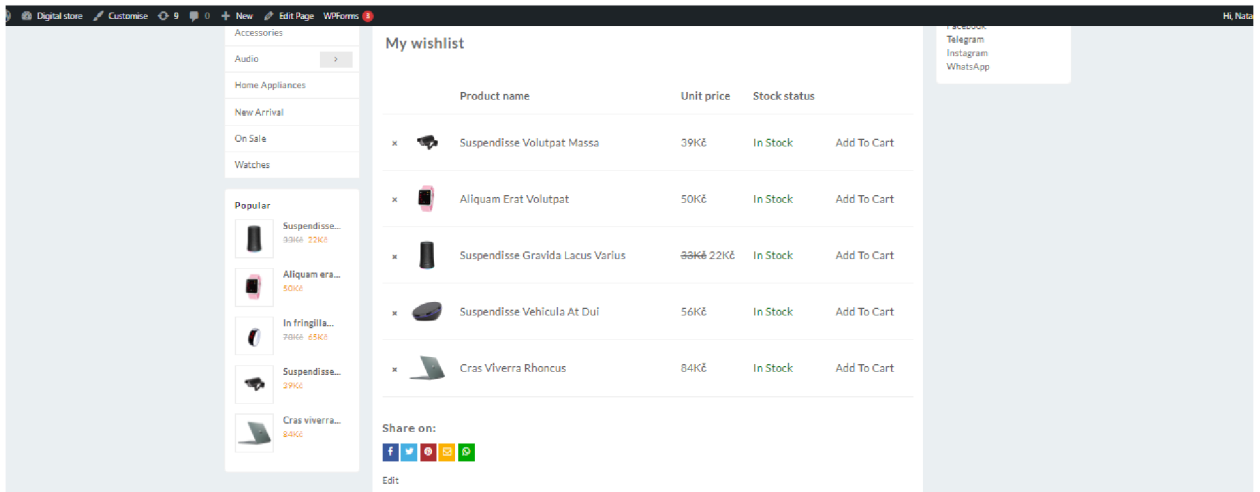


Figure 56 Wishlist

When placing an order, the buyer can choose the type of payment: cash or pay by card.

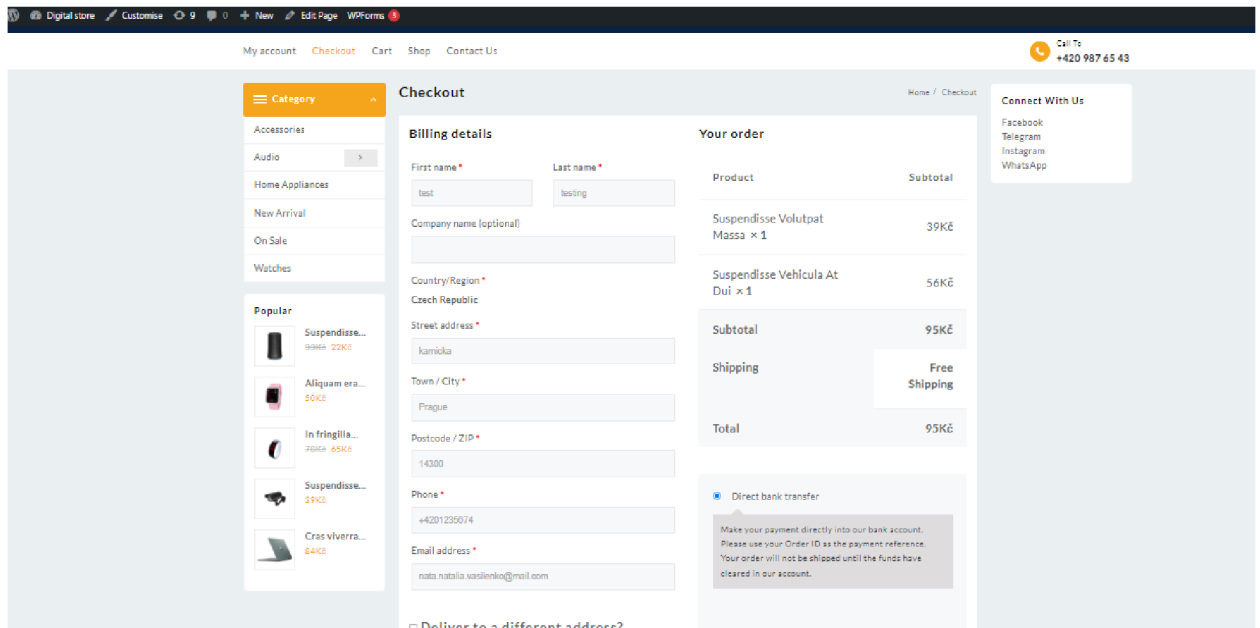


Figure 57 Checkout

After placing an order, the order number is returned to the buyer. This order will be displayed in the admin panel.

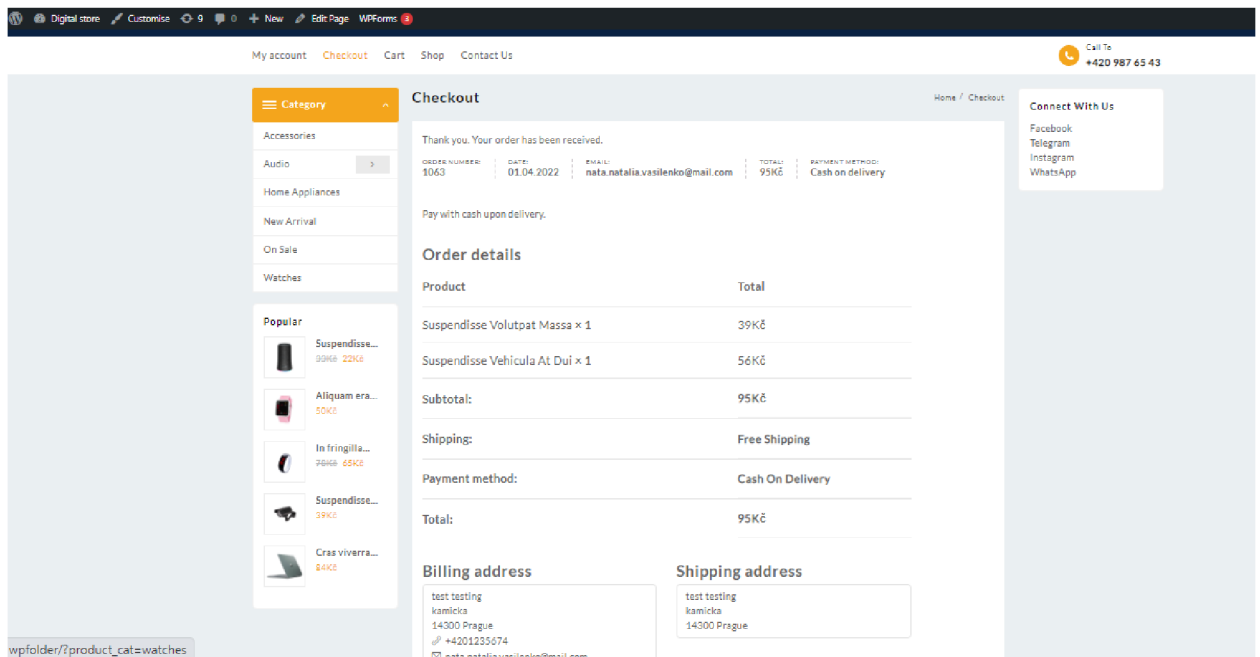


Figure 58 Number order

The buyer can leave a review about the product.

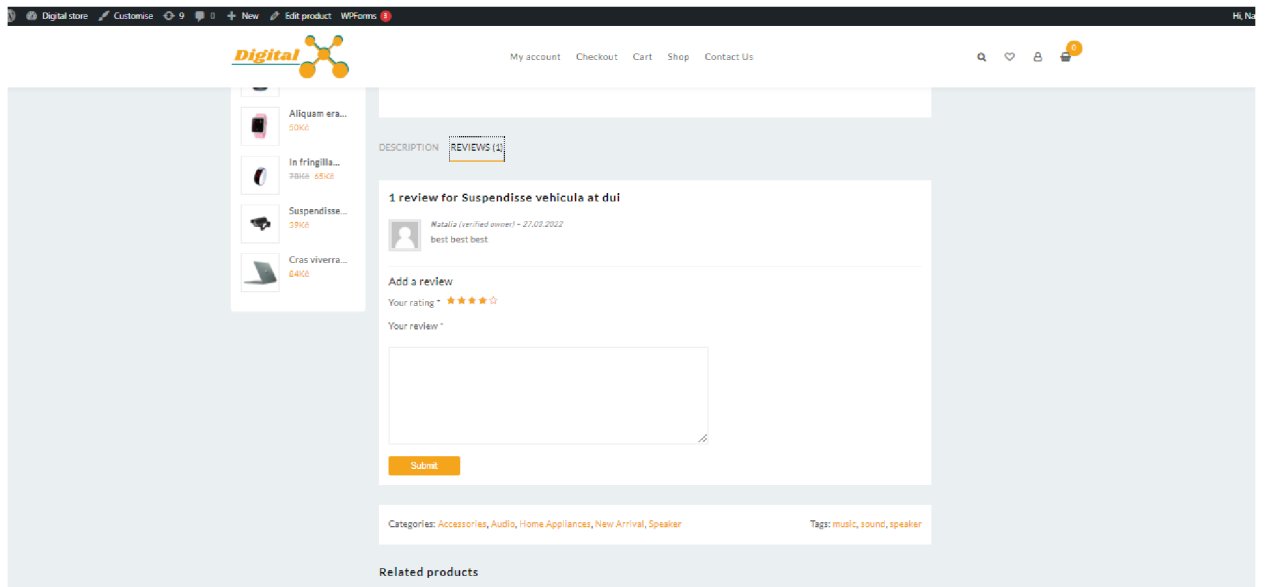


Figure 59 Review

The buyer can place an order by an authorized user and not authorized. He can also register and log in.

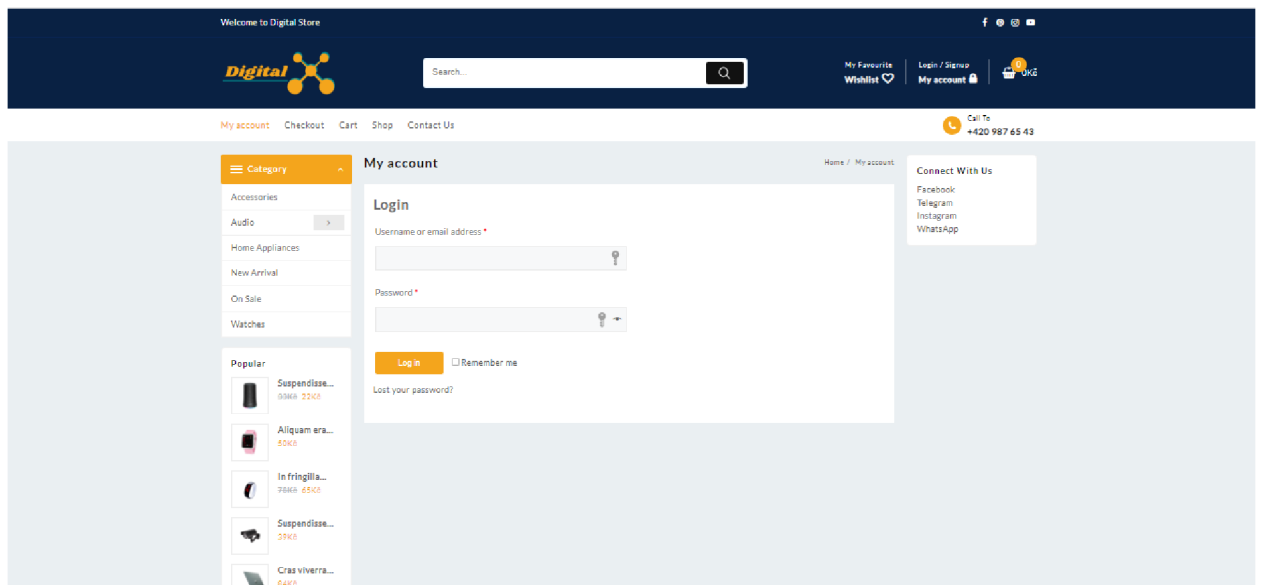


Figure 60 Login and register

## 4.6 Calculation of function point for store on wordpress

The calculation of function point is necessary in order to find out SLOC. For store on react, this calculation is not needed since the number of rows is known.

In order to calculate Value adjustment factor (VAF), it needs to know Total Degree of Influence (TDI)

Formula:

$$\text{VAF} = (\text{TDI} \times 0.01) + 0.65$$

Calculate Total Degree of Influence (TDI)

Analysis General System Characteristics

Characteristic	Description	Grade
Data communications	The software system is implemented as a single package on a stand-alone computer;	0
Distributed functions	The software system prepares data for final processing by other means, such as spreadsheets or DBMS;	1
Performance	No special performance requirements for the software system have been set by the user;	0
Heavily used configuration	There are no explicit or implicit restrictions on the use of equipment resources;	0
Transaction rate	Peak periods of transactions are not expected;	0
Online data entry	All transactions are processed in batch mode;	0
End user efficiency	Present from 1 to 3 items: <ul style="list-style-type: none"> <li>• navigation assistance (function keys, links, dynamic menus);</li> <li>• scrolling;</li> <li>• popup windows;</li> </ul>	1
Online update	Absent;	0
Complex processing	Absent;	0
Reusability	There is no reusable code in the Software System;	0
Installation ease	No special user requirements, and no special installation program is required;	0
Operational ease	No special user requirements except for the normal backup procedure;	0
Multiple sites	The software system is not designed to be used by more than one user or installed on more than one computer.	0
Facilitate change	The ability to organize flexible queries and reports that process simple requirements, for example, applying logical operations and/or to only one internal logical file (counts as 1 point);	1
	<b>Total</b>	<b>3</b>

*Table 1 Analysis General System Characteristics*

Total:

$$\text{VAF} = (3 \times 0.01) + 0.65 = 0,68$$

Calculate Unadjusted function points (UFP)

Difficulty levels for internal and external files (ILF and EIF) depending on the number of RET and DET are shown in the table



	1-19 DET	20-50 DET	51 and more DET
1 RET	Low	Low	Average
2-5 RET	Low	Average	High
6 more RET	Average	High	High

*Table 2 Difficulty levels for internal and external files (ILF and EIF)*

Difficulty levels for external inputs (EI) depending on the number of FTR and DET are shown in the table

	1 - 4 DET	5 - 15DET	16 and more DET
0 - 1 FTR	Low	Low	Average
2 FTR	Low	Average	High
3 more FTR	Average	High	High

*Table 3 Difficulty levels for external inputs (EI)*

Difficulty levels for external outputs (EO) depending on the number of FTR and DET are shown in the table

	1-5DET	6-19 DET	20 and more DET
0-1 FTR	Low	Low	Average
2-3 FTR	Low	Average	High
4 and more FTR	Average	High	High

*Table 4 Difficulty levels for external outputs (EO)*

Difficulty levels for external queries (EQ) are determined using the following simple algorithm:

- 1) The input of the external queries is treated similarly to the external input (EI) see tab. 3;
- 2) The output of the external inquiry is treated similarly to the external output (EO) see tab. 4;
- 3) As a result, the largest value obtained is used

The weighting coefficients of complexity levels for various characteristics of the functionality of the software system are given in Table 5.

	Low	Average	High
ILF	7	10	15
EIF	5	7	10
EI	3	4	6
EO	4	5	7
EQ	3	4	6

*Table 5 The weighting coefficients of complexity levels for various characteristics of the functionality of the software system*

Result:

Internal Logical File

The database contains over 6 types of data and up to 50 fields. All this is stored in one logical file, so Difficulty levels = 15 and Number=1

External Interface File

There are no external interface files.

External input

There are six external inputs (EI) in the software system: "Registration", "Checkout", "Contact us", "Leave a review", "Product search", "Save changes according to account data" and 15 DET.

Based on this difficulty level = Low (3)

External output

There are also four external outputs (EO) in the software system:

- Displaying a notification message when the data in the account is changed;
- Displaying a notification message when creating a product review if the rating is not completed;
- Display a notification message for order numbers when placing an order;
- Displaying a notification message when filling out the "Contact us" form.

The difficulty level of these external findings is low (4)

External Inquiry

There are four external query (EQ) in the software system:

- Output of the list of sorted goods;
- Output of the list of goods;
- Output of the list of orders;
- Output of the product card.

The difficulty level of these external requests is low (3)

Function type	Number	Difficulty levels
Internal Logical File	1	15
External Interface File	0	5
External Input	6	3
External Output	4	4
External Inquiry	4	3

*Table 6 Result*

$$UFP = (1*15)+(0*5)+(6*3)+(4*4)+(4*3) = 61$$

Function Points

$$FP = UFP * VAF$$

$$FP = 61 * 0,68 = 41$$

Finally, let's estimate the number of lines of source code (SLOC), based on the fact that the program needs to be developed using the PHP and SQL programming language. According to the data from the table, the average value between PHP and SQL was found =  $(53+21)/2 = 37$ . The solution for finding the average value between PHP and SQL was taken from the article (Barry Boehm, 2002). For PHP, the value Javascript =53 was taken. The data was taken from the table (QSM Resources, 2022)

$$\text{SLOC} = \text{FP} * \text{LF}$$

$$\text{SLOC} = 41 \times 37 = 1517$$

Thus, the store on Wordpress contains approximately 1517 lines of source code in PHP and SQL.

#### 4.7 Calculation COCOMO

Basic model was taken Base model, since most of the technical characteristics described in COCOMO II will not be implemented. It takes a lot of time to implement them.

The Organic system was taken as a basis, since the team size is small (1 person) and the team members have little experience in solving the problem.

##### React store

In the code, the number of rows was counted along with the Database = 758

The values for a, b, c and d were taken from the table 7.

Software projects	a	b	c	d
Organic	2,4	1,05	2,5	0,38
Semi - detached	3,0	1,12	2,5	0,35
Embedded	3,6	1,2	2,5	0,32

*Table 7 The values for a, b, c and d*

$$\text{Effort} = a * (\text{KLOC})^b = 2,4 * 0,758^{1,05} = 1,8 \text{ person - months}$$

$$\text{Time} = c * (\text{Effort})^d = 2,5 * 1,8^{0,38} = 3,12 \text{ months}$$

$$\text{Personrequired} = \text{Effort} / \text{Time} = 1,8 / 3,12 = 0,57 \text{ person}$$

##### Store on WordPress

From the calculation of the functional point, KLOC was obtained = 1517

$$\text{Effort} = a * (\text{KLOC})^b = 2,4 * 1,517^{1,05} = 3,71 \text{ person - months}$$

$$\text{Time} = c * (\text{Effort})^d = 2,5 * 3,71^{0,38} = 4,11 \text{ months}$$

$$\text{Personrequired} = \text{Effort} / \text{Time} = 3,71 / 4,11 = 0,9 \text{ person}$$

Since the WordPress store was created with the source code, the necessary plugins were added and changes were made to the design, content and all the necessary information were added, the settings were adjusted, but no code was required, which means that it took about 20% of the received data to create the store.

$$\text{Effort} = 3,71 * 0,2 = 0,74 \text{ person - months}$$

$$\text{Time} = 4,11 * 0,2 = 0,82 \text{ months}$$

$$\text{Personrequired} = 0,9 * 0,2 = 0,18 \text{ person}$$

Based on the data obtained, the table shows the results

	React	Wordpress
Effort	1.8	0.74
Time	3.12	0.82
Personrequired	0.57	0.18

*Table 8 Results*

Based on the data obtained, it can be seen that creating the store on WordPress requires 3.8 times less time than for the store on React, 2.4 times less Effort than for the store on React and 3.1 times less human resources than for the store on React .

## 4.8 ISO/IEC 9126 software quality standards

Below is a comparison of six characteristics that describe software quality.

Portability

React	Wordpress
Easy installation and portability of the project on another computer with different operating systems.	Reuse is not possible due to the fact that WordPress is not modular. Difficult transfer to another computer or hosting. When installed locally, WordPress depends on a local web server. Each operating system has its own list of available local servers. There are configuration issues when installing on a different computer and a different local server.

*Table 9 Portability*

Functional Suitability

React	Wordpress
Library for creating dynamic and interactive user interfaces. It has many possibilities for implementing different levels of interfaces.	A tool that allows you to create any type of website, personal blog, magazine or news site, online community, application without writing code.

<p>Creation of functional reusable components to reduce time on repetitive blocks.</p> <p>Reuse of components.</p> <p>Can fine-tune the components.</p> <p>Can be customized to customer requirements.</p> <p>Reduces coding.</p> <p>Flexible.</p> <p>More adaptive.</p> <p>DOM usage.</p> <p>Provides easy migration between versions.</p> <p>Requires additional modules or libraries for full functionality.</p> <p>Clear code structure.</p> <p>Optimize (SEO).</p>	<p>Allows you to edit and manage a website from the back end without learning the code.</p> <p>Its features include a plugin architecture and a templating system called themes in WordPress.</p> <p>Possibility of management under different roles.</p> <p>Works in all browsers.</p> <p>Free, but there is also a premium.</p> <p>There are various paid and free plugins, themes.</p> <p>Difficult to customize to specific customer requirements.</p> <p>Responsive layout is enabled by default.</p> <p>There are optimization tools (SEO)</p> <p>Open source code that can be improved, made changes.</p> <p>Not flexible. It's difficult to add additional functionality through code, as there can be many conflicts.</p>
---	--

*Table 10 Functional Suitability*

Reliability

React	Wordpress
<p>Recoverability: There is no backup to react.</p> <p>Fault tolerance: React uses "Fuses". Fuses are React components that catch JavaScript errors anywhere in their child component trees, store them in the error log, and output a fallback UI in place of the collapsed component tree.</p> <p>React.js does not have default security settings.</p> <p>The JSEncrypt library is used to encrypt data</p>	<p>Recoverability: Mysql have backup. Deleted files can be recovered.</p> <p>Fault tolerance: WordPress has special security plugins.</p> <p>WordPress has:</p> <ul style="list-style-type: none"> <li>• Password Management</li> <li>• Role based security</li> <li>• Groups support</li> <li>• Encryption Options:</li> <li>• Network Data – SSL</li> <li>• Password-MD5</li> </ul>

*Table 11 Reliability*

Efficiency

React	WordPress
<p>Fast.</p> <p>Reloading is not required to navigate to other pages of the site.</p> <p>Fast event handling.</p> <p>React allows to significantly reduce the code, which contributes to fast loading.</p> <p>High efficiency and performance due to the Virtual DOM.</p> <p>React and SEO work well together.</p> <p>Applications and content developed with React are much more scalable.</p>	<p>Not resistant to heavy loads.</p> <p>Slow loading.</p> <p>WordPress plugins make websites run slow and buggy. WordPress is also notorious for not supporting large or multiple images or significant amounts of text. A large number of irrelevant common code reduces the site's fast loading.</p> <p>Limited SEO features.</p> <p>WordPress takes up a lot of Random Access Memory (RAM).</p>

	Applications and content developed with WordPress are less scalable.
--	--

*Table 12 Efficiency*

Usability

React	WordPress
Problems updating documentation for new technologies. Expensive hosting. Installation is complicated and requires prior knowledge of JavaScript, HTML, CSS to understand and Virtual DOM. React has no interface since programs are created using code. React offers a wide range of design options.	Limitations in the individuality of the design, to make changes you need to resort to the code. Cheap hosting. Easy Installation. Easy barrier to entry. Intuitive interface. Convenient admin panel. Complex code page structure.

*Table 13 Usability*

Maintainability

React	Wordpress
Full compatibility between versions. It is possible to connect libraries of different versions to the application, update obsolete properties with inheritance.	Wordpress updates are a big problem. WordPress requires frequent updates to work. New updates and plugins often cause bugs and code conflicts.

*Table 14 Maintainability*

Advantages and disadvantages

Based on the analysis of the six characteristics of the ISO/IEC 9126 software quality standards, the main advantages and disadvantages of React and WordPress are derived:

React	WordPress
Advantages: <ul style="list-style-type: none"> <li>• Easy installation and portability</li> <li>• Reuse of components.</li> <li>• Can be customized to customer requirements.</li> <li>• Reduces coding and clear code structure</li> <li>• Flexible.</li> <li>• More adaptive.</li> <li>• DOM usage.</li> <li>• Provides easy migration between versions.</li> <li>• The JSEncrypt library is used to encrypt data</li> <li>• Reloading is not required to navigate to other pages of the site.</li> <li>• High efficiency and performance due to the Virtual DOM.</li> </ul>	Advantages: <ul style="list-style-type: none"> <li>• Allows you to create any type of website, personal blog, magazine or news site, online community, application and edit, manage a website from the back end without learning the code.</li> <li>• Possibility of management under different roles.</li> <li>• Works in all browsers.</li> <li>• Free, but there is also a premium.</li> <li>• There are various paid and free plugins, themes.</li> <li>• Responsive layout is enabled by default.</li> <li>• There are optimization tools (SEO)</li> <li>• Mysql have backup. Deleted files can be recovered.</li> </ul>

<ul style="list-style-type: none"> <li>• React and SEO work well together.</li> <li>• Applications and content developed with React.js are much more scalable.</li> <li>• React offers a wide range of design options.</li> </ul>	<ul style="list-style-type: none"> <li>• WordPress has: <ul style="list-style-type: none"> <li>○ Password Management;</li> <li>○ Role based security;</li> <li>○ Groups support;</li> <li>○ Encryption Options;</li> <li>○ Network Data – SSL;</li> <li>○ Password-MD5.</li> </ul> </li> <li>• Cheap hosting.</li> <li>• Easy Installation.</li> <li>• Easy barrier to entry.</li> <li>• Intuitive interface.</li> <li>• Convenient admin panel.</li> </ul>
<p>Disadvantages:</p> <ul style="list-style-type: none"> <li>• Vulnerable to attacks</li> <li>• Requires additional modules or libraries for full functionality.</li> <li>• There is no backup to react.</li> <li>• React.js does not have default security settings.</li> <li>• Problems updating documentation for new technologies.</li> <li>• Expensive hosting.</li> <li>• Installation is complicated and requires prior knowledge of JavaScript, HTML, CSS to understand and Virtual DOM.</li> <li>• React has no interface since programs are created using code.</li> <li>• Full compatibility between versions. It is possible to connect libraries of different versions to the application, update obsolete properties with inheritance.</li> </ul>	<p>Disadvantages:</p> <ul style="list-style-type: none"> <li>• Difficult transfer to another computer or hosting.</li> <li>• Vulnerable to attacks</li> <li>• Difficult to customize to specific customer requirements.</li> <li>• Not flexible. It is difficult to add additional functionality through code, as there can be many conflicts.</li> <li>• Not resistant to heavy loads.</li> <li>• Slow loading.</li> <li>• Limited SEO features.</li> <li>• WordPress takes up a lot of Random Access Memory (RAM).</li> <li>• Applications and content developed with WordPress are less scalable.</li> <li>• Limitations in the individuality of the design, to make changes you need to resort to the code.</li> <li>• Complex code page structure.</li> <li>• WordPress requires frequent updates to work.</li> <li>• New updates and plugins often cause bugs and code conflicts.</li> </ul>

*Table 15 Advantages and disadvantages*

## 5. Results and Discussion

From the above analysis and calculations, we can see that React and WordPress solve different problems. It needs to separate the goals for which need to use WordPress or React. Each has its own specifics.

Complex and large projects that require a lot of dynamics on WordPress will not be able to be done, for this React is suitable.

Smaller projects that do not have strict requirements and require quick implementation are suitable for WordPress.

React requires more financial costs, human resources and time, but it is more flexible and allows to fulfill all the requirements of customers. One of the plus points is performance. Be sure to have at least basic knowledge of JavaScript, HTML and CSS. React is not a constructor and for it need to write code.

WordPress is less flexible, but allows to create a store, website, blog, etc. without knowledge of programming languages. It is free. With WordPress, it is hard to create a custom design. When injecting code, there are many conflicts. It has poor performance, but a clear interface.

Under certain conditions, you can combine WordPress and React in one project, but this requires more human resources.



## **6. Conclusion**

The main task of the thesis was to create an online digital technology store using React and WordPress and compare two build structures using React and WordPress. The theoretical part describes the differences in the principles of construction of these two tools. Software development standards such as UML and ISO/IEC 9126 software quality standards are described.

In the practical part, differences in the functioning and usability of the site in the development of the backend and user-friendly interface were revealed. The advantages and disadvantages of these two tools are described. Two digital technology online stores were created using React and WordPress and evaluated in terms of ease of use, operation and maintenance according to the ISO / IEC 9126 standard and the COCOMO model. All goals were achieved and a comparative analysis was carried out.

## 7. References

Bonnie Eisenman. Learning react native: building native mobile apps with JavaScript. Second edition. Boston: O'Reilly, 2018. ISBN 1491989149

Jonathan Lebensold. React native cookbook: bringing the web to native platforms. Sebastopol: O'Reilly, 2018. ISBN 978-1-491-99384-2

Luke & Laura THOMPSON. PHP and MySQL Web Development, Sams Publishing 2003, ISBN 80-86497-60-7

Alex Banks, Eve Porcello, Learning React. Released May 2017. Publisher(s): O'Reilly Media, Inc

Sinan Si Alhir, Learning UML. Editor: Jonathan Gennick, Copyright © 2003 O'Reilly Media, Inc.

Martin Fowler, UML Distilled, 2003

Wikipedia, 2009, [Online] [https://en.wikipedia.org/wiki/Class\\_diagram](https://en.wikipedia.org/wiki/Class_diagram)

MockAPI, 2022, [Online] <https://mockapi.io/docs>

Warakorn Jetlohasiri, 2020, COCOMO: A procedural cost estimate model for software projects [Online] <https://medium.com/@warakornjetlohasiri/cocomo-a-regression-model-in-procedural-cost-estimate-model-for-software-projects-65ab5222a1f5>

Wikipedia, 2015, [Online] <https://en.wikipedia.org/wiki/COCOMO>

GeeksforGeeks, 2022, [Online] <https://www.geeksforgeeks.org/software-engineering-cocomo-model/?ref=lbp>

Pratt, 2022, [Online] <https://sites.google.com/site/prattshomepage/home/estimation-overview/function-point-analysis/fp-counting-process/value-adjustment-factor>

Cost Estimation, 2022, [Online] <https://people.cs.ksu.edu/~padmaja/Project/CostEstimate.htm>

Codex, 2022, [Online] [https://codex.wordpress.org/Database\\_Description](https://codex.wordpress.org/Database_Description)

QSM Resources, Function Point Languages Table, [Online] <https://www.qsm.com/resources/function-point-languages-table#MoreInfo>

Barry Boehm, 2002, Cost estimation with COCOMO II, [Online] [https://www.researchgate.net/publication/228600814\\_Cost\\_estimation\\_with\\_COCOMO\\_II](https://www.researchgate.net/publication/228600814_Cost_estimation_with_COCOMO_II)

Context, 2022, [Online] <https://reactjs.org/docs/context.html>

## 8. Appendix

### Folder “Card”

#### Card. Module.scss

```
src > components > Card > Card.module.scss > .card
1 .card{
2   border: 1px solid #d9d9d9;
3   padding: 30px;
4   width: 220px;
5   border-radius: 40px;
6   margin-right: 30px;
7   margin-bottom: 30px;
8   transition: box-shadow 0.2s ease-in-out, transform 0.2s ease-in-out;
9
10
11  &:hover{
12    box-shadow: 0px 20px 35px rgba(0, 0, 0, 0.12);
13    transform: translateY(-5px);
14  }
15
16  .favorite{
17    position: absolute;
18    margin-left: 130px;
19    cursor: pointer;
20  }
21
22  span{
23    font-size: 13px;
24    opacity: 0.5;
25    text-transform: uppercase;
26  }
27
28  b{
29    font-size: 13px;
30  }
31
32  h5{
33    font-weight: 400;
34    font-size: 14px;
35  }
36
37  .plus{
38    cursor: pointer;
39  }
```

#### Index.js

```
src > components > Card > index.js > Card
1 import styles from './Card.module.scss'
2 import ContentLoader from 'react-content-loader';
3 import AppContext from '../context';
4
5 import React from 'react';
6
7 function Card({
8   onFavorite,
9   onPlus,
10  id,
11  title,
12  imageUrl,
13  price,
14  favorited = false,
15  added = false,
16  loading = false
17 }) {
18   const { setIsAdded } = React.useContext(AppContext);
19   const [isFavorite, setIsFavorite] = React.useState(favorited);
20
21   const onClickPlus = () => {
22     onPlus({ id, title, imageUrl, price });
23   };
24
25   const onClickFavorite = () => {
26     onFavorite({ id, title, imageUrl, price });
27     setIsFavorite(!isFavorite);
28   };
29
30   return (
31     <div className={styles.card}>
32       {
33         loading ? <ContentLoader
34           speed={21}
35           width={150}
36           height={200}
37           viewBox="0 0 150 200"
38           backgroundColor="#f9f9f9"
39           foregroundColor="#ccc"
40         />
41         : <div style={{
42             width: 150px, height: 200px,
43             display: flex, flex-direction: column, align-items: center, justify-content: center,
44             gap: 10px, text-align: center, font-family: sans-serif, font-size: 14px,
45             background-color: #f9f9f9, border-radius: 40px, padding: 30px 0 0 0,
46             border: 1px solid #d9d9d9, transition: box-shadow 0.2s ease-in-out, transform 0.2s ease-in-out;
47             box-shadow: 0px 20px 35px rgba(0, 0, 0, 0.12);
48           }}>
```

```

REACT-STORE src > components > Card > # index.js > Card
  build 35 | width={150}
  > node_modules 36 | height={200}
  > public 37 | viewBox="0 0 150 200"
  > src 38 | backgroundColor="#F3F3F3"
  > components 39 | foregroundColor="#ececbeb"
  > Card 40 |
  > Card.module.scss 41 |
  # index.js 42 | <rect x="0" y="0" rx="10" ry="10" width="150" height="200" />
  # Drawer.js 43 | <rect x="2" y="103" rx="5" ry="5" width="150" height="15" />
  # Header.js 44 | <rect x="2" y="126" rx="5" ry="5" width="100" height="15" />
  # Info.js 45 | <rect x="2" y="150" rx="5" ry="5" width="80" height="25" />
  > pages 46 | <rect x="116" y="145" rx="10" ry="10" width="32" height="32" />
  > Favorites.js 47 | </ContentLoader> : <>
  > Home.js 48 | <div className={styles.favorite} onClick={onClickFavorite}>
  > App.js 49 | <img src={isFavorite ? '/img/liked.png' : '/img/unliked.png'} alt="unliked"/>
  > context.js 50 | </div>
  > index.js 51 | <img width="110" height="95" src={imageUrl} alt="iphone13.jpg" />
  > index.scss 52 | <h3>{title}</h3>
  > gitignore 53 | <div className="d-flex justify-between align-center">
  > context.js 54 | <span>Price:</span>
  > package-lock.json 55 | <b>{price}</b>
  > package.json 56 | </div>
  > README.md 57 | <img
  58 |   className={styles.plus}
  59 |   onClick={onClickPlus}
  60 |   src={isItemAdded(id) ? '/img/btn-checked.png' : '/img/btn-plus.svg'}
  61 |   alt="Plus"
  62 | />
  63 | </div>
  64 | </>
  65 | }
  66 |
  67 | </div>
  68 | );
  69 | }
  70 | export default Card;
  71 |
  72 |

```

## Drawer.js

```

REACT-STORE src > components > # Drawer.js > Drawer
  build 1 | import React from 'react';
  > node_modules 2 | import Info from './Info';
  > public 3 | import axios from 'axios';
  > src 4 |
  > components 5 | import AppContext from './context';
  > Card 6 |
  > Card.module.scss 7 |
  # Drawer.js 8 | const delay = (ms) => new Promise((resolve) => setTimeout(resolve, ms));
  # Header.js 9 |
  # Info.js 10 | function Drawer({onClose, onRemove, items = []}) {
  > pages 11 |   const [cartItems, setCartItems] = React.useState(AppContext);
  > Favorites.js 12 |   const [orderId, setOrderId] = React.useState(null);
  > Home.js 13 |   const [isLoading, setIsLoading] = React.useState(false);
  > App.js 14 |   const [isOrderComplete, setIsOrderComplete] = React.useState(false);
  > context.js 15 |   const [totalPrice, setTotalPrice] = React.useState(0);
  > index.js 16 |   const totalPrice = cartItems.reduce((sum, obj) => obj.price + sum, 0);
  > index.scss 17 |
  > gitignore 18 |
  > context.js 19 |   const onClickOrder = async () => {
  > package-lock.json 20 |     try {
  > package.json 21 |       setIsLoading(true);
  > README.md 22 |       const {data} = await axios.post('https://6185878923a2fe0017fff680.mockapi.io/orders', {
  23 |         items: cartItems,
  24 |       });
  25 |       setOrderId(data.id);
  26 |       setIsOrderComplete(true);
  27 |       setCartItems([]);
  28 |       for(let i = 0; i < cartItems.length; i++) {
  29 |         const item = cartItems[i];
  30 |         await axios.delete('https://6185878923a2fe0017fff680.mockapi.io/cart/' + item.id);
  31 |         await delay(1000);
  32 |       }
  33 |     } catch(error) {
  34 |       alert('Failed to create order');
  35 |       console.error(error);
  36 |     }
  37 |     setIsLoading(false);
  38 |   };
  39 |
  40 |   return (
  41 |     <div className="overlay">
  42 |       <div className="drawer">
  43 |         <h2 className="d-flex justify-between mb-30">
  44 |           Cart </h2>
  45 |
  46 |           {
  47 |             items.length > 0 ?
  48 |             <div className="d-flex flex-column flex">
  49 |               <div className="items">
  50 |                 {items.map((obj) => (

```

```

src > components > JS Drawerjs > Drawer
43 Cart </li>;
44
45 {
46   items.length > 0 ?
47   <div className="d-flex flex-column flex">
48     <div className="items">
49       {items.map((obj) => (
50         <div key={obj.id} className="cartItem d-flex align-center">
51           <div style={{backgroundImage: `url(${obj.imageUrl})`}}
52             className="cartItemImage"></div>
53         <div className="mr-20 flex">
54           <p className="mb-0">{obj.title}</p>
55           <p>{obj.price}</p>
56         </div>
57         <img onClick={() => onRemove(obj.id)} className="removeBtn" src="/img/remove.svg" alt="remove" />
58       </div>
59     </div>
60   </div>
61   <div className="cartTotalBlock">
62     <ul>
63       <li>
64         <span>Total</span>
65       </li>
66     </ul>
67     <p>{totalPrice}</p>
68   </div>
69   <button disabled={isLoading} onClick={onClickOrder} className="orangeButton">Checkout</button>
70 </div>
71 </div>
72 </div>
73 <Info
74   title={isOrderComplete ? "Order is processed" : "Cart empty"}
75   description={
76     isOrderComplete
77     ? `Order# ${orderId}`
78     : "Add something"
79   }
80   image={isOrderComplete ? "/img/complete-order.jpg" : "/img/empty-cart.jpg"}
81 </div>
82 </div>
83 </div>
84 </div>
85 </div>
86 </div>
87 </div>
88 </div>
89
90 export default Drawer;

```

## Header.js

```

src > components > JS Headerjs > Header
1 import React from 'react';
2 import { Link } from 'react-router-dom';
3 import AppContext from '../context';
4
5 function Header(props) {
6   const { cartItems } = React.useContext(AppContext);
7   const totalPrice = cartItems.reduce((sum, obj) => obj.price + sum, 0);
8
9   return(
10     <header className="d-flex justify-between align-center p-40">
11       <Link to="/" />
12       <div className="d-flex align-center">
13         
14         <div>
15           <h3 className="text-uppercase">Digital Shop</h3>
16           <p className="opacity-5">Best digital store</p>
17         </div>
18       </div>
19       <Link>
20         <ul className="d-flex">
21           <li onClick={props.onClickCart} className="mr-30 cu-p">
22             
23             <span>{totalPrice}</span>
24           </li>
25           <li className="mr-20 cu-p">
26             <Link to="/favorites" />
27           </li>
28           <li>
29             
30           </li>
31           <li>
32             
33           </li>
34         </ul>
35       </header>
36     </div>
37   );
38 }
39
40 export default Header;

```

## Info.jsx

```

src > components > JS Infojsx > Info
1 import React from 'react';
2 import AppContext from '../context';
3
4 const Info = ({title, image, description}) => {
5   const {setCartOpened} = React.useContext(AppContext);
6
7   return (
8     <div>
9       <div className="cartEmpty d-flex align-center justify-center flex-column flex">
10        <img className="mb-20 width:120px height:120px src={image} alt="empty" />
11        <h2>{title}</h2>
12        <p className="opacity-0">{description}</p>
13        <button onClick={() => setCartOpened(false)} className="orangeButton">Back</button>
14      </div>
15    </div>
16  );
17 }
18
19 export default Info;

```

## Folder "Pages"

## Favorites.jsx

```
src > pages > Favorites.jsx ...
1 import React from 'react';
2 import Card from '../components/Card/index';
3 import AppContext from '../context';
4
5 function Favorites() {
6
7
8     const { favorites, onAddToFavorite } = React.useContext(AppContext);
9
10
11     return(
12     <div className="content p-40">
13     <div className="d-flex align-center justify-between mb-40">
14     <h1>My favorites</h1>
15
16     </div>
17
18     <div className="d-flex flex-wrap">
19     {favorites.map((item) => (
20     <Card
21     id={item.id}
22     title={item.title}
23     price={item.price}
24     imageUrl={item.imageUrl}
25     key={item.title}
26     favorited={true}
27     onFavorite={onAddToFavorite}
28     />
29     ))}
30     </div>
31     </div>
32     )
33     export default Favorites;
```

## Home.jsx

```
src > pages > Home.jsx > default
1 import React from 'react';
2 import Card from '../components/Card/index';
3 //import AppContext from '../context';
4 function Home({ items,
5     cartItems,
6     searchValue,
7     setSearchValue,
8     onChangeSearchInput,
9     onAddToFavorite,
10    onAddToCart,
11    isLoading
12 }) {
13     //const { isItemAdded } = React.useContext(AppContext);
14     const renderItem = () => {
15         const filteredItems = items.filter((item) =>
16             item.title.toLowerCase().includes(searchValue.toLowerCase()),
17         );
18         return (isLoading ? [...Array(12)] : filteredItems).map((item, index) => (
19             <Card
20             key={index}
21             onFavorite={(obj) => onAddToFavorite(obj)}
22             onPlus={(obj) => onAddToCart(obj)}
23             loading = {isLoading}
24             {...item}
25             />
26         ));
27     };
28     return(
29     <div className="content p-40">
30     <div className="d-flex align-center justify-between mb-40">
31     <h1>{searchValue ? `Search by "${searchValue}"` : 'All products'}</h1>
32     <div className="search-block">
33     
34     {searchValue && (
35     <img
36     onClick={() => setSearchValue('')}
37     className="clear cu-p"
38     src="/img/remove.svg"
39     alt="clear"
40     />
41     )}
42     <input onChange = {onChangeSearchInput} value={searchValue} placeholder="Search..." />
43     </div>
44     </div>
45     <div className="d-flex flex-wrap">
46     {renderItem()}
47     </div>
48     </div>
49     </div>
50     )
```

# App.js

```
src > JS App.js > @ onAddToCart
1 import React from 'react';
2 import { Routes, Route } from 'react-router-dom';
3 import Home from './pages/Home';
4 import Drawer from './components/Drawer';
5 import Header from './components/Header';
6 import axios from 'axios';
7 import Favorites from './pages/Favorites';
8 import AppContext from './context';
9
10 function App() {
11   const [items, setItems] = React.useState([]);
12   const [cartItems, setCartItems] = React.useState([]);
13   const [favorites, setFavorites] = React.useState([]);
14   const [searchValue, setSearchValue] = React.useState("");
15   const [cartOpened, setCartOpened] = React.useState(false);
16   const [isLoading, setIsLoading] = React.useState(true);
17
18   React.useEffect(() => {
19     async function fetchData() {
20       try {
21         const cartResponse = await axios.get('https://6185878923a2fe0017fff680.mockapi.io/cart');
22         const favoritesResponse = await axios.get('https://6185878923a2fe0017fff680.mockapi.io/favorites');
23         const itemsResponse = await axios.get('https://6185878923a2fe0017fff680.mockapi.io/items');
24
25         setIsLoading(false);
26
27         setCartItems(cartResponse.data);
28         setFavorites(favoritesResponse.data);
29         setItems(itemsResponse.data);
30       } catch (error) {
31         alert('Error while requesting data');
32       }
33     }
34     fetchData();
35   }, []);
36
37   const onAddToCart = async (obj) => {
38     try {
39       if (cartItems.find(item => Number(item.id) === Number(obj.id)) {
40         setCartItems(prev => prev.filter(item => Number(item.id) !== Number(obj.id)));
41         await axios.delete('https://6185878923a2fe0017fff680.mockapi.io/cart/' + obj.id);
42       } else {
43         setCartItems(prev => [...prev, obj]);
44         await axios.post('https://6185878923a2fe0017fff680.mockapi.io/cart', obj);
45       }
46     } catch (error) {
47     }
48   }
49 }
```

```
src > JS App.js > @ onAddToCart
59 } catch (error) {
60   alert('Error deleting to cart');
61   console.error(error);
62 }
63
64 const onAddToFavorites = async (obj) => {
65   try {
66     if (favorites.find(favObj => Number(favObj.id) === Number(obj.id)) {
67       axios.delete('https://6185878923a2fe0017fff680.mockapi.io/favorites/' + obj.id);
68     } else {
69       const { data } = await axios.post('https://6185878923a2fe0017fff680.mockapi.io/favorites', obj);
70       setFavorites(prev => [...prev, data]);
71     }
72   } catch (error) {
73     alert('Failed to add to favorites');
74     console.error(error);
75   }
76 }
77
78 const onChangeSearchInput = (event) => {
79   setSearchValue(event.target.value);
80 }
81
82 const isItemAdded = (id) => {
83   return cartItems.some(obj => Number(obj.id) === Number(id));
84 }
85
86
87 return (
88   <AppContext.Provider value={{ items, cartItems, favorites, isItemAdded, onAddToFavorites, setCartOpened, setCartItems }}>
89     <div className="wrapper clear">
90       <Drawer items={cartItems} onClose={() => setCartOpened(false)} onRemove={onRemoveItem}/>
91       <Header onClickCart={() => setCartOpened(true)} />
92
93       <Routes>
94         <Route path="/" element={Home items={items}} />
95         <Route path="/cart" element={Cart items={cartItems} />
96         <Route path="/favorites" element={Favorites searchValue={searchValue} setSearchValue={setSearchValue} onChangeSearchInput={onChangeSearchInput} onAddToFavorites={onAddToFavorites} onAddToCart={onAddToCart} isLoading={isLoading} />
97       </Routes>
98     </div>
99   </AppContext.Provider>
100 );
101
102 </>
103
104 </>
105
106 </>
107 }
```

# Context.js

```
src > JS context.js > @ default
1 import React from "react";
2
3
4 const AppContext = React.createContext({});
5
6 export default AppContext;
```

## Index.js

```
REACT-STORE src > JS index.js
> build 1 import React from 'react';
> node_modules 2 import { BrowserRouter as Router } from 'react-router-dom';
> public 3 import ReactDOM from 'react-dom';
src 4 import './index.scss';
  components 5 import App from './App';
  Card 6 import 'macro-css';
  Card.module.scss 7
  index.js 8
  Drawer.js 9 ReactDOM.render(
  Header.js 10   <React.StrictMode>
  Info.jsx 11     <Router>
  pages 12       <App />
  Favorites.jsx 13     </Router>
  Home.jsx 14   </React.StrictMode>,
  App.js 15   document.getElementById('root')
  context.js 16 );
  index.js 17
  index.scss 18 // If you want to start measuring performance in your app, pass a function
19 // to log results (for example: reportWebVitals(console.log))
20 // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
21
```

## Index.scss

```
REACT-STORE src > index.scss > .search-block > .clear
1 body {
2   margin: 0;
3   -webkit-font-smoothing: antialiased;
4   -moz-osx-font-smoothing: grayscale;
5   background-color: #e766ff;
6 }
7
8 + {
9   font-family: 'Inter', system-ui;
10 }
11
12 .wrapper {
13   background: #ffffff;
14   box-shadow: 0px 10px 20px rgba(0, 0, 0, 0.04);
15   border-radius: 20px;
16   max-width: 1080px;
17   margin: 50px auto;
18 }
19
20 header {
21   border-bottom: 1px solid #eaeaea;
22   img {
23     margin-right: 15px;
24   }
25
26   h3,
27   p {
28     margin: 0;
29   }
30 }
31
32 .content {
33   h1 {
34     margin: 0;
35   }
36 }
37
38 input {
39   border: 0;
40   padding: 13px;
41   font-size: 16px;
42   width: 200px;
43 }
44
45 .overlay {
46   position: absolute;
47   left: 0;
48   top: 0;
49   width: 100%;
50 }
```



```

REACT-STORE src > index.scss > search-block > .clear
> build 51 background-color: rgba(0, 0, 0, 0.5);
> node_modules 52 z-index: 1;
> public 53 }
> src 54 .drawer{
components 55 display: flex;
> Card 56 flex-direction: column;
Card.module.scss 57 position: absolute;
index.js 58 width: 420px;
Drawer.js 59 height: 100%;
Header.js 60 right: 0;
Info.js 61 background: #ffffff;
pages 62 box-shadow: -10px 4px 24px rgba(0,0,0,0.1);
Favorites.jsx 63 padding: 30px;
Home.jsx 64 .items{
App.js 65 flex: 1;
context.js 66 overflow: auto;
index.js 67 margin-bottom: 40px;
index.scss 68 }
.gitignore 69
context.js 70
package-lock.json 71
package.json 72
README.md 73
74
75
76 .cartTotalBlock{
77 ul {
78 display: block;
79 margin-bottom: 40px !important;
80
81 li {
82 display: flex;
83 align-items: flex-end;
84 margin-bottom: 20px;
85 margin-top: 50px;
86
87 div {
88 flex: 1;
89 height: 1px;
90 border-bottom: 1px dashed #dfdfdf;
91 position: relative;
92 top: -4px;
93 margin: 0 7px;
94 }
95 }
96 }
97
98
99 .cartItem {

```

```

REACT-STORE src > index.scss > @keyframes button-loading
node_modules 104
public 105
src 106
components 107
Card 108
Card.module.scss 109
index.js 110
Drawer.js 111
Header.js 112
Info.js 113
pages 114
Favorites.jsx 115
Home.jsx 116
App.js 117
context.js 118
index.js 119
index.scss 120
.gitignore 121
context.js 122
package-lock.json 123
package.json 124
README.md 125
126
127
128
129
130
131
132
133 .cartItemImg {
134 height: 70px;
135 width: 70px;
136 background-size: contain;
137 background-position: 0 -1px;
138 background-repeat: no-repeat;
139 margin-right: 20px;
140 }
141
142 p {
143 font-size: 16px;
144 margin: 0;
145 }
146
147 b {
148 font-size: 14px;
149 }
150
151 .removeBtn {
152 opacity: 0.5;
153 cursor: pointer;
154 transition: opacity 0.15s ease-in-out;
155
156 &:hover {
157 opacity: 1;
158 }
159 }
160
161 .cartEmpty {
162 text-align: center;
163
164 p {
165 width: 280px;
166 line-height: 24px;
167 }
168
169 .orangeButton {
170 width: 245px;
171 margin-top: 20px;
172
173 &:hover {
174 img {
175 transform: rotate(180deg) translateX(3px);
176 }
177 }
178
179 img {
180 position: relative;

```

```
REACT-STORE
├── build
├── node_modules
├── public
├── src
│   ├── components
│   │   ├── Card
│   │   └── Card.module.scss
│   ├── index.js
│   ├── Drawer.js
│   ├── Header.js
│   ├── Info.jsx
│   └── pages
│       ├── Favorites.jsx
│       ├── Home.jsx
│       ├── App.js
│       ├── context.js
│       ├── index.js
│       └── index.scss
├── .gitignore
├── context.js
├── package-lock.json
├── package.json
└── README.md

src > index.scss > @keyframes button-loading
164 padding: 0 15px;
165 position: relative;
166
167
168 .clear {
169     position: absolute;
170     right: 0;
171     width: 18px;
172     height: 18px;
173     top: 14px;
174     right: 15px;
175 }
176
177
178 .orangeButton {
179     width: 100%;
180     height: 55px;
181     background: #ffa500;
182     border-radius: 18px;
183     border: 0;
184     color: #fff;
185     font-size: 16px;
186     font-weight: 500;
187     cursor: pointer;
188     transition: background 0.1s ease-in-out;
189
190     &:disabled {
191         background-color: #bebebe !important;
192         cursor: default;
193     }
194
195     &:hover {
196         background: lighten(#ffa500, 10%);
197     }
198
199     &:active {
200         background: darken(#ffa500, 5%);
201     }
202 }
203
204 @keyframes button-loading {
205     0% {
206         opacity: 1;
207     }
208     50% {
209         opacity: 0.7;
210     }
211     100% {
212         opacity: 1;
213     }
214 }
```