Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

(повна назва)

Кафедра Інформатика

(повна назва)

**ПОЯСНЮВАЛЬНА ЗАПИСКА**

до курсової роботи

з дисципліни “ Організація баз даних та знань ”

за темою

 “Розробка веб-сайту „ Інтернет-магазин взуття ” з використанням патернів програмування мови JavaScript та мікросервісу”

Виконав:

студент 3 курсу, групи ІТІНФ-20 -1

Самченко Станіслав Олександрович

(прізвище, ініціали)

Спеціальності 122 Комп’ютерні науки

(код і повна назва спеціальності)

Освітня програма Інформатика

(повна назва освітньої програми)

Керівник ст. викладач Кіношенко Д. К.

(посада, прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

(найменування вищого учбового закладу)

Факультет Інформаційно-аналітичних технологій та менеджменту

(повна назва)

Кафедра Інформатики

(повна назва)

Спеціальність 122 Комп’ютерні науки

(код і повна назва)

Освітня програма Інформатика

(повна назва освітньої програми)

Дисципліна Теорія програмування

(повна назва освітньої програми)

Курс 3 Группа ІТІНФ-20 -1 Семестр 6 .

ЗАВДАННЯ

на курсовий проект (роботу) студента

Самченко Станіслав Олександрович

(прізвище, ім’я, по батькові)

1. Тема проекту Розробка веб-сайту “Інтернет-магазин кросівок” з використанням мови програмування JavaScript, теорії програмування, патернів програмування, мікросервісу, мови запитів SQL, СУБД MySQL, мови декорування CSS(SCSS), мови гіпертекстової розмітки HTML, середовища розробки програмних систем Visual Studio Code та середовища виконання JavaScript Reat.   
2. Строк здачі студентом закінченого проекту    11.06.2023                              .

3. Вихідні данні проекту мова програмування JavaScript, мова запитів SQL, СУБД  MySQL, мова декорування CSS(SCSS), мова гіпертекстової розмітки HTML, середовище розробки програмних систем Visual Studio Code, середовище виконання JavaScript React

4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)

1. Особливості чистого коду та рефакторингу.

2. Основні властивості патернів програмування.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Поняття мікросервісу.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Побудова структури веб-сайту.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5.Відображення взаємодії програмних компонентів.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Огляд та вибір засобів програмної реалізації.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

7.Програмне створення веб-сайту «Інтернет-магазин кросівок».

6. Дата видачі завдання     13.02.23 .

КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| № | Найменування етапів курсового проекту | Термін виконання етапів проекту | Примітки |
| 1 | Отримання завдання | 13.02.2023 | Виконано |
| 2 | Вивчення теоретичного матеріалу, ознайомлення з програмним забезпеченням | 02.05.23 – 16.05.23 | Виконано |
| 3 | Розробка структури веб-сайту | 18.05.23 – 25.05.23 | Виконано |
| 4 | Розробка функціональності веб-сайту | 25.05.23 – 01.06.23 | Виконано |
| 5 | Тестування веб-сайту | 02.06.23 – 03.06.23 | Виконано |
| 6 | Оформлення пояснювальної записки | 03.06.23 – 04.06.23 | Виконано |
| 7 | Захист курсового проекту | 11.06.23 | Виконано |

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Керівник   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

                               (підпис)       (підпис) **РЕФЕРАТ/ABSTRACT**

Пояснювальна записка: 35 с., 16 рис., 1 дод., 3 джерел.

ТЕОРІЯ ПРОГРАМУВАННЯ, ПАТЕРНИ ПРОГРАМУВАННЯ, МІКРОСЕРВІС, МОВА ПРОГРАМУВАННЯ JAVASCRIPT, МОВА ЗАПИТІВ SQL, СУБД MYSQL, МОВА ДЕКОРУВАННЯ CSS, МОВА ГІПЕРТЕКСТОВОЇ РОЗМІТКИ HTML, СЕРЕДОВИЩЕ РОЗРОБКИ ПРОГРАМНИХ СИСТЕМ VISUAL STUDIO CODE, СЕРЕДОВИЩЕ ВИКОНАННЯ JAVASCRIPT REACT.

Проект присвячений розробці веб-сайту з використанням патернів програмування та мікросервісу.

Веб-сайт призначений для реалізації здійснення покупки та зручного вибору із великого асортименту взуття за допомогою сучасних технологій.

Під час розробки веб-сайту були використані: теорія програмування, патерни програмування, мікросервіс, мова програмування JavaScript, мова запитів SQL, СУБД MySQL, мова декорування CSS(SCSS, мова гіпертекстової розмітки HTML, середовище розробки програмних систем Visual Studio Code, середовище виконання JavaScript React.

PROGRAMMING THEORY, PROGRAMMING PATTERNS, MICROSERVICE, JAVASCRIPT PROGRAMMING LANGUAGE, SQL QUERY LANGUAGE, MYSQL DATABASE, CSS(SCSS) DECORATION LANGUAGE, HTML HYPERTEXT MARKUP LANGUAGE, VISUAL STUDIO CODE SOFTWARE DEVELOPMENT ENVIRONMENT, REACT JAVASCRIPT RUNNING ENVIRONMENT.

The project is dedicated to website development using programming patterns and microservices.

The website is designed to generate names, surnames, fictional countries, that is, it can serve as a kind of assistant for writers, etc. Users are granted different levels of access: unregistered website visitors can generate data with a smaller set of capabilities, while registered users are granted more functions. The administrator, in turn, can review all possible generation options.

During the development of the website, the following were used: programming theory, programming patterns, microservices, JavaScript programming language, SQL query language, MySQL DBMS, CSS(SCSS) decoration language, HTML hypertext markup language, Visual Studio Code software development environment, the React JavaScript runtime.

**ЗМІСТ**

Перелік умовних позначень, символів, одиниць, скорочень і термінів 7

Вступ 8

1 Поняття чистого коду, патернів програмування та мікросервісів 10

1.1 Особливості чистого коду та рефакторинг ................................... 10

1.2 Основні властивості патернів програмування .............................. 11

1.3 Поняття мікросервісу ...................................................................... 12

1.4 Постановка задачі курсового проекту ........................................... 13

1.4.1 Мета курсового проекту ....................................................... 13

1.4.2 Опис суті проекту ................................................................. 14

1.4.3 Задачі, необхідні для вирішення .......................................... 14

1.4.4 Необхідні засоби для реалізаціі ........................................... 15

2 Проектування веб-сайту 16

2.1 Структура проекту .......................................................................... 16

2.2 Відображення взаємодії програмних компонентів ...................... 18

2.3 Огляд використання патерну ......................................................... 19

2.4 Огляд використання мікросервісу ................................................. 19

3 Розробка веб-сайту 21

3.1 Обґрунтування вибору середовища програмної реалізації ......... 21

3.2 Програмна реалізація патерну програмування ............................ 21

3.3 Програмна реалізація мікросервісу ............................................... 23

3.4 Ілюстрація роботи ........................................................................... 24

Висновки 27

Перелік джерел та посилання 29

Додаток А Інстукція користувачу 30

# Перелік умовних позначень, символів, одиниць, скорочень і термінів

JS – мова програмування JavaScript

CSS – Cascading Style Sheets

HTML – HyperText Markup Language

СУБД – Система управління базою даних

SQL - Structured query language

БД – База даних

# **ВСТУП**

Розробка ефективних програм є важливим аспектом діяльності програмістів. Важливим критерієм ефективного коду є його читабельність, що спрощує взаємодію з ним і розуміння розробниками. Таким чином, запорукою успішної програми є так званий «чистий або хороший» код – код, який легко і приємно читається і зрозумілий розробникам з першого погляду. Для коригування програм використовується рефакторинг.

Рефакторингом називається процес зміни коду з метою реалізації його структури та читабельності, не змінюючи його зовнішньої поведінки. У мові JavaScript рефакторинг відіграє важливу роль у розробці високого коду. Рефакторинг дозволяє збільшити читаність, зменшити дублювання коду та зробити його більш модульним, що розширить можливості його підтримки та розвитку.

Крім читабельності коду, є інші чинники, що впливають на його ефективність. Одним не менш важливим критерієм є використання такого підходу, як шаблони проектування – патерни.

Простими словами, патерн – усталений спосіб вирішення типового завдання. У широкому значенні патерн означає загальноприйняту практику чи підхід, договір між розробниками. Програмісти пишуть багато кодів, але ще більше коду вони читають. Основне призначення патернів у тому, що вони допомагають і новачкам, і досвідченим програмістам економити час розробки, мінімізувати кількість помилок і, загалом, виглядати грамотнішим під час розмови з колегами. Вони використовують моделі або архітектурні підходи, які допомагають прискорити розробку та підвищити її якість. JavaScript має високий спектр патернів. Ці шаблони дозволяють розробникам створювати гнучкий, розширюваний і реалізований код.

В теорії програмування існує підхід, який допомагає не тільки прискорити розробку продукту, але й зробити її гнучкою і керованою: проект із неподільного цілого перетворюється на систему пов'язаних між собою блоків – сервісів. Цей підхід називається мікросервісною архітектурою.

Мікросервісна архітектура (або просто «мікросервіси») – це підхід до створення програми у вигляді набору сервісів, що незалежно розгортаються, які є децентралізованими і розробляються незалежно один від одного. Ці послуги слабко пов'язані, незалежно розгортаються і легко обслуговуються. Кожен мікросервіс виконує окремі функції та взаємодіє з іншими мікросервісами через легковажні протоколи. У мові програмування мікросервісів JavaScript можна використовувати різні інструменти, такі як Node.js та Express.js. Вони мають гнучкість, масштабованість і можливість незалежної розробки і розгортання компонентів.

Використання шаблонів програмування, рефакторингу та мікросервісів на мові JavaScript дозволяє розробникам створювати вражаючі та ефективні програми.

Таким чином, цей проект присвячений розробці веб-сайту з використанням патернів програмування та мікросервісу.

## Поняття Чистого коду, патернів програмування та мікросервісів

* 1. Особливості чистого коду та рефакторинг

### Чистий код та рефакторинг є ключовими практиками у розробці, спрямованої на створення особливого коду. Вони відіграють важливу роль у покращенні читаності, модульності та ефективності розробки.

### Чистий код – це код, який легко читається, зрозумілий і доступний для інших. Він добре структурований та легко сприймається. Чистий код також слідує практиці його кодування, яка охоплює послідовні дії. Чистий код зазвичай містить добре названі змінні, функції та класи, коментарі, що описують його призначення, та логічну структуру, яка допомагає його зрозуміти.

### Рефакторинг – це процес зміни коду з метою досягнення його ефективної структури та забезпечення вищої якості без зміни зовнішньої поведінки. Він дозволяє виключити дублювання коду, підвищити читання, ускладнити конструкцію та підвищити продуктивність. Рефакторинг також має більш мінливий код, який легко дотримується та модифікується. Він здійснюється за допомогою набору технічних засобів та інструментів, які дозволяють значно змінити код без ризику порушення його функціональності.

Чистий код та рефакторинг мають велике значення:

1. Розуміння та підтримка: розуміння коду та підтримка розробниками, які працюють над проектом. Він робить код зрозумілим та передбачуваним, що допомагає розробникам швидше знаходити та виправляти помилки.
2. Модульність та повторне використання: чистий код, поділений на невеликі модулі, дозволяє повторно використовувати код та отримати його розширення.
3. Зменшення технічного боргу: рефакторинг дозволяє скоротити технічний борг – неоптимальні накопичення та проблеми у коді. Це допомагає знизити ймовірність виникнення помилок та покращити загальну архітектуру додатків.
4. Поліпшення співробітництва: чистий код та рефакторинг полегшують співпрацю між розробниками. Коли код є чистим і зрозумілим, команда може легко працювати над проектом, відповідальністю суспільства і проводити код-рев'ю.

### 

* 1. Основні властивості патернів програмування

Патерни програмування – це способи побудови програм, які вважаються гарним тоном для розробників. Вони є формалізованими описами архітектурних конструкцій, які відносяться до розробки програмного забезпечення та структурованого коду.

Патерни розвивають самі схеми або шаблони, які взаємодіють між класами та об'єктами, а також організують їхню взаємодію. Вони є багаторічним досвідом, виробленим підходом до вирішення поширених проблем.

Основна мета патернів програмування полягає в тому, щоб мати гнучкість, повторне використання та зрозумілість коду. Вони допомагають розробникам створювати програмне забезпечення, яке легко підтримувати, модифікувати та масштабувати. Крім того, розвивають підвищення продуктивності, зменшення кількості помилок та підвищують рівень розвитку.

Патерни програмування можна розділити на кілька категорій залежно від своєї функціональності. Деякі з найпоширеніших категорій патернів програмування включають:

1. Шаблони, що породжують (Creational): створення нових об'єктів: Конструктор (Constructor), Модульний (Module), Фабрика (Factory), Сінглтон(Singletion).
2. Структурні шаблони (Structural): впорядковують об'єкти: Декоратор(Decorator), Фасад (Façade).
3. Поведінкові (Behavioral): як об'єкти взаємодіють один з одним: Спостерігач(Observer), Посередник (Mediator), Команда(Command).

Важливо, що використання шаблонів програмування є обов'язковим, але допомагають собі цінний інструмент.

* 1. Поняття мікросервісу

Мікросервісна архітектура, також звана мікросервісами, є методологією розробки програмного забезпечення, де додаток розбивається на набір окремих сервісів, які можуть бути реалізовані незалежно один від одного. Ці послуги легко взаємодіють один з одним за допомогою легких протоколів. На мові розробки JS під час створення мікросервісів можна використовувати інструменти, включаючи Node.js та Express.js. Node.js надає виконання JavaScript на сервері, а Express.js є легковажним фреймворком, що полегшує розробку та розгортання мікросервісів. Ці інструменти вимагають гнучкості, масштабованості та незалежної можливості розробки та розгортання компонентів.

Мікросервісна архітектура, що пропонує ряд переваг, включає:

1. Гнучкість та масштабованість: завдяки поділу додатків на окремі мікросервіси збирачі отримують більшу гнучкість при внесенні змін або додаванні нової функціональності. Кожен мікросервіс може бути розроблений та протестований незалежно, що дозволяє легко масштабувати певні компоненти відповідно до вказівки.
2. Легка підтримка та обслуговування: кожен мікросервіс існує незалежно від інших, його обслуговування та підтримка стають більш і передбачуваними. Зміни, внесені до одного мікросервіс, не мають впливу на інші компоненти системи.
3. Мікросервісна архітектура дозволяє збільшити навантаження між мікросервісами. Це означає, що при призначенні навантаження на систему можна масштабувати лише необхідні компоненти, не торкаючись інших сервісів. Це більш ефективне використання ресурсів та підвищення продуктивності системи.
4. Легка взаємодія: мікросервіси зазвичай взаємодіють один з одним через легкі протоколи, такі як HTTP або повідомлення на основі шаблонів.
   1. Постановка задачі курсового проекту
      1. Мета курсового проекту

Метою курсового проекту є розробка веб-сайту з використанням принципів чистого коду, патернів програмування та мікросервісу.

* + 1. Опис суті проекту

Проект "Інтернет-магазин кросівок" є віртуальною платформою, призначеною для продажу кросівок через інтернет. Головна мета проекту - надати покупцям широкий вибір кросівок різних брендів, моделей та стилів, та забезпечити зручність та комфорт при покупці.

Суть проекту полягає у створенні та підтримці онлайн-магазину, де покупці можуть легко знайти та придбати кросівки, що відповідають їх перевагам та вимогам. Користувачі мають можливість переглядати каталог кросівок з описом та зображеннями, ознайомитися з характеристиками товарів, цінами та наявністю.

* + 1. Задачі, необхідні для вирішення

Для досягнення мети курсового проекту необхідно вирішити такі завдання:

- вивчити такі теоретичні питання: теорію програмування; основні принципи рефакторингу коду; патерни програмування JavaScript; принципи створення мікросервісів з використанням Node.js;

- ознайомитись з наступними інструментами проектування та програмування: середовище виконання JavaScript React, MS Visual Studio Code;

- застосувати отримані теоретичні та практичні знання для розробки веб-сайту;

- сформувати структуру веб-сайту;

- розробити дизайн веб-сайту;

- розробити програмний код для забезпечення необхідної функціональності веб-сайту з використанням патернів програмування та мікросервісу;

- провести тестування проекту;

- відобразити процес та результати розробки веб-сайту у пояснювальній записці до курсового проекту.

* + 1. Необхідні засоби для реалізації

Для розробки веб-сайту необхідно використовувати такі інструменти:

Visual Studio Code, JavaScript – середовище та мова розробки програмного забезпечення;

React – фреймворк мови JavaScript;

SCSS – препроцесор мови декорування CSS;

MySQL – СУБД;

HTML – мова гіпертекстової розмітки.

1. **ПРОЕКТУВАННЯ ВЕБ-САЙТУ**
   1. Структура проекту
      1. картка товару

Файл Index.jsx – розмітка картки товару

Файл Card.module.scss – стилі карти товару

* + 1. кошик

Файл index.jsx – розмітка кошику

Файл Drawer.module.scss – стилі кошику

Файл info.jsx – додаткова інформація для кошику

* + 1. шапка

Файл index.jsx – розмітка шапки

Файл Header.module.scss – стилі шапки

* + 1. хуки

useCart.js – логіка хуку кошику

useSearch.js – логіка для реалізації пошуку на сайті

* + 1. сторінки

Favorites.jsx – розмітка сторінки обране

Home.jsx – розмітка головної сторінки

Order.jsx – розмітка сторінки замовлень

2.1.6 база даних

таблиця Items – відповідає за зберігання усіх необхідних даних про кросівок.

таблиця Cart – відповідає за зберігання товарів доданих до кошику.

таблиця Favorites – відповідає за зберігання товарів доданих до обраних товарів.

таблиця Orders – відповідає за зберігання товарів доданих до замовлення.

2.2 Відображення взаємодії програмних компонентів

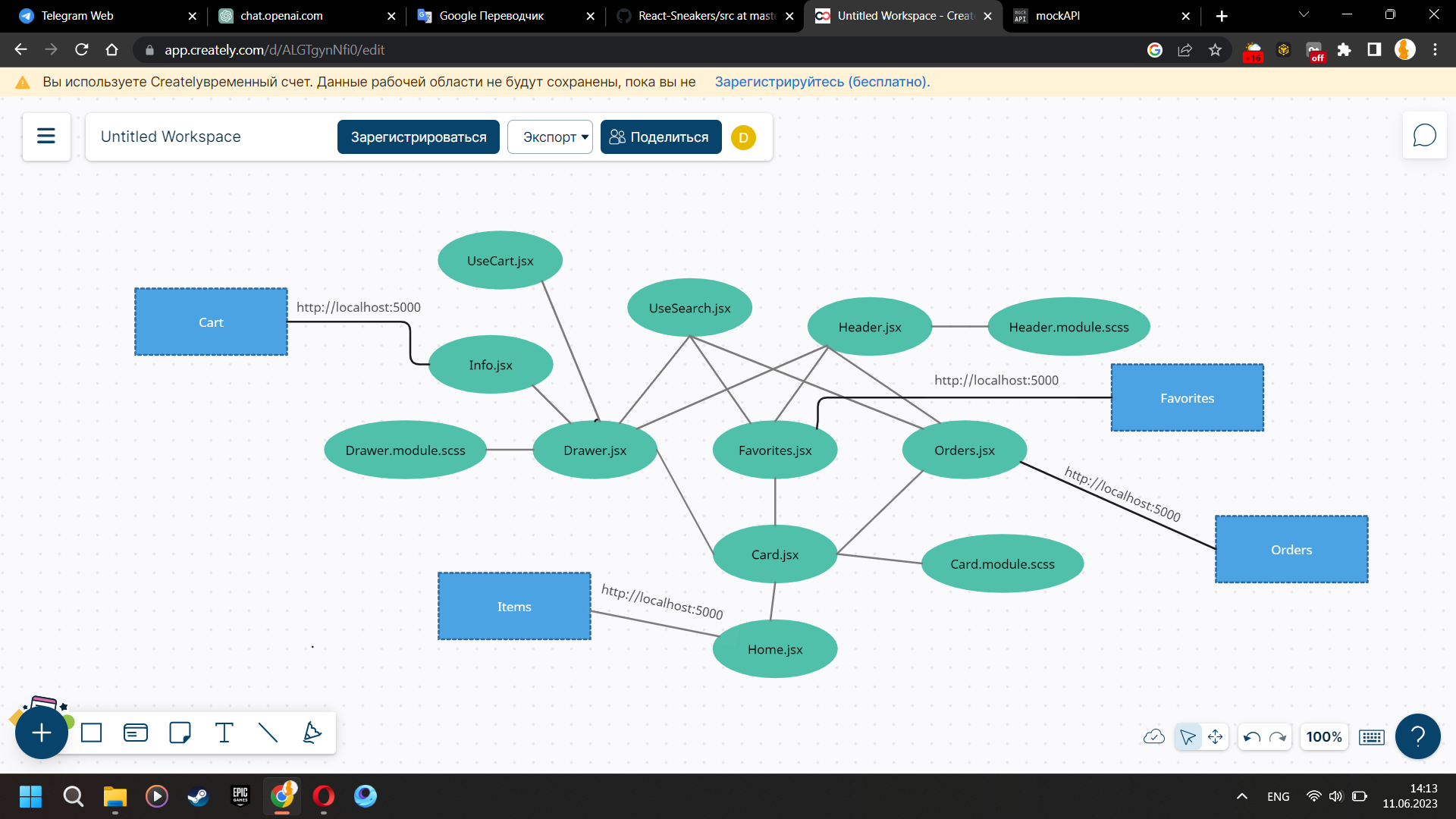


Рисунок 2.1 – Діаграма взаємодії компонентів програми

2.3 Огляд використання патерну

У даному проекті використовується ряд декількох патернів:

* Компоненти: У проекті використовується підхід компонентної архітектури React, де кожна частина інтерфейсу представлена окремим компонентом. Наприклад, компоненти "Header", "Main", "Footer" та інші.
* Маршрутизація: Використовується бібліотека React Router для керування маршрутизацією та відображення різних компонентів на основі URL. У файлі App.js налаштовані маршрути для різних сторінок.
* Використання хуків: Проект використовує хуки React, такі як useState і useEffect, для керування станом компонентів та виконання побічних ефектів.
* Стилізація: У структурі проекту присутні CSS-файли, такі як "index.css" та "App.css", які використовуються для стилізації компонентів.
* Використання контексту: Файл "context.js" показує використання контексту React для передачі даних між компонентами.
* Компоненти вищого ладу (Higher-Order Components, HOC): У файлі "hoc" знаходяться компоненти вищого ладу, які обертають інші компоненти та надають їм додаткові функціональні можливості. Наприклад, HOC "withModal" додається функціональність модального вікна до компонентів.
* Робота з формами: У папці "components" є компоненти, що відповідають за роботу з формами, наприклад, "CartForm" та "OrderForm". Ці компоненти демонструють патерн контрольованої форми, де значення полів форми зберігаються у стані компонента
* Робота з Redux: У папці Redux містяться файли, пов'язані з керуванням станом за допомогою Redux. Тут використовуються патерни, такі як "actions", "reducers" і "selectors", щоб організувати зберігання та оновлення даних у додатку.
* Використання CSS-препроцесора: У файлі "App.css" видно використання SCSS (Sass) для зручнішої та потужнішої роботи зі стилями.
* Використання бібліотеки Axios: У файлі "api.js" застосовується бібліотека Axios для виконання запитів HTTP до сервера.
* Використання React Hooks: Компоненти у багатьох файлах використовують хуки, такі як useState, useEffect та useContext, для керування станом та виконання побічних ефектів.
* Компоненти-контейнери: У папці "containers" знаходяться компоненти, які обертають інші компоненти та надають їм дані та функції через Redux або контекст.

2.4 Огляд використання мікросервісу

Коли користувач переходить за посиланням/відкриває програму в браузері, завантажується home.jsx, відображає інтерфейс програми.

Користувач обирає товари та обирає куди він хоче їх додати. Якщо він додає їх до кошику, він натикає кнопку «оформити замовлення» і на сервер, через HTTP-запитів, відправляється замовлення у JSON форматі з переліком обраних товарів.

Серверна частина програми отримує та обробляє замовлення у JSON форматі, відправивши повідомлення на пошту адміністрації та в базу даних замовлень про суму та розмір товару.

Клієнт бачить повідомлення про успішну сплату та прийняття його замовлення.

Таким чином, програма взаємодіє між клієнтською та серверною частинами, використовуючи HTTP-запити та JSON-відповіді для передачі даних. Клієнтська частина оновлює інтерфейс відповідно до отриманих даних.

**3 РОЗРОБКА ВЕБ-САЙТУ**

* 1. Обґрунтування вибору середовища програмної реалізації

У рамках даного курсового проекту був розроблений веб-сайт. Для реалізації було обране середовище Visual Studio Code. Це обумовлено тим, що Visual Studio Code – це програмне середовище з розробки програм на використаних мовах. Також Visual Studio Code має ряд переваг, дозволяє легко писати, форматувати та редагувати код різними мовами. З його допомогою можна швидко створити проект та структуру файлів у ньому, він підсвічує синтаксис коду та допомагає автоматично правити помилки. У ньому є можливості для налагодження та запуску коду деякими мовами.

Основною мовою написання було обрано – JavaScript – мову програмування, яку використовують для написання frontend- та backend-частин сайтів. Це мова програмування високого рівня, тобто код нею зрозумілий добре читається і найкраще підходить для створення веб-сайтів.

Для забезпечення програми стилями була використана мова декорування CSS.

* 1. Програмна реалізація патерну програмування

Використання компонентів таких як header і drawer винесених в окремі файли. Використання бібліотеки React Router для керування маршрутизацією та відображення різних компонентів. Використання контекту:

...

    <AppContext.Provider value={{items, cartItems, onAddToCart, isItemAddedToCart, cartOpened, setCartOpened, setCartItems, onRemoveFromCart, favorites, setFavorites, onAddToFavorites, isItemAddedToFavorited, isLoading}}>

      <div className="wrapper clear">

      <Drawer/>

        <Header/>

        <Routes>

          <Route path={process.env.PUBLIC\_URL + "/"} exact element={<Home/>}/>

          <Route path={process.env.PUBLIC\_URL + "/favorites"} element={<Favorites/>}/>

          <Route path={process.env.PUBLIC\_URL + "/orders"} element={<Orders/>}/>

        </Routes>

      </div>

    </AppContext.Provider>

  );

...

3.3 Програмна реалізація мікросервісу

...

   const [isLoading, setIsLoading] = React.useState(true);

  const [favorites, setFavorites] = React.useState([]);

  const [items, setItems] = React.useState([]);

  const [cartItems, setCartItems] = React.useState([]);

  const [cartOpened, setCartOpened] = React.useState(false);

  React.useEffect(() => {

    (async ()=> {

      try {

        const [cartsRes, itemsRes, favoritesRes] = await Promise.all([

          axios.get('https://637691c4b5f0e1eb850f3de5.mockapi.io/cart'),

          axios.get('https://637691c4b5f0e1eb850f3de5.mockapi.io/items'),

          axios.get('https://637691c4b5f0e1eb850f3de5.mockapi.io/favorites')

        ])

        setIsLoading(false);

        setFavorites(favoritesRes.data);

        setCartItems(cartsRes.data);

        setItems(itemsRes.data);

      } catch (error) {

        alert('Ошибка загрузки данных');

        console.error(error);

      }

    })();

  }, []);

  const onAddToCart = async (obj) => {

    try {

      const findItems = cartItems.find(item =>  Number(item.parentId) === Number(obj.id))

      if (findItems) {

        setCartItems(prev => prev.filter(item => Number(item.parentId) !== Number(obj.id)));

        await axios.delete(`https://637691c4b5f0e1eb850f3de5.mockapi.io/cart/${findItems.id}`);

      } else {

        setCartItems(prev => [...prev, obj]);

        const {data} = await axios.post('https://637691c4b5f0e1eb850f3de5.mockapi.io/cart', obj);

        setCartItems(prev => prev.map(item => {

          if (item.parentId === data.parentId) {

            return {...item, id: data.id}

          };

          return item;

        }));

      }

    } catch (error) {

      console.error(error);

      alert('Не удалось добавить товар в корзину..');

    }

  };

  const onRemoveFromCart = (id) => {

    try {

      axios.delete(`https://637691c4b5f0e1eb850f3de5.mockapi.io/cart/${id}`);

      setCartItems(prev => prev.filter(item => Number(item.id) !== Number(id)));

    } catch (error) {

      console.error(error);

      alert('Не удалось удалить товар из корзины');

    }

  };

  const isItemAddedToCart = (id) => {

    return cartItems.some(obj => Number(obj.parentId) === Number(id));

  };

  const isItemAddedToFavorited = (id) => {

    return favorites.some(obj => Number(obj.parentId) === Number(id));

  };

  const onAddToFavorites = async (obj) => {

    try {

      const favoritedItem = favorites.find(fevoriteObj => Number(fevoriteObj.parentId) === Number(obj.parentId))

      if (favoritedItem) {

        axios.delete(`https://637691c4b5f0e1eb850f3de5.mockapi.io/favorites/${obj.id}`);

        setFavorites(prev => prev.filter(item => Number(item.id) !== Number(obj.id)));

      } else {

      const {data} = await axios.post('https://637691c4b5f0e1eb850f3de5.mockapi.io/favorites', obj);

      setFavorites(prev => [...prev, data]);

      }

    } catch (error) {

      alert('Не удалось добавить в избранное');

    }

  };

...

3.4 Ілюстрація роботи

Функціонал веб-сайту представлений нижче скріншотами роботи програми.

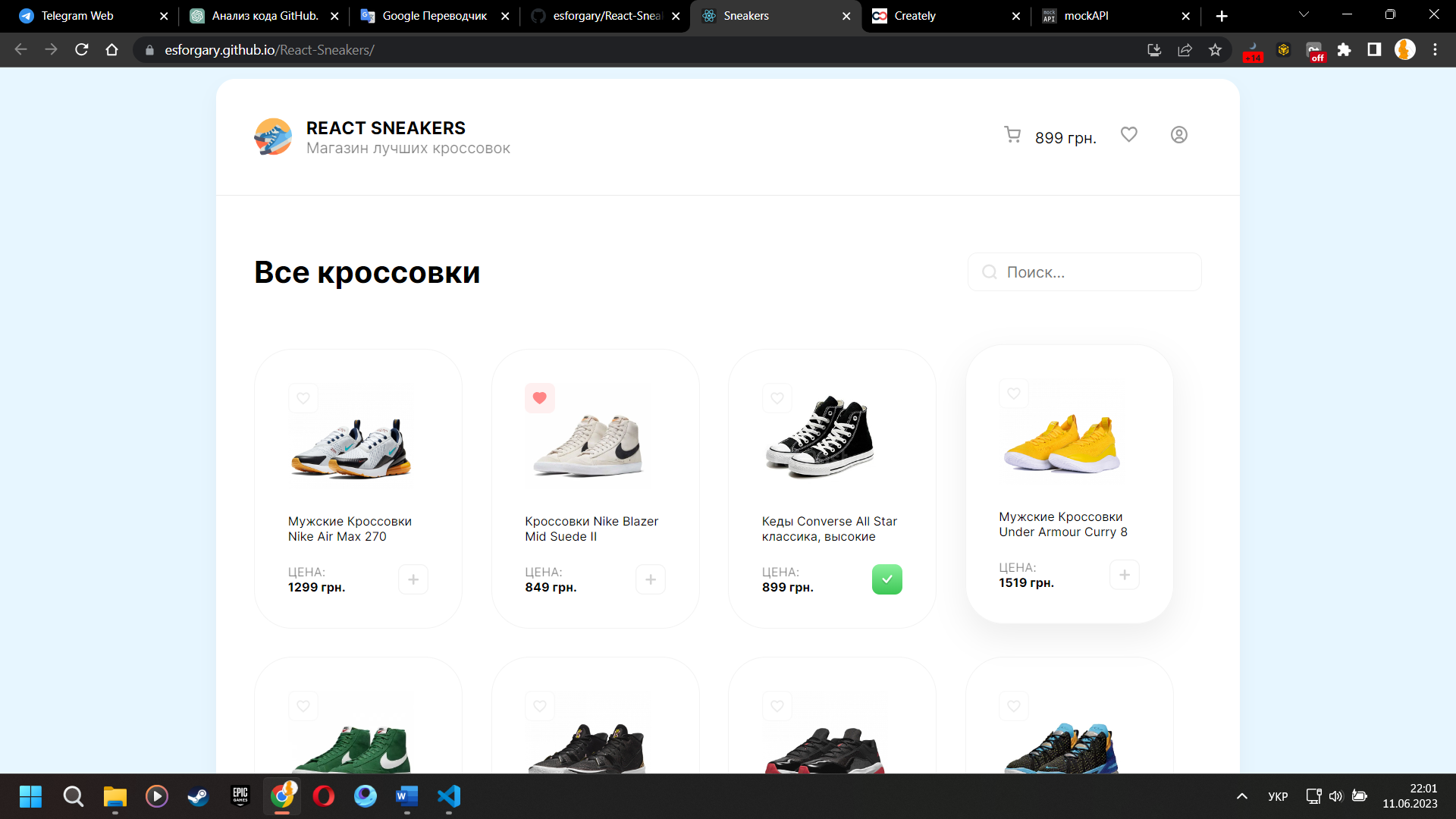


Рисунок 3.1 – Головна сторінка

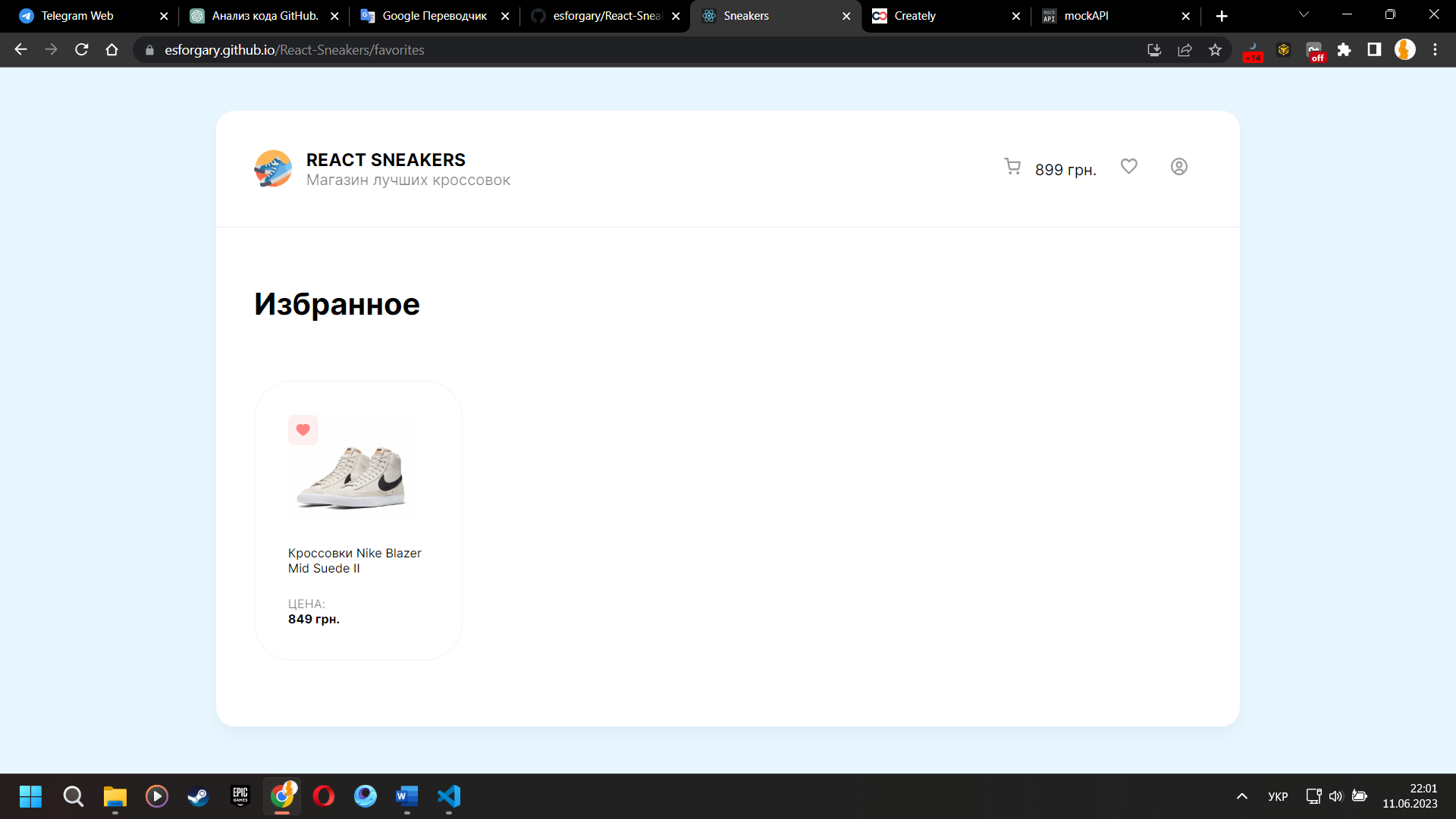


Рисунок 3.2 – Сторінка обраних товарів

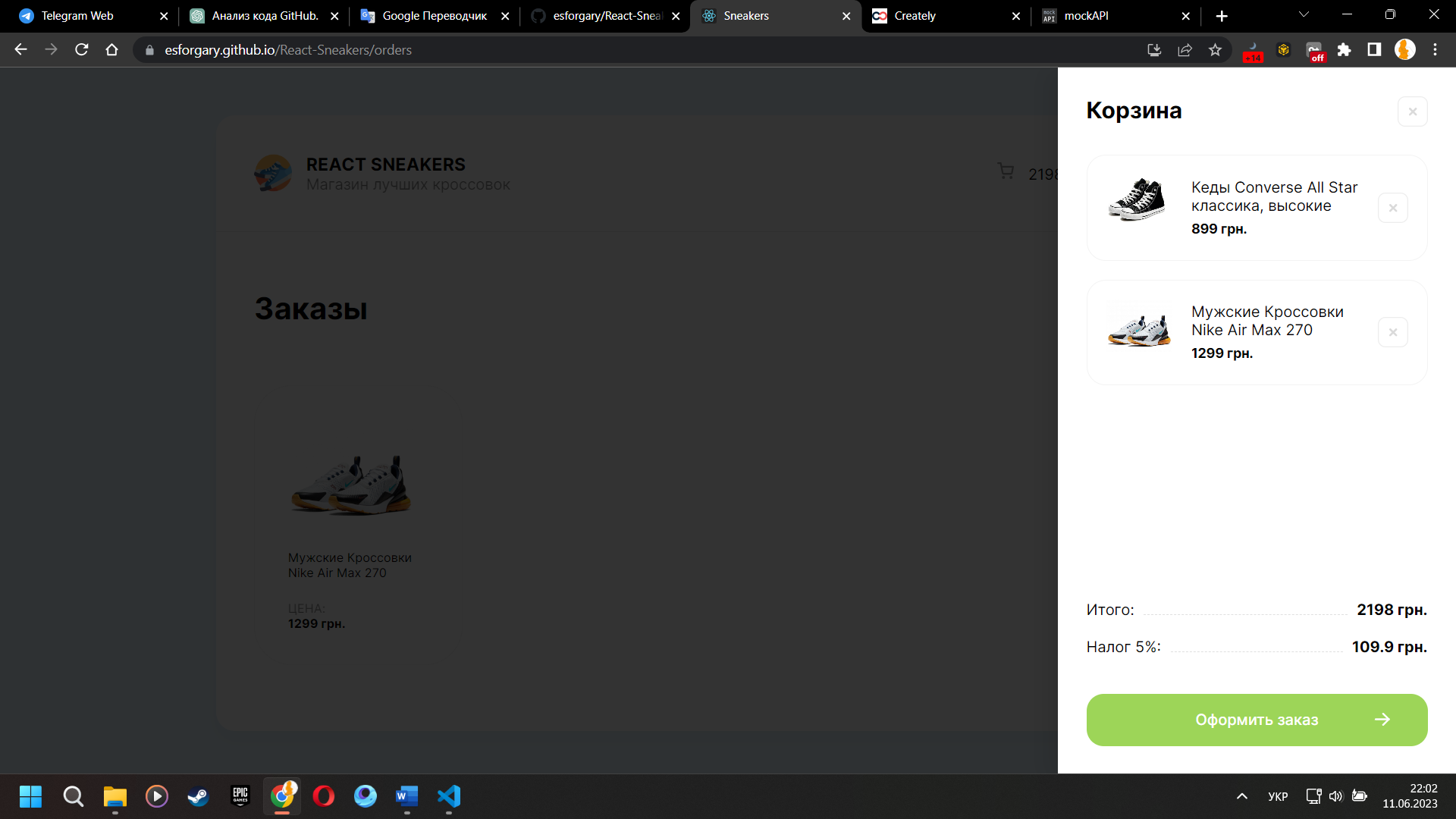


Рисунок 3.3 – Кошик

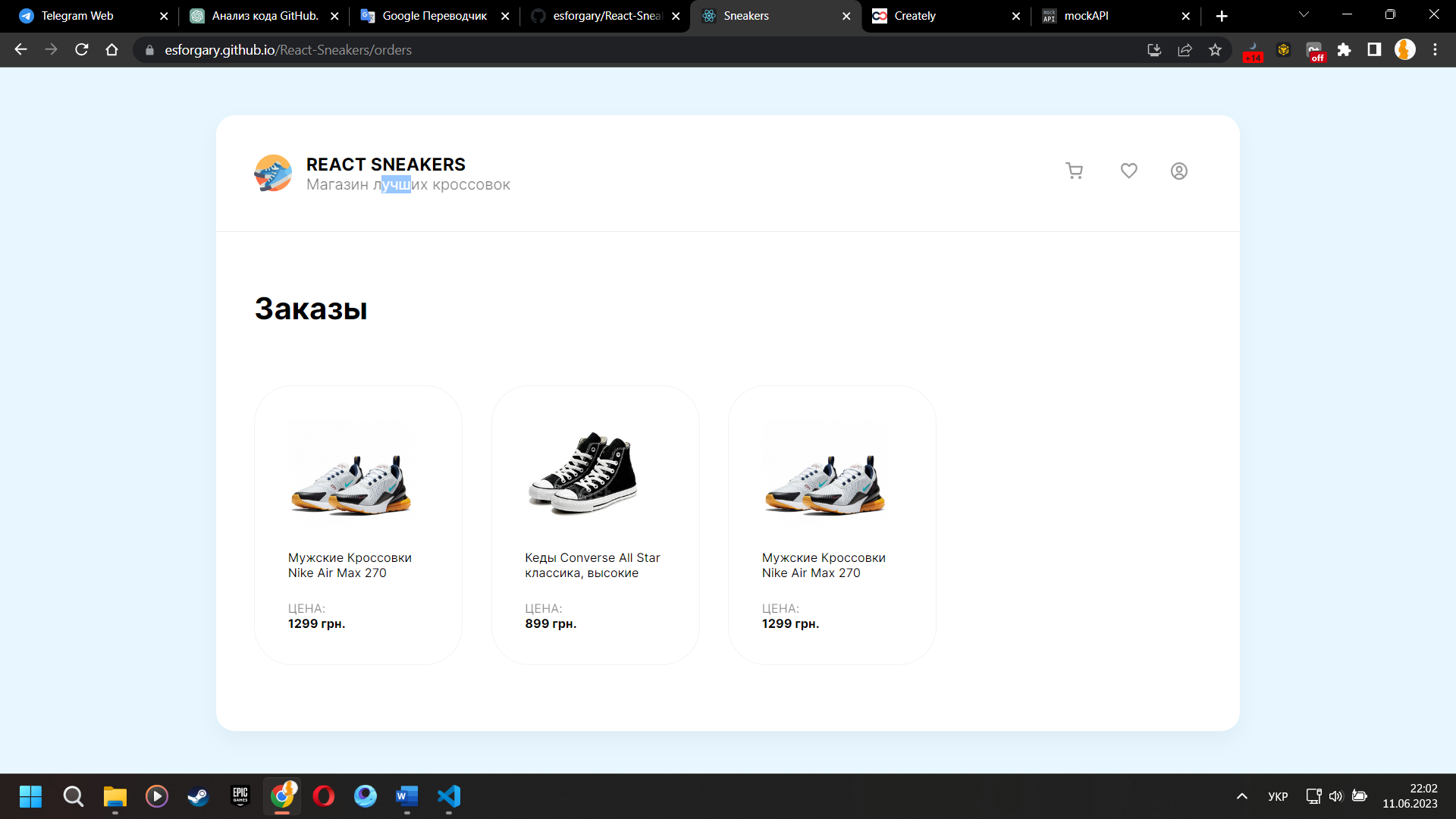


Рисунок 3.4 – Сторінка здійснених замовлень

# **ВИСНОВКИ**

У рамках курсового проекту було виконане наступне:

- вивчено такі теоретичні питання: теорію програмування; основні принципи рефакторингу коду; патерни програмування JavaScript; принципи створення мікросервісів;

- ознайомлено з наступними інструментами проектування та програмування: середовищем виконання JavaScript React, MS Visual Studio Code;

- застосовано отримані теоретичні та практичні знання для розробки веб-сайту;

- сформовано структуру веб-сайту;

- розроблено дизайн веб-сайту;

- розроблено програмний код для забезпечення необхідної функціональності веб-сайту з використанням патернів програмування та мікросервісу;

- проведено тестування проекту;

- відображено процес та результати розробки веб-сайту у пояснювальній записці до курсового проекту.

Для розробки веб-сайту було використано такі інструменти:

Visual Studio Code, JavaScript – середовище та мова розробки програмного забезпечення;

React – фреймворк мови JavaScript;

SCSS – препроцесор мови декорування;

HTML – мова гіпертекстової розмітки.

# **ПЕРЕЛІК ДЖЕРЕЛ ТА ПОСИЛАННЯ**

1. Observer паттерн в Javascript. Retrieved from

URL: https://monsterlessons.com/project/lessons/observer-pattern-v-javascript

1. Паттерны проектирования: для чего нужны, какими бывают и чем отличаются от архитектурных. Retrieved from

URL: https://academy.mediasoft.team/article/patterny-proektirovaniya-dlya-chego-nuzhny-kakimi-byvayut-i-chem-otlichayutsya-ot-arkhitekturnykh/#:~:text=%D0%9F%D0%B0%D1%82%D1%82%D0%B5%D1%80%D0%BD%D1%8B%20%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%20%D0%BF%D1%80%D0%B5%D0%B4%D0%BE%D1%81%D1%82%D0%B0%D0%B2%D0%BB%D1%8F%D1%8E%D1%82%20%D1%80%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D1%8F%20%D0%B4%D0%BB%D1%8F,%D0%BF%D1%80%D0%B8%D0%BD%D0%B8%D0%BC%D0%B0%D0%B5%D1%82%20%D1%80%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B5%2C%20%D0%BA%D0%B0%D0%BA%D0%BE%D0%B9%20%D0%BF%D0%B0%D1%82%D1%82%D0%B5%D1%80%D0%BD%20%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D1%8C.

1. Напишите свой первый микросервис на JavaScript. Retrieved from

URL: https://medium.com/nuances-of-programming/%D0%BD%D0%B0%D0%BF%D0%B8%D1%88%D0%B8%D1%82%D0%B5-%D1%81%D0%B2%D0%BE%D0%B9-%D0%BF%D0%B5%D1%80%D0%B2%D1%8B%D0%B9-%D0%BC%D0%B8%D0%BA%D1%80%D0%BE%D1%81%D0%B5%D1%80%D0%B2%D0%B8%D1%81-%D0%BD%D0%B0-javascript-67fddb23dbdd

1. Додаток А

**ІНСТРУКЦІЯ КОРИСТУВАЧУ**

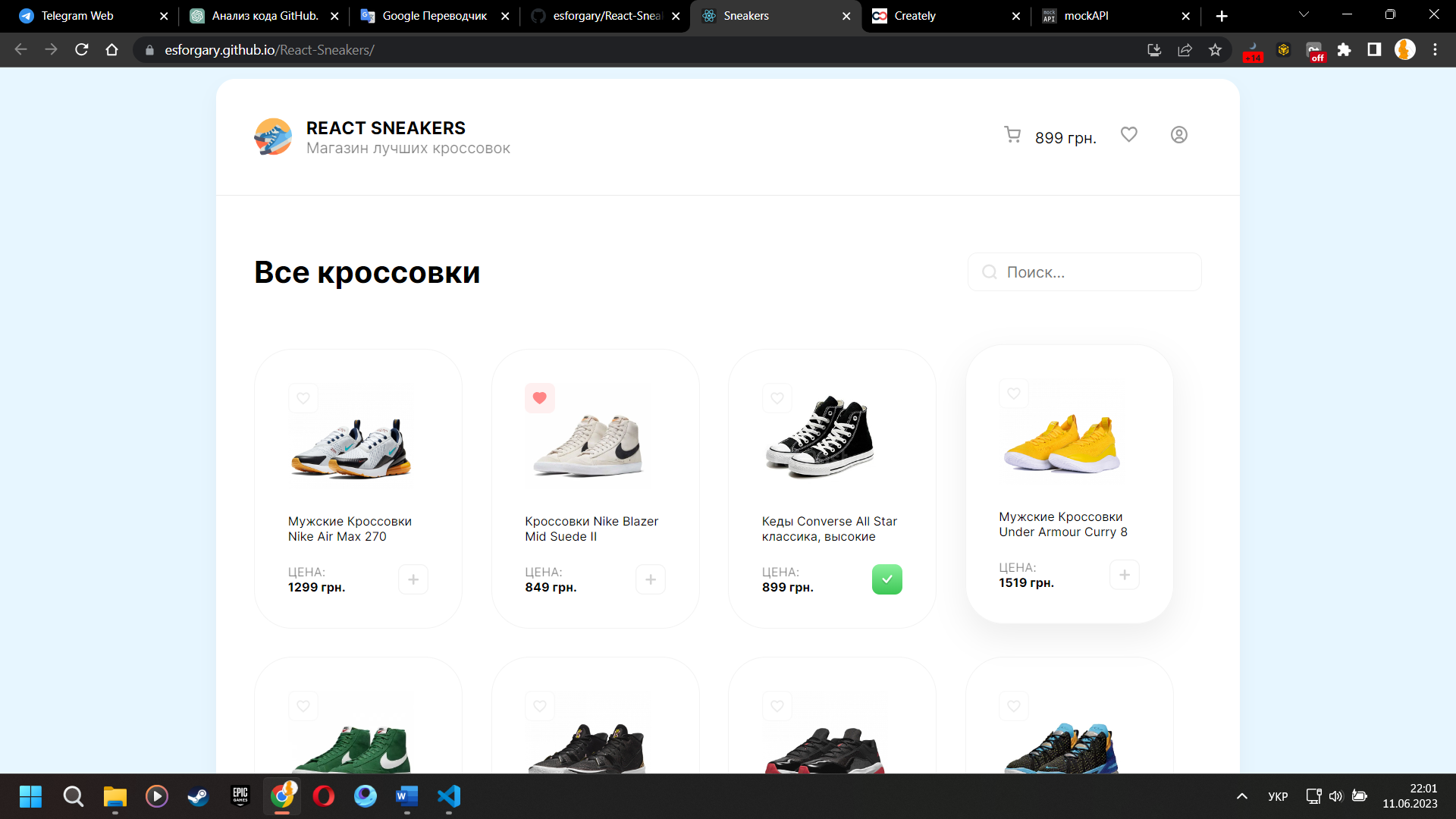


Рисунок А.1 – Головна сторінка на якій знаходяться усі товари. При натисканні на кнопку «❤️» товар додається у «обрані товари». При натисканні на кнопку «✅» товар додається до кошику.

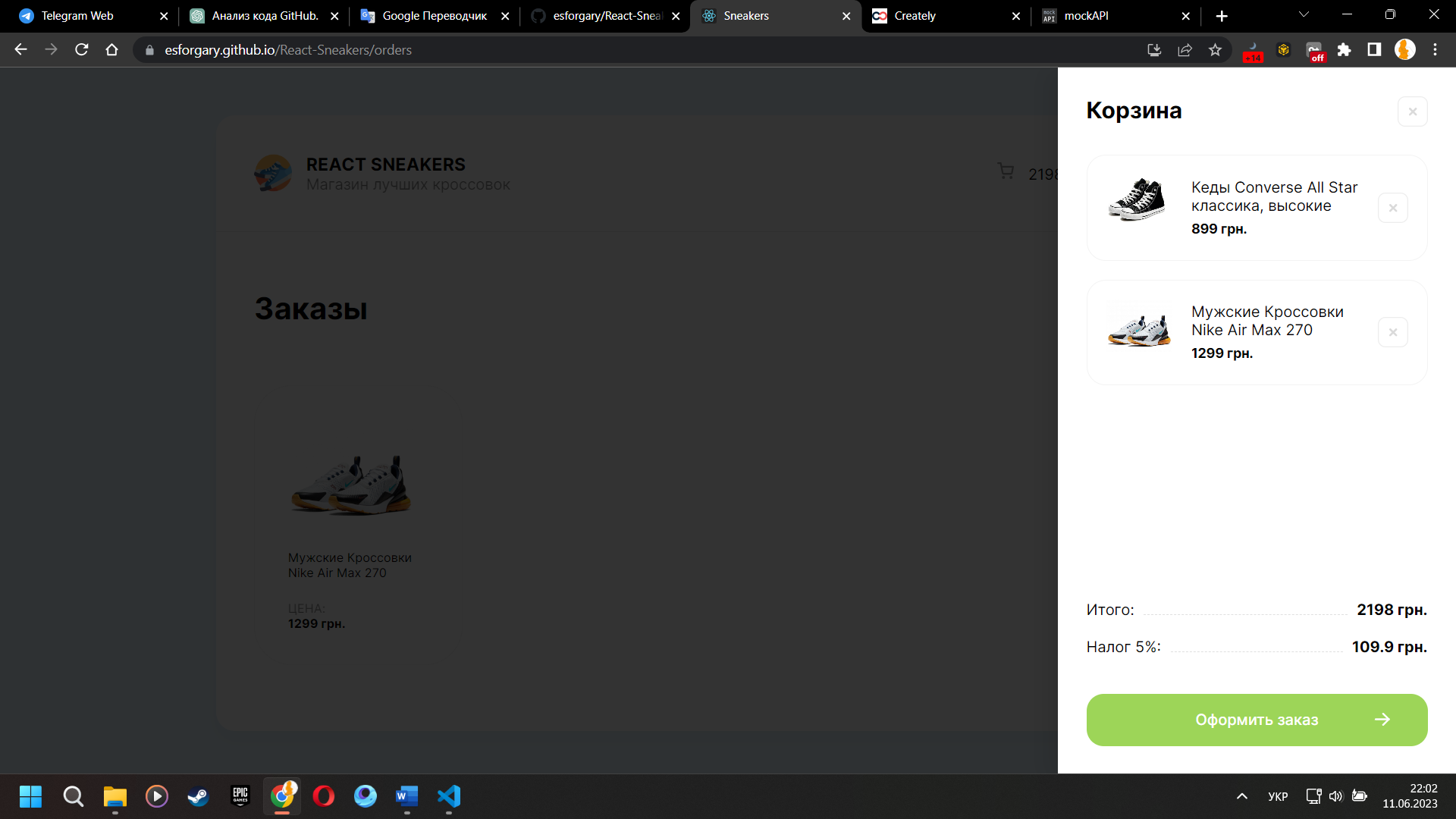


Рисунок А.2 – Кошик з доданими до нього товарами. У кошику відображається загальна сума товару, податкова плата та присутня кнопка «оформити замовлення».

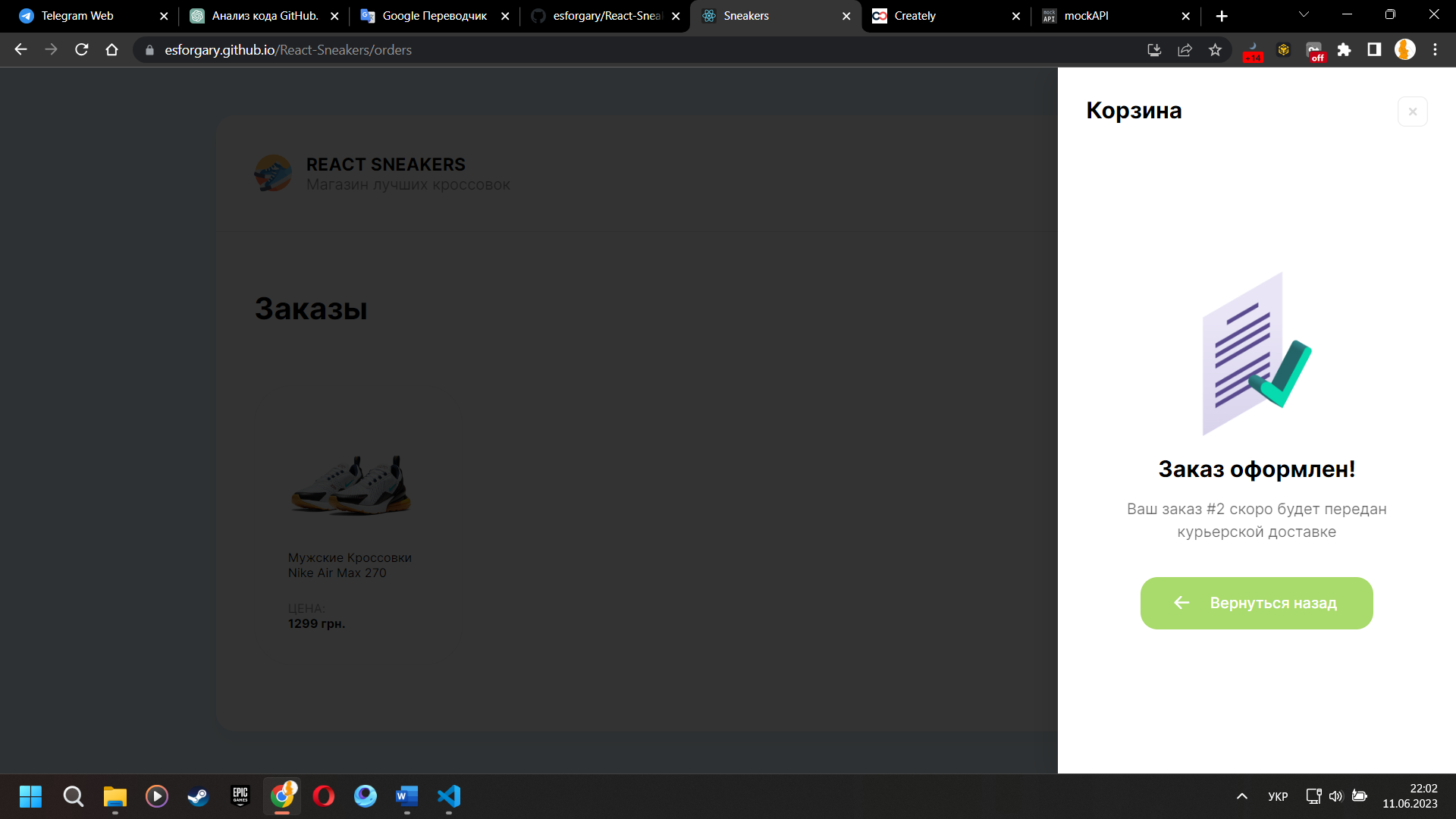


Рисунок А.3 – Після оформлення замовлення кошик спорожнюється і замість минулого інтерфейсу з’являється кнопка, яка повертає користувача до головної сторінки магазину.

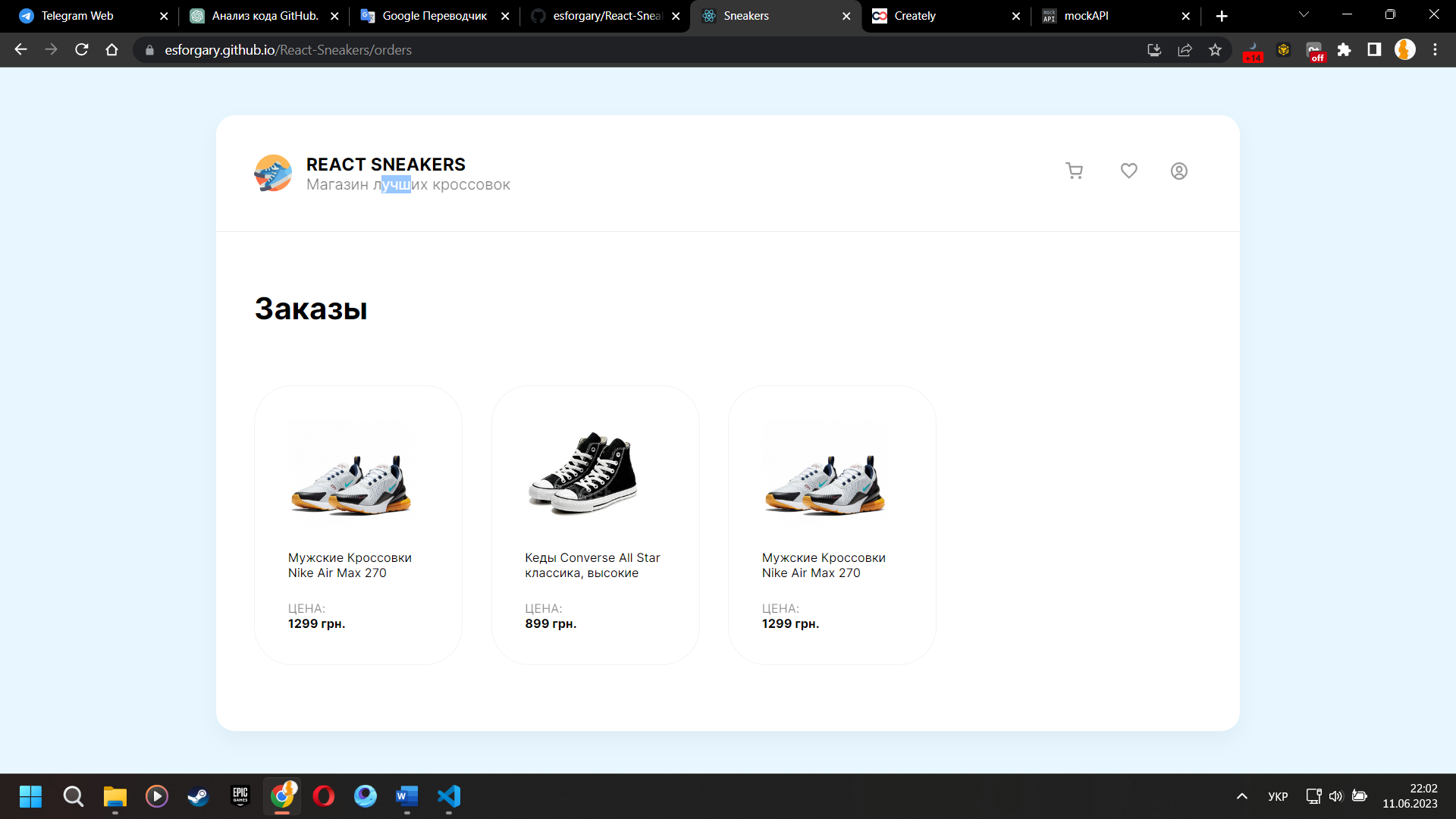


Рисунок А.4 – Сторінка заказів, на якій можна переглянути усі замовлення, які коли-небудь здійснювалися з цього акаунту.