УДК 004.9 DOI: 10.31673/2786-8362.2024.011010

Шикула О.М., д.ф.-м.н.; **Білоусова С.В.,** к.ф.-м.н.
Блєднов О.С., Бондаренко К.І.

РОЗРОБКА API ДЛЯ ОНЛАЙН-МАГАЗИНУ

Bondarenko K.I., Shykula O.M., Blednov O.S., Bilousova S.V. API development for an online store. The article examines contemporary approaches to developing online stores, with a primary focus on crafting APIs to ensure the effective operation of the store in the online realm. A survey of prevalent solutions and literature in the field has enabled the identification of key challenges and unresolved issues.

Through the analysis, it was discerned that many small online stores encounter speed and scaling issues attributed to non-optimized API development. The article underscores the significance of leveraging modern technologies, such as Node.js, JavaScript, and PostgreSQL, to streamline development and enhance productivity.

The article outlines the work's objective—creating an efficient API for an online store—and defines the research task. It elucidates the importance of utilizing Node.js, JavaScript, and other tools to secure endpoints with Guards, thereby improving system security and reliability.

In the Development Environment and Tooling section, a range of tools, including JavaScript, Node.js, Nest.js, PostgreSQL, Git, and WebStorm, are detailed for implementing the online store's API. Emphasis is placed on how these tools contribute to ensuring system reliability and speed. The adoption of the Nest.js framework is highlighted as pivotal in crafting a secure and efficient API for the online store. The incorporation of Guards effectively segregates controllers and provides robust protection for endpoints. Consequently, this integration has simplified the enhancement of the store's functionality, rendering it more reliable and flexible for future developments.

Keywords: online store, API, Nest.js, JavaScript, Node.js, PostgreSQL, Endpoints, Guards, JWT token, Security, Reliability

Бондаренко К.І., Шикула О.М., Блєднов О.С., Білоусова С.В. Розробка API для онлайн-магазину. У статті розглядаються сучасні підходи до розробки інтернет-магазинів, при цьому основна увага приділяється розробці API для забезпечення ефективної роботи магазину в онлайн-сфері. Огляд поширених рішень і літератури в цій галузі дозволив визначити ключові проблеми та невирішені проблеми.

Під час аналізу було виявлено, що багато невеликих онлайн-магазинів стикаються зі швидкістю та проблемами масштабування, пов’язаними з неоптимізованою розробкою API. У статті підкреслюється важливість використання сучасних технологій, таких як Node.js, JavaScript і PostgreSQL, для оптимізації розробки та підвищення продуктивності.

У статті окреслено мету роботи — створення ефективного API для інтернет-магазину — та визначено завдання дослідження. Він пояснює важливість використання Node.js, JavaScript та інших інструментів для захисту кінцевих точок за допомогою Guards, тим самим покращуючи безпеку та надійність системи.

У розділі «Середовище розробки та інструменти» детально описано низку інструментів, включаючи JavaScript, Node.js, Nest.js, PostgreSQL, Git і WebStorm, для реалізації API онлайн-магазину. Акцент робиться на тому, як ці інструменти сприяють забезпеченню надійності та швидкості системи. Прийняття фреймворку Nest.js виділяється як ключове у створенні безпечного та ефективного API для онлайн-магазину. Включення Guards ефективно відокремлює контролери та забезпечує надійний захист для кінцевих точок. Таким чином, ця інтеграція спростила вдосконалення функціональності магазину, зробивши його більш надійним і гнучким для майбутніх розробок.

Ключеві слова: онлайн-магазин, API, Nest.js, JavaScript, Node.js, PostgreSQL, Endpoints, Guards, JWT- token, безпека, надійність

Вступ

Сучасний етап розвитку електронної комерції визначає важливість належного функціонування онлайн-магазинів. Однак, разом зі зростанням їх популярності, з'являється необхідність у вдосконаленні та оптимізації процесів управління та взаємодії з покупцями. Одним із ключових інструментів для досягнення цих цілей є розробка Application Programming Interface (API) [1] для онлайн-магазинів.

Актуальність проблеми полягає в необхідності створення ефективного механізму, який дозволить забезпечити високий рівень взаємодії між різноманітними компонентами онлайн-магазину. Відсутність чіткого та ефективного інтерфейсу може призвести до розриву у взаємодії, а це, в свою чергу, може вплинути на якість обслуговування клієнтів та загальну ефективність бізнесу. Незважаючи на існуючі розробки, існують невирішені питання, пов'язані з оптимізацією процесів обробки запитів, підвищенням ефективності та забезпеченням високої стабільності API під час масштабування. Також актуальною є проблема забезпечення сумісності API з різноманітними платформами та сервісами, що розширює можливості використання.

Метою даної статті є детальний аналіз та розгляд основних проблем, пов'язаних з розробкою API для онлайн-магазинів. Також висвітлено найбільш важливі аспекти створення ефективного та гнучкого інтерфейсу, а також розглянуто стратегії вирішення актуальних питань у даній галузі. Це дозволить розширити розуміння та сприятиме подальшому розвитку електронної комерції, роблячи її більш доступною та ефективною для бізнесу та споживачів.

Постановка завдання. У галузі електронної комерції великі онлайн-магазини успішно впораються з дизайном та серверною інфраструктурою, проте розробка ефективного API для менших та менш відомих платформ залишається важливим завданням. Однак замість вдосконалення технічних аспектів багато малих платформ надають перевагу дизайну та інтерфейсу, що призводить до ряду проблем. У багатьох випадках інфраструктура маленьких платформ не забезпечує належних умов для розробки та підтримки високоефективного API. Відсутність ресурсів та можливостей може обмежити їх здатність створювати інноваційні та ефективні інтерфейси.

Ще однією проблемою є відсутність оптимізації швидкості обміну даними через API. Магазини, які зосереджуються виключно на дизайні, часто мають труднощі зі збереженням високої швидкості роботи та оперативною відповіддю на запити клієнтів. Масштабованість також стає проблемою, коли магазин зростає в обсягах продажів. Відсутність правильного планування можливостей масштабування може призвести до труднощів при збільшенні обсягів роботи.

Нарешті, питання безпеки даних [2] залишається актуальним. Занедбання аспектів безпеки може вплинути на конфіденційність інформації клієнтів, ставлячи під загрозу репутацію магазину та його здатність залучати нових клієнтів.

Отже проблеми, пов'язані з розробкою API для малих онлайн-магазинів, свідчать про необхідність поглибленого вивчення технічної інфраструктури та інтеграції ефективних рішень для забезпечення стабільності, швидкості та безпеки у роботі цих платформ.

Метою роботи є створення універсального та надійного інтерфейсу для взаємодії з іншими сервісами, спрямованого на поліпшення функціональності та конкурентоздатності магазину.

В ході дослідження передбачається розробка API з використанням технологій Node.js, JavaScript та PostgreSQL [3-5]. Одним із основних завдань є забезпечення швидкого та надійного обміну даними між онлайн-магазином та іншими системами. Особлива увага приділяється використанню Node.js та JavaScript для забезпечення ефективної швидкодії та легкості розширення. Інтеграція з PostgreSQL, реляційною системою управління базами даних, визначається як ключовий елемент розробки, спрямований на забезпечення надійного зберігання та оптимальну обробку інформації.

Також висвітлюється питання безпеки даних, де визначається необхідність застосування шифрування та автентифікації для забезпечення конфіденційності під час обміну інформацією через API. Однією з ключових задач є створення гнучкого та розширюваного API, яке легко інтегрується з новими функціями та забезпечує можливість швидкої адаптації до змін в умовах ринку.

Загалом, мета дослідження визначається як створення ефективного інструменту для взаємодії онлайн-магазину з іншими сервісами, що дозволить підвищити його функціональність та забезпечити конкурентні переваги.

Виклад основного матеріалу дослідження.

Середовище та інструментальні засоби розробки для створення API онлайн-магазину. У процесі розробки API для онлайн-магазину, визначено низку середовищ та інструментів, які забезпечать ефективну та надійну реалізацію програмного забезпечення.

Для розробки бек-енду частини API використовується мова програмування JavaScript в середовищі виконання Node.js. Node.js вибирається через свою високу продуктивність, асинхронність та широкий спектр доступних бібліотек.

У якості фреймворку для створення веб-застосунків вибраний NestJS [6]. NestJS базується на Node.js і забезпечує платформу для побудови масштабованих та модульних серверних додатків. Вибір NestJS обумовлений його підтримкою TypeScript, яка дозволяє покращити стабільність та підтримку коду.

В якості системи управління базами даних використовується PostgreSQL. PostgreSQL відзначається високою надійністю, підтримкою SQL [7] та розширеними можливостями для оптимізації запитів.

Для контролю версій та спільної роботи над кодом використовується система контролю версій Git. Git забезпечує ефективну спільну роботу, відслідковування змін та розвиток кодової бази.

Інтегроване середовище розробки (IDE) WebStorm вибрано для комфортної та продуктивної роботи з кодом. WebStorm підтримує JavaScript, TypeScript, Node.js та інші технології, що використовуються в даному проекті.

Зазначене середовище та інструментальні засоби були вибрані внаслідок їхньої надійності та здатності сприяти швидкій та ефективній розробці API для онлайн-магазину.

Створення бази даних. Під час розробки програмного продукту було створено базу даних, що включає чотири основні таблиці. (рис. 1):

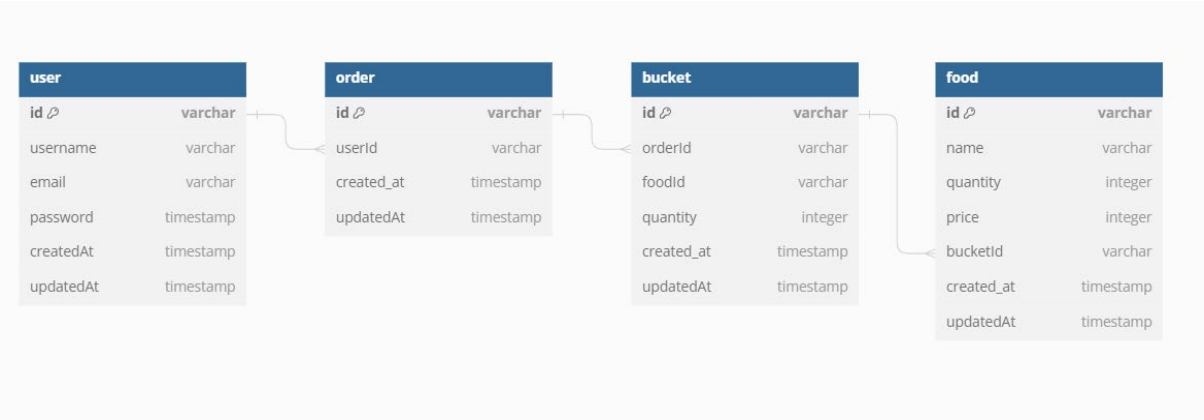

Рис. 1. Схема бази даних

- User – містить інформацію про користувача, його email та пароль;
- Order – містить інформацію про замовлення;
- Bucket – містить інформацію про склад корзини;
- Food – містить інформацію про товар та його ціну, та наявність на складі.

Розробка програмного забезпечення. При розробці програмного продукту були спроектовані та розроблені наступні класи (рис. 2):

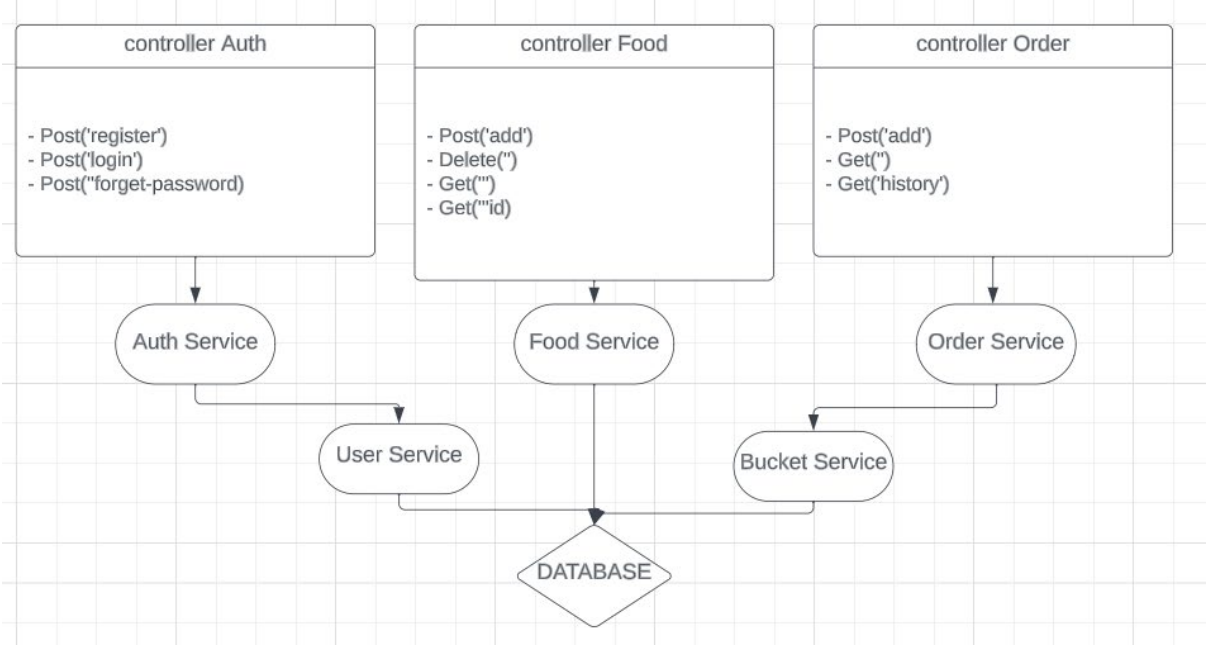

Рис. 2. Діаграма класів

- Класс Auth – контроллер для всіх запитів авторизації та аутентифікації [8];
- Класс Food – контроллер для всіх запитів товарів;
- Класс Order – контроллер для всіх запитів замовлення;
- Auth Service – бізнес логіка авторизації та аутентифікації;
- Food Service – бізнес логіка товарів;
- Order Service – бізнес логіка замовлення;
- User Service – бізнес логіка користувачів;
- Bucket Service – бізнес логіка корзини та історії.

Захист endpoints: використання Guard у веб-розробці. Перевага використання Nest.js в сфері веб-розробки надає значущий функціонал для ефективного впровадження Guard [9]. Nest.js, будучи фреймворком для побудови масштабованих та модульних додатків, забезпечує зручний інтерфейс для створення та застосування Guards.

Декоратори та вбудовані функції Nest.js дозволяють легко інтегрувати Guards в різноманітні частини додатку, надаючи гнучкий механізм контролю доступу та забезпечення надійності. Це робить процес розробки безпечним та ефективним, здійснюючи надійний захист endpoints від несанкціонованого доступу.

У світі сучасної веб-розробки, де безпека є пріоритетом, використання Guard стає важливою складовою для захисту endpoints системи. Розглянемо, як Nest.js Guards стає відмінним інструментом для забезпечення безпеки та автентифікації у веб-додатках.

Nest.js Guard – це елемент маршрутизації та захисту endpoints у веб-додатках, який забезпечує обмеження доступу до конкретних частин додатку. Використання Guard є ключовим для запобігання несанкціонованого доступу, контролювання прав користувачів та впровадження системи безпеки.

Розглянемо роботу Nest.js Guard.

1. Аутентифікація: Guard використовується як декоратор над конкретним endpoint. У випадку, якщо користувач намагається звернутися до цього endpoint, Guard перевіряє наявність аутентифікаційного токену, який гарантує, що користувач є дійсним і залогіненим у системі.

2. Авторизація: Гуард також відповідає за перевірку прав доступу користувача. Використовуючи ролі та привілеї, визначені у токені, Guard визначає, чи користувач має відповідні дозволи для отримання доступу до даного endpoint.

3. Логування та моніторинг: Guard може вести журнал спроб доступу, реєструючи події аутентифікації та авторизації. Це надає можливість виявити потенційні загрози та вжити відповідних заходів безпеки.

Практичне використання Nest.js Guards в розробленому веб-додатку стало ключовим елементом для підвищення рівня безпеки та надійності. Переважна більшість endpoints, які надають доступ до функціоналу додатку, знаходяться під захистом Guards. Це важливий крок у напрямку запобігання несанкціонованому доступу та забезпечення конфіденційності користувальницької інформації. Guards стали надійним засобом фільтрації та контролю прав доступу, роблячи систему менш вразливою до потенційних загроз та забезпечуючи користувачів високим рівнем безпеки та впевненості у захищеності їх даних.

Інструкція користувача по використанню системи. Перед тим як розпочати використання розробленого веб-додатку, слід виконати кілька важливих кроків. Спочатку необхідно запустити сервер, що відповідає за обробку запитів та забезпечення безперебійної роботи системи. Після запуску сервера користувач може надсилати запити на доступ до різних endpoints, але перед цим слід отримати JWT–token [10], який буде використовуватися для аутентифікації та авторизації.

Процедура отримання JWT-tokena передбачає відправлення запиту на один із endpoints системи, який відповідає за аутентифікацію. Цей JWT-token буде ключем для доступу до функціоналу системи, але його отримання вважається обов'язковим кроком для подальшого користування системою. Без наявності дійсного JWT-tokena, Guards не надаватимуть доступ до більшості функцій додатку, гарантуючи тим самим захист від несанкціонованого використання системи.

Перш за все, нам потрібно зайти у систему, тож відправляємо POST запит [11] на адресу localhost:3000/auth/login (рис. 3). У відповіді отримаємо JWT-token, який потрібен для наступних запитів.

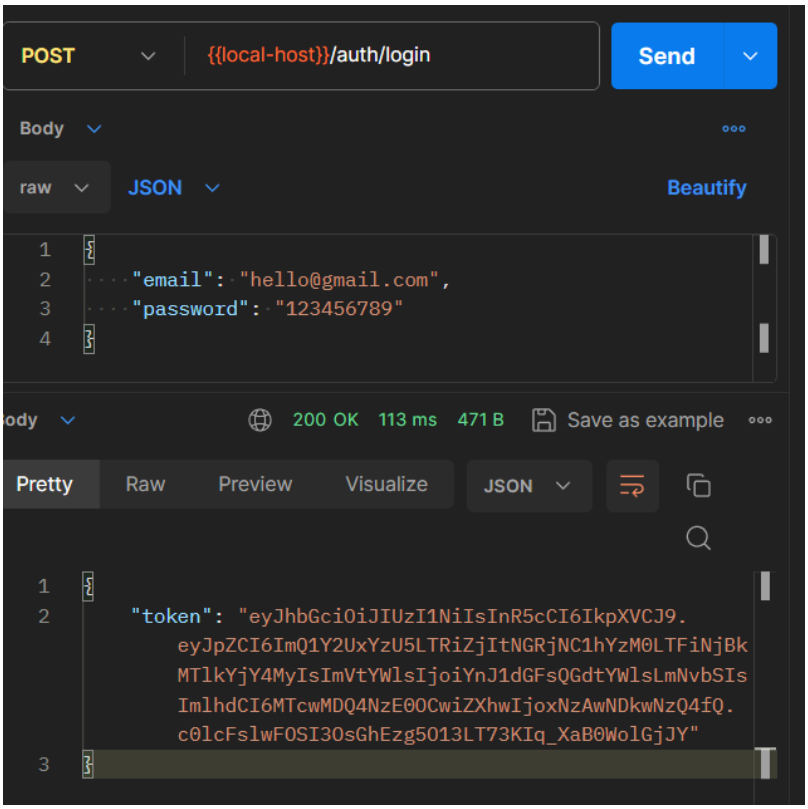

Рис. 3. Запит на отримання JWT–token

Тепер потрібно створити нову корзину, щоб користувач міг додати товари в неї і оформити в кінці покупку. Для цього потрібно відправити POST запит на адресу localhost:3000/order/create. У відповіді отримаємо інформацію про нову корзину (рис. 4).

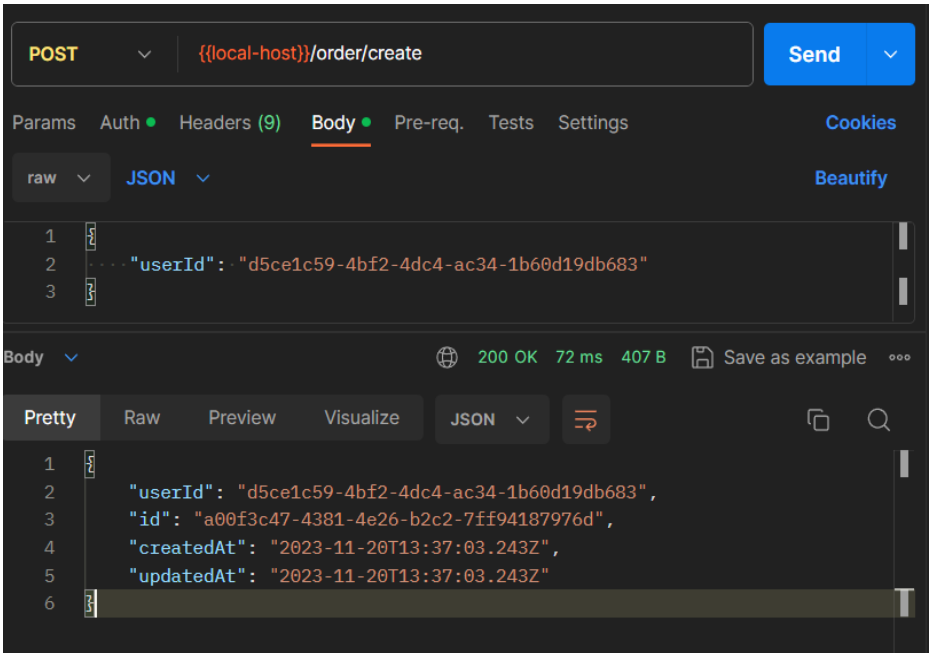

Рис. 4. Запит на створення нової корзини

Далі користувач може додати до своєї корзини товари, але спочатку потрібно щоб ці товари з’явились в магазині. Для цього адміністратор магазину використовує адмінпанель, до якої користувач не має доступу. Завдяки Nest.js можна швидко та притримуючись вимог безпеки реалізувати таку бізнес-логіку, що в інших фрейморках займає доволі досить часу. Адміністратор повинен надіслати POST-запит за адресою localhost:3000/food/add, вказавши товар, його ціну та кількість, щоб додати його на склад. (рис. 5).

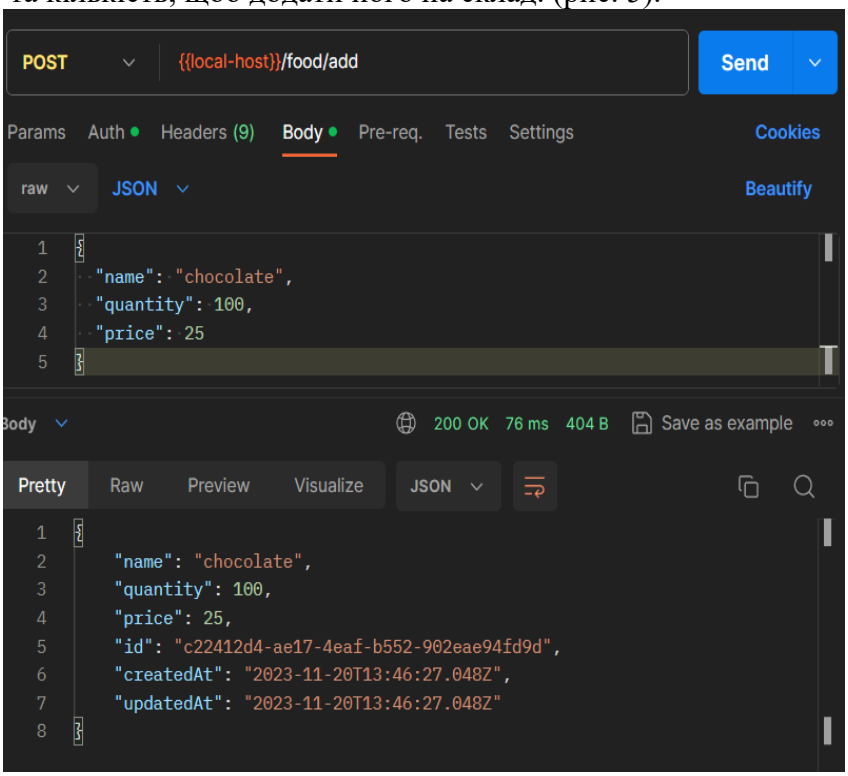

Рис. 5. Запит про додавання товару на склад

Тепер користувач може додати це товар собі у корзину, для цього потрібно відправити POST запит на адресу localhost:3000/order/add (рис. 6).

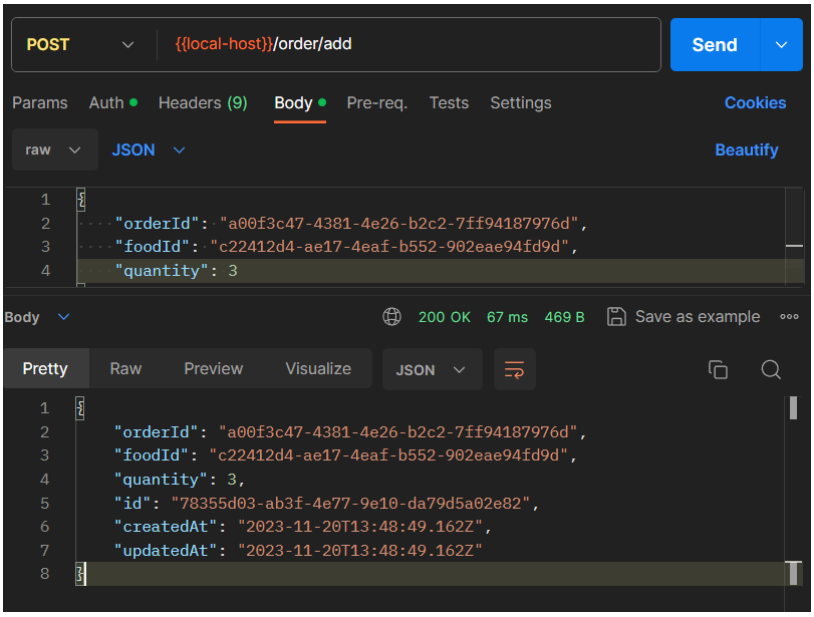

Рис. 6. Додавання товару в корзину

Таким чином користувач може використовувати й інші endpoints, щоб видалити товар з корзини, або подивитись на вміст корзині, прогледити історію замовлень та ін. Адміністратор в свою чергу може додати або видалити товар зі складу, подивитись скільки товару залишилось тощо.

Висновки

У ході розробки онлайн-магазину виявлено значущі переваги використання фреймворку Nest.js. Однією з ключових переваг стала можливість створення високонадійної та захищеної системи. Використання Guards та декораторів в Nest.js дозволило ефективно розділити контролери для адміністраторів та звичайних користувачів.

Особливо важливим було виявлення можливостей Nest.js у сфері безпеки, що дало зручний інструментарій для реалізації різноманітних Guards та стратегій авторизації. Це дозволило створити гнучку та надійну систему контролю доступу, забезпечуючи високий рівень безпеки та конфіденційності даних.

Також важливо відзначити, що використання Nest.js полегшило інтеграцію та розширення функціоналу онлайн-магазину. Зручний синтаксис та підтримка об'єктно-орієнтованого підходу дозволили швидко та ефективно розгортати нові функції, що забезпечило швидкий розвиток та адаптацію системи до зростаючих потреб користувачів.

Список використаної літератури:

1. Що таке API? *URL:* https://qalight.ua/baza-znaniy/shho-take-api/
2. Безпека даних – Шифрування критично важливих даних *URL:* https://www.kingston.com/ua/solutions/data-security
3. Сайт Node.js *URL:* https://nodejs.org/en/about
4. JavaScript. Основи веб-програмування *URL:* https://w3schoolsua.github.io/js/index.html#gsc.tab=0.
5. Сайт PostgreSQL *URL:* https://www.postgresql.org/about/
6. Сайт NestJS *URL:* https://docs.nestjs.com/
7. Що таке SQL та Бази даних?https://acode.com.ua/sql-intro/
8. Олексій Васильєв Програмування мовою Java – Тернопіль, Видавництво Богдан, 2019. – 696 с.
9. Сайт NestJS *URL:* https://docs.nestjs.com/guards

10. Сайт JWT.IO *URL:* https://jwt.io/introduction

11. Мельник Р.А. Програмування веб-застосувань (фронт-енд та бек-енд). – Львів: Львівська політехніка, 2018. –248 с.

Автори статті

Бондаренко Кирило – студент, Державний університет інформаційно–комунікаційних технологій, Київ, Україна.

Шикула Олена – доктор фізико–математичних наук, професор, Державний університет інформаційно-комунікаційних технологій, Київ, Україна.

Блєднов Олексій – викладач, Державний університет інформаційно-комунікаційних технологій, Київ, Україна.

Білоусова Світлана – кандидат фізико–математичних наук, доцент, Державний університет інформаційно-комунікаційних технологій, Київ, Україна.

Authors of the article

Bondarenko Kyrylo – student, State University of Information and Communication Technologies, Kyiv, Ukraine.

Shykula Olena – Doctor of Science (physics and mathematics), Professor, State University of Information and Communication Technologies, Kyiv, Ukraine.

Blednov Oleksiy – lecturer, State University of Information and Communication Technologies, Kyiv, Ukraine.

Bilousova Svitlana – Candidate of Science (physics and mathematics), Associate Professor, State University of Information and Communication Technologies, Kyiv, Ukraine.