**Практична робота №2**

Метою даної практичної роботи є здобуття навичок самостійного ознайомлення з популярними бібліотеками та їх застосування, а також налаштування CI/CD в рамках створення програмного прототипу.

**Хід роботи:**

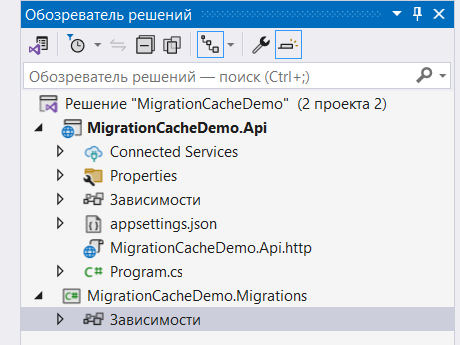


Рис. 1. Результат виконання

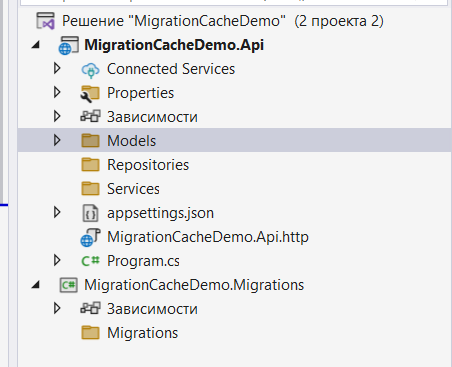


Рис. 2. Результат виконання

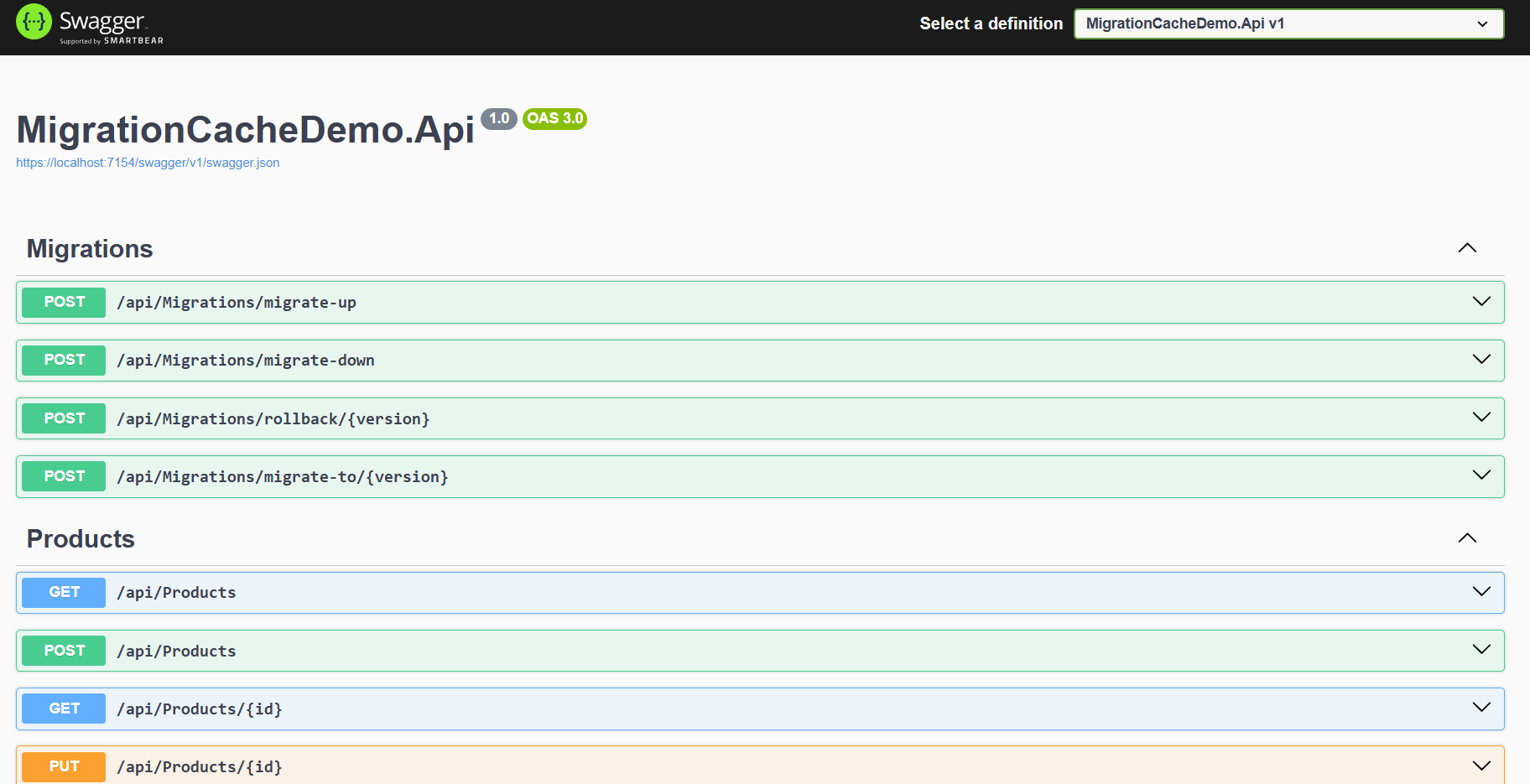


Рис. 3. Результат виконання

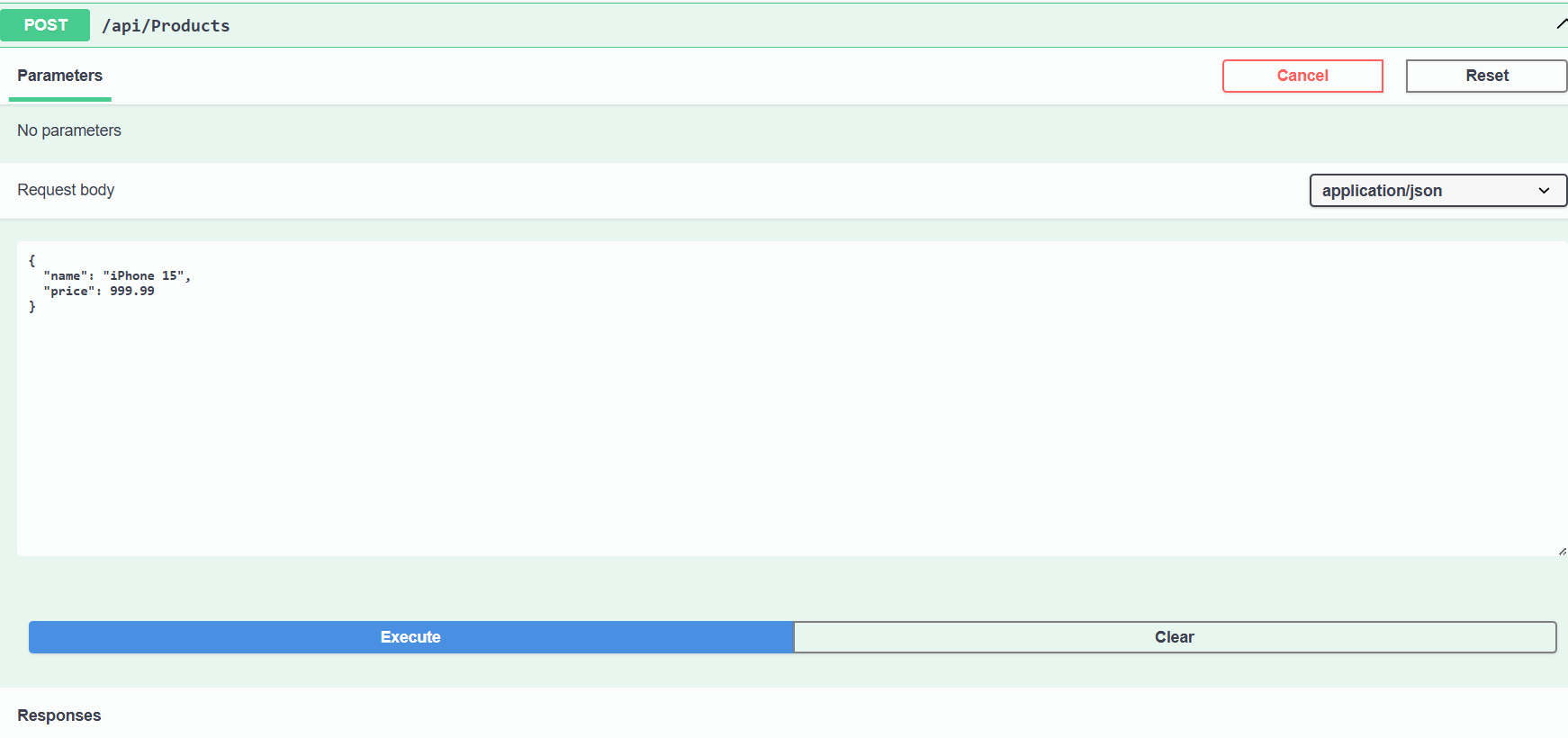


Рис. 4. Результат виконання

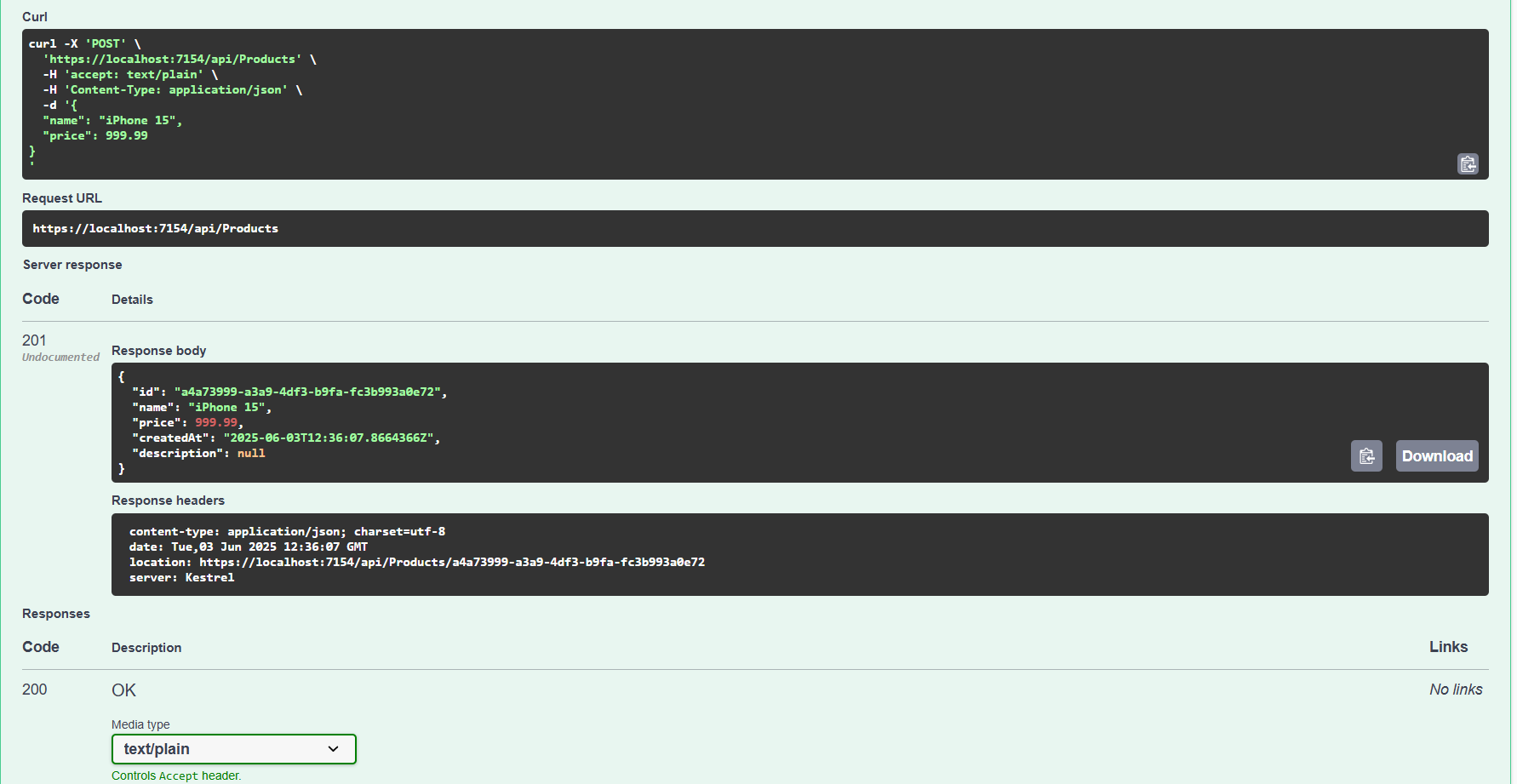


Рис. 5. Результат виконання

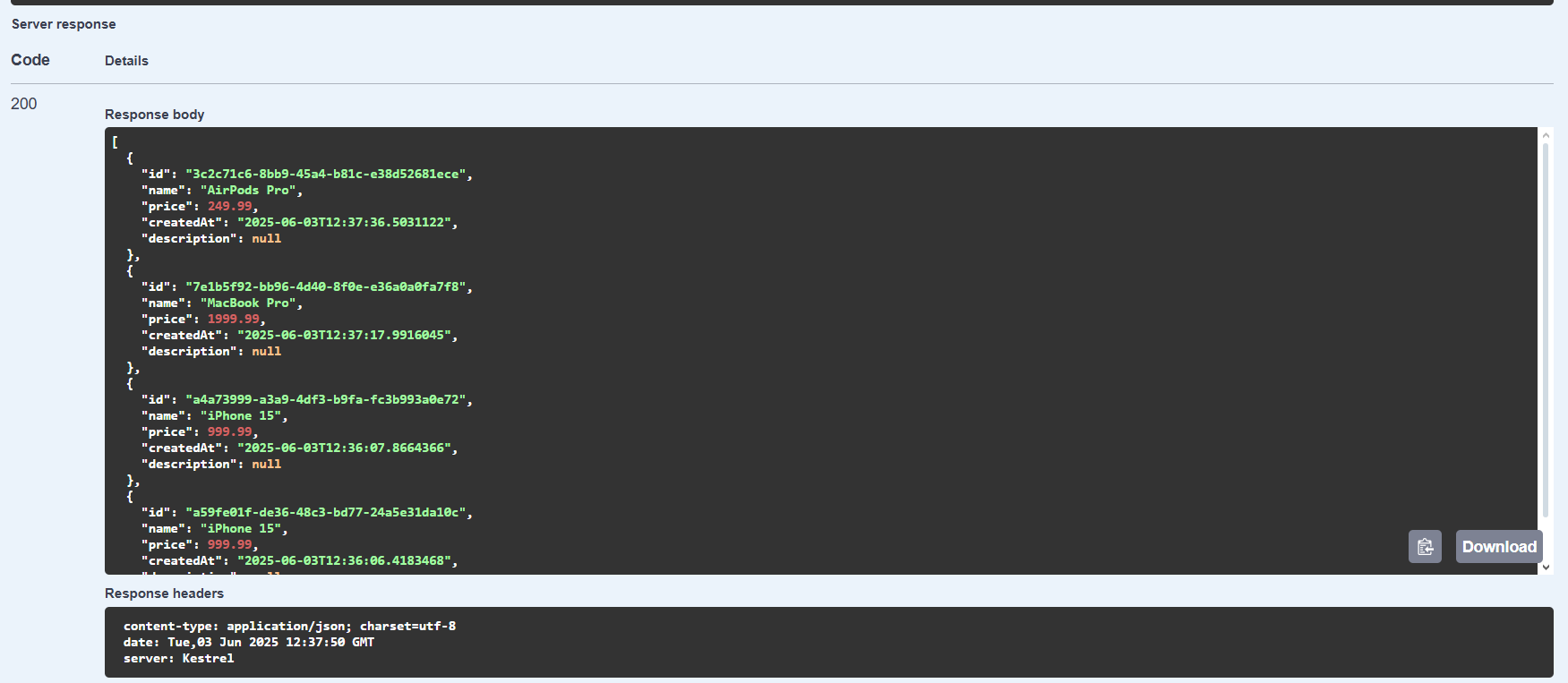


Рис. 6. Результат виконання

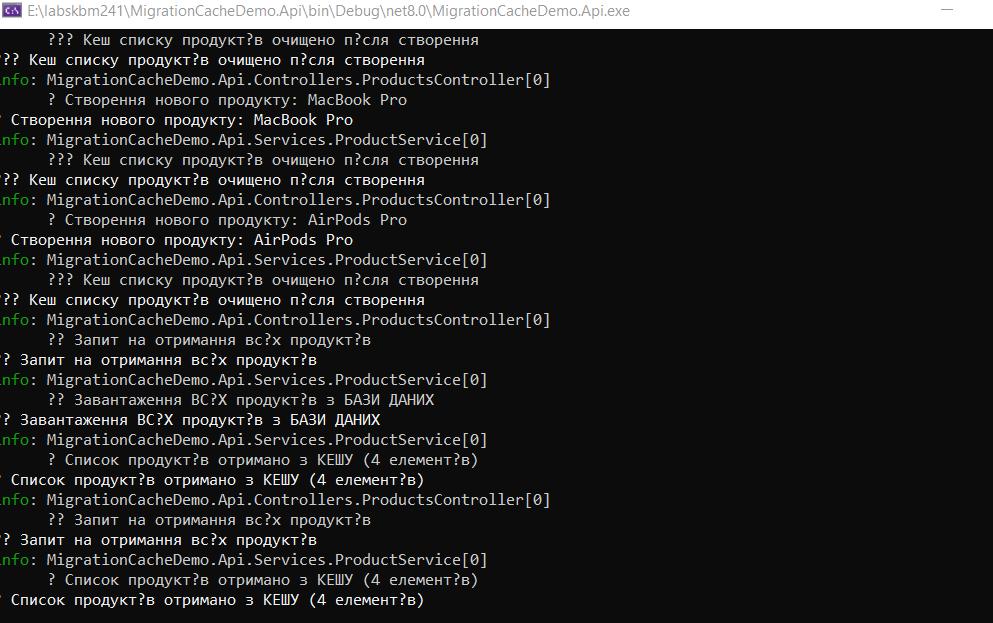


Рис. 7. Результат виконання

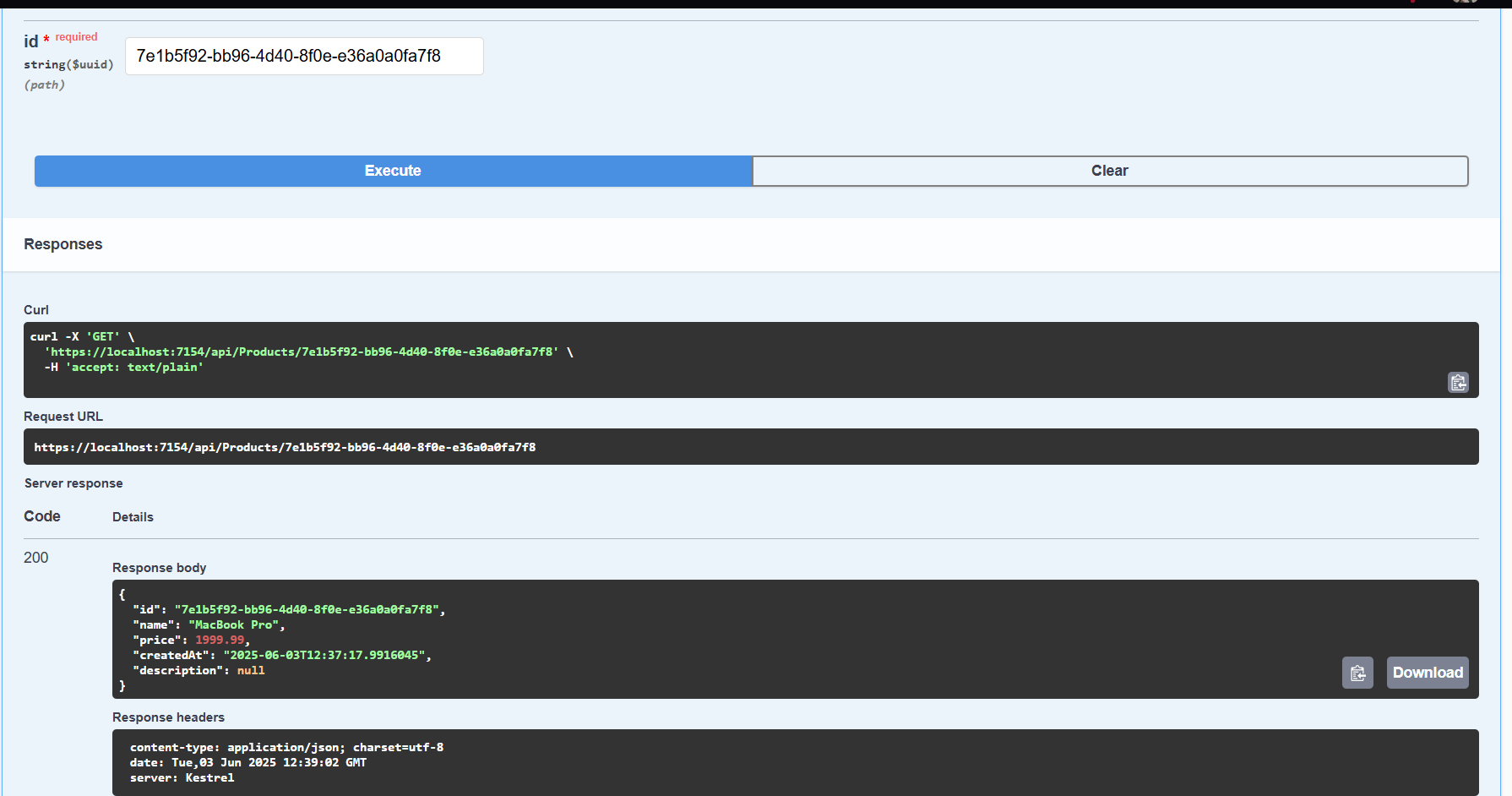


Рис. 8. Результат виконання

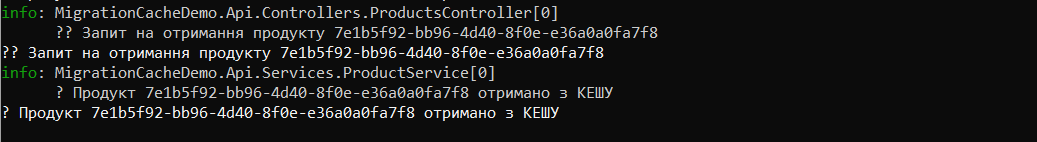


Рис. 9. Результат виконання

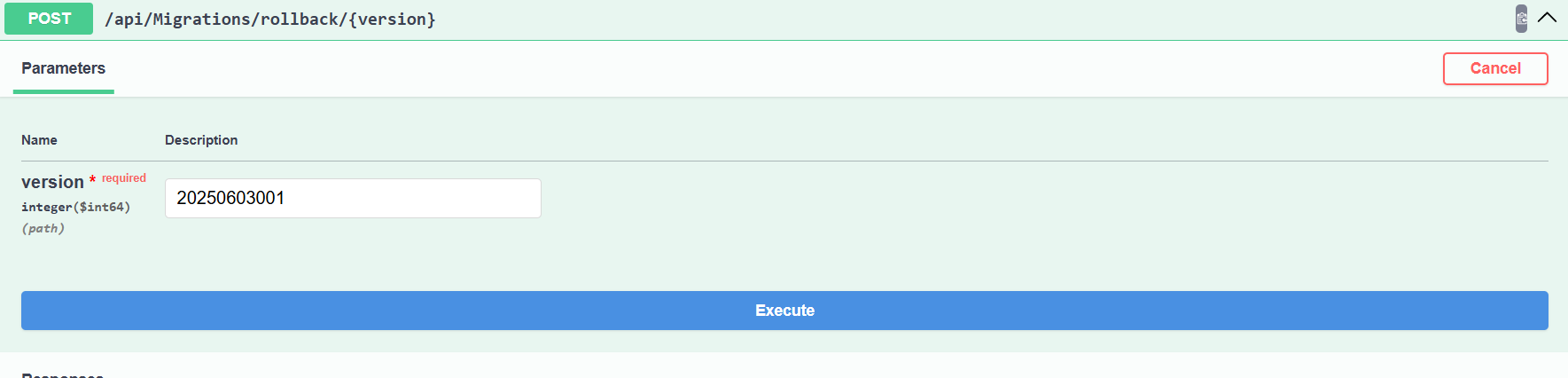


Рис. 10. Результат виконання



Рис. 11. Результат виконання

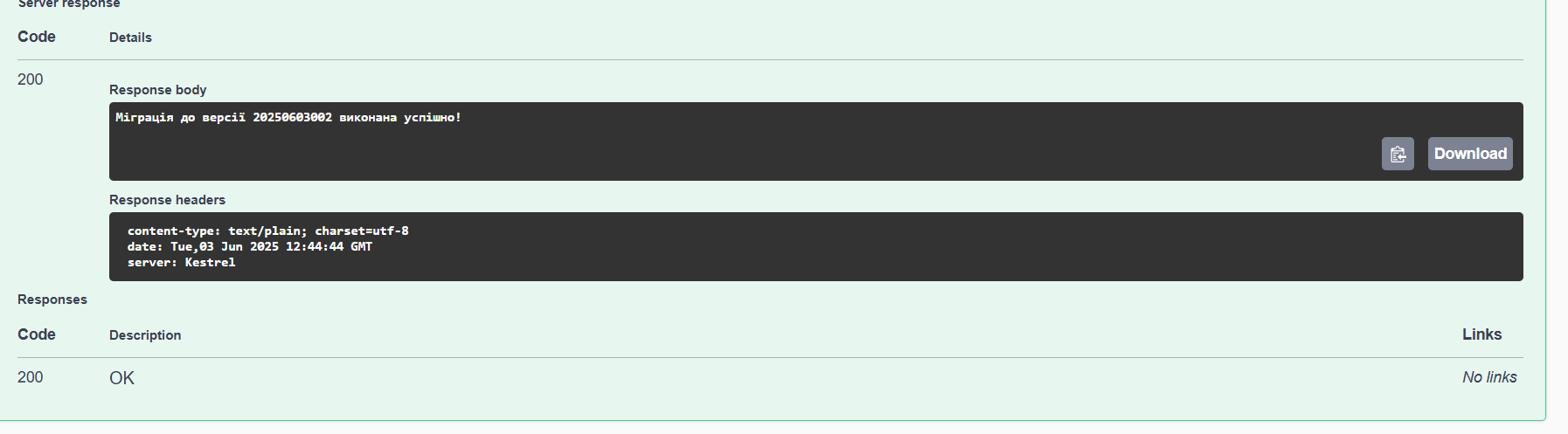


Рис. 12. Результат виконання

Проект було створено з використанням ASP.NET Core Web API з інтеграцією двох ключових бібліотек: FluentMigrator для управління міграціями бази даних та LazyCache для кешування даних.

FluentMigrator є потужним фреймворком для .NET, який дозволяє керувати схемою бази даних через код замість ручного виконання SQL скриптів. Налаштування включало встановлення NuGet пакетів FluentMigrator.Runner та FluentMigrator.Runner.SQLite, після чого було сконфігуровано Dependency Injection в Program.cs для автоматичного виконання міграцій при запуску додатку. Створено дві міграції: базову для створення таблиці Products та додаткову для розширення структури полем Description. Кожна міграція містить методи Up() для застосування змін та Down() для їх відміни.

LazyCache надає простий та ефективний спосіб реалізації кешування в .NET додатках. Бібліотека забезпечує thread-safe операції, автоматичне управління експірацією кешу та легку інтеграцію з існуючим кодом. Налаштування включало реєстрацію IAppCache як сервісу та впровадження кешування на рівні бізнес-логіки. Реалізовано кешування окремих продуктів з TTL 30 хвилин та списку продуктів з TTL 15 хвилин, з автоматичною інвалідацією при операціях створення, оновлення та видалення.

Для демонстрації роботи створено REST API endpoints, які дозволяють керувати міграціями через веб-інтерфейс, включаючи можливість застосування конкретних версій міграцій та їх відміни. Система логування показує ефективність кешування, відображаючи коли дані завантажуються з бази даних, а коли з кешу. Особливості роботи з SQLite включали реалізацію workaround для видалення колонок через перестворення таблиці, оскільки SQLite не підтримує операцію DROP COLUMN безпосередньо.

Налаштовано GitHub Actions для автоматизованого CI/CD pipeline, який включає збірку проекту, тестування міграцій та публікацію артефактів. Результатом є повнофункціональний демонстраційний проект, який показує практичне використання сучасних підходів до управління базами даних та оптимізації продуктивності через кешування в .NET екосистемі.