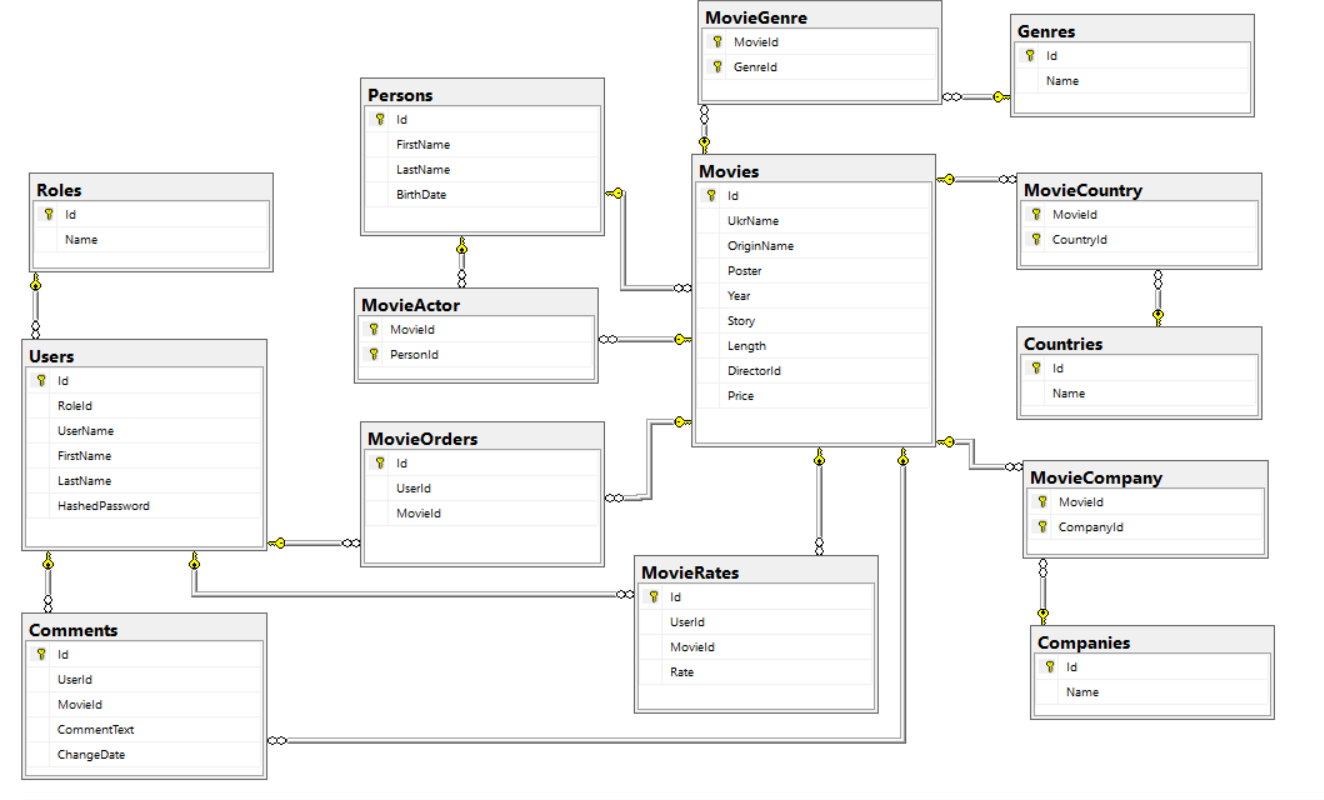
# Storage

СКБД – MS SQL Server.

Діаграма бази даних:



Цілісність бази даних забезпечується за допомогою таких концепцій:

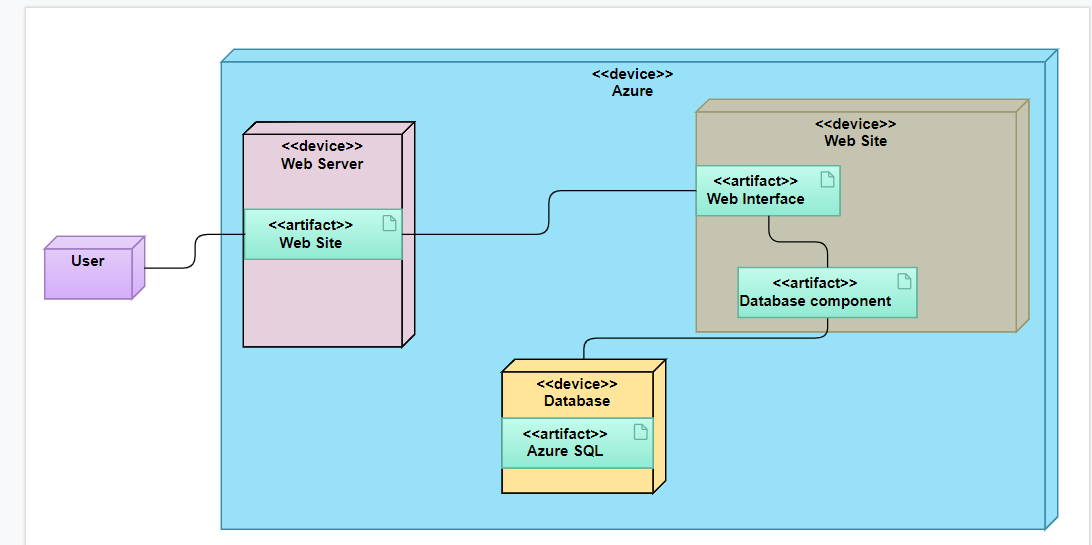
* **Атомарність даних** – кожне поле кожної таблиці може містити нероздільну у даному контексті (атомарну) одиницю інформації.
* **Первинний ключ** (Primary Key) – кожна таблиця, що представляє певну сутність, має первинний ключ – унікальний ідентифікатор (колонка Id). Таблиці, що забезпечують зв’язок багато-до-багатьох, не мають первинного ключа, лише зовнішні.
* **Зовнішній ключ** (Foreign key) – кожна таблиця, що має зв’язок один-до-одного чи один-до-багатьох з іншою таблицею має зовнішній ключ, що вказує на первинний ключ іншої таблиці. Якщо такого зв’язку для певного запису не існує, поле зовнішнього ключа має значення null.
* **Нормалізація** – база даних виконує умови 5НФ.
* **Обмеження** – для збереження коректності моделі даних, на певні колонки таблиць накладені додаткові обмеження (Як, наприклад, NOT NULL на певні зовнішні ключі).

# Resiliency model

# RMA

Azure надає можливість розгортати додаток в хмарних сервісах.

Для підвищення стійкості додатку різні його сервіси можуть знаходитись на різних фізичних серверах. Якщо з одним сервісом щось станеться, інші далі будуть доступні.



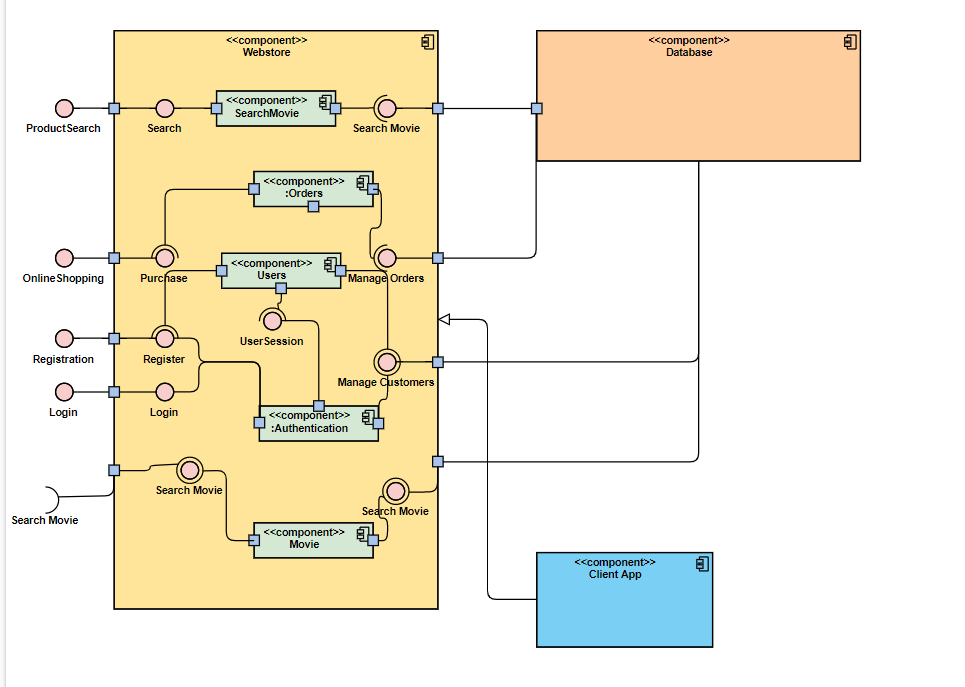
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Component/ Dependency Interaction | Failure Short Name | Failure Description | Response |
| 1 | Web Server -> Web Site | Веб сторіка не можу бети завантажена. | Через помилку на стороні бд. Або на стороні фронт енду чи бек енду. При якому неможливо побачити, що відбувається на сайті. | Дати запит на перезавантажання одного з компонентів. Перезавантажити бд, фронт та бек енд. І повторно дати запит на сайт для його відображення. |
| 2 | Web Site -> Web Interface | Помилка завантаження веб інтерфейсу для відображення сайту | Неможливість завантаження веб інтерфейсу сайту. Фронт енд не відповідає на запити. | Спробувати підвантажити закешовані файли фронт енду, якщо не виходить підвантажити сайт без інтерфейсу. Якщо і це не вийшло перезавантажити фронт енд і повторно дати запит на сайт для його відображення. |
| 3 | Web Site -> Database component | Помилка завантаження бази даних для повного функціонування сайту, а також показ всіх компонентів на сайті. | БД не може підвантажится до веб інтерфейсу і видає порожню сторіку. | Підтягнути сайт, щоб було відомо чи він взагалі працює. Після того спробувати знову дати запит до БД, щоб можна було перезавантажити її. Після цього дати повторний запит для відображення сайту. |
| 4 | Database -> Azure SQL | Неможливість завантаження бази даних. При чому ламається весь бек енд. Та неможливість відображення. | БД. Не може підвантажитсия взагалі, що призводить до помилки, яка не дозволить запустити сайт взагалі. | Дати повторний запит на сервер з БД. Якщо запит не відбувається, то спробувати перезавантажити сервер. Якщо і це не допомогло то дати запит спеціалістам, щоб пофіксили проблему. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Component/ Dependency Interaction | Failure Short Name | Portion Affected | Detection | Resolution | Likelihood | Risk |
| Web Server -> Web Site | Веб сторіка не можу бети завантажена. | Не працює сайте | Кілька секунд | Менше 1 хв | Ймовірно |  |
| Web Site -> Web Interface | Помилка завантаження веб інтерфейсу для відображення сайту | Не працює інтерфейс | Кілька секунд | Менше 1 хв | Ймовірно |  |
| Web Site -> Database component | Помилка завантаження бази даних для повного функціонування сайту, а також показ всіх компонентів на сайті. | Не працює бд. | Кілька секунд | Менше 1 хв 30 сек | Ймовірно |  |
| Database -> Azure SQL | Неможливість завантаження бази даних. При чому ламається весь бек енд. Та неможливість відображення. | Не працює сервер з БД. | До хвилини і більше | Більше 5 хв | Малоймовірно |  |

Система працює таким чином: користувач відкриває у браузері клієнтську частину системи, що хоститься на сервері Azure. Для отримання та відображення відповідних даних клієнтська частина відсилає запити на API, що також хоститься на Azure. У свою чергу, API комунікує з базою даних Azure SQL Database.

# CID

Так виглядає діаграма взаємодії:



# Security

Threat Model Summary:

|  |  |
| --- | --- |
| Not Started | 56 |
| Not Applicable | 0 |
| Needs Investigation | 1 |
| Mitigation Implemented | 1 |
| Total | 58 |
| Total Migrated | 0 |

1. An adversary may gain unauthorized access to Web API due to poor access control checks

2. An adversary can gain access to sensitive information from an API through error messages

3. An adversary can gain access to sensitive data by sniffing traffic to Web API

4. An adversary can gain access to sensitive data stored in Web API's config files

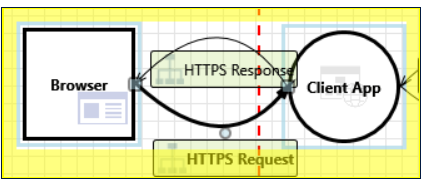
5. Attacker can deny a malicious act on an API leading to repudiation issue

6. An adversary may spoof Client App and gain access to Web API

7. An adversary may inject malicious inputs into an API and affect downstream processes

8. An adversary can gain access to sensitive data by performing SQL injection through Web API

Interaction: HTTPS Request



9. An adversary can gain access to sensitive data stored in Web App's config files

10. An adversary can gain access to sensitive data by performing SQL injection through Web App

11. An attacker steals messages off the network and replays them in order to steal a user's session

12. An adversary can deface the target web application by injecting malicious code or uploading dangerous files

13. An adversary may spoof Browser and gain access to Web Application

14. An adversary can create a fake website and launch phishing attacks

15. Attackers can steal user session cookies due to insecure cookie attributes

Threat model діаграма системи:

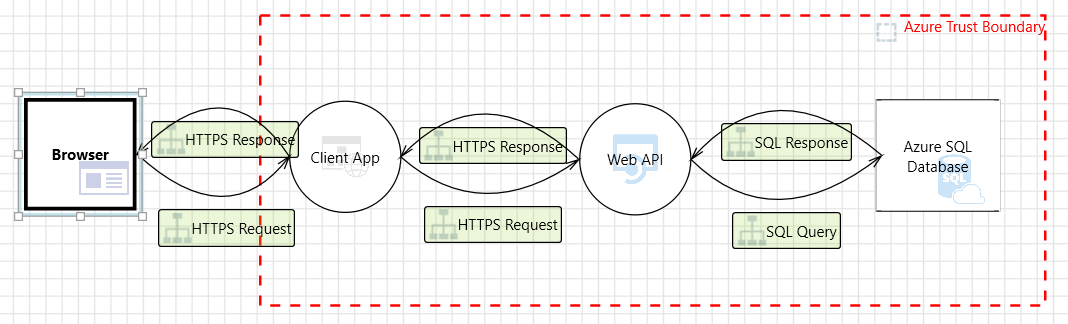


Diagram 1 Diagram Summary:

|  |  |
| --- | --- |
| Not Started | 56 |
| Not Applicable | 0 |
| Needs Investigation | 1 |
| Mitigation Implemented | 1 |
| Total | 58 |
| Total Migrated | 0 |

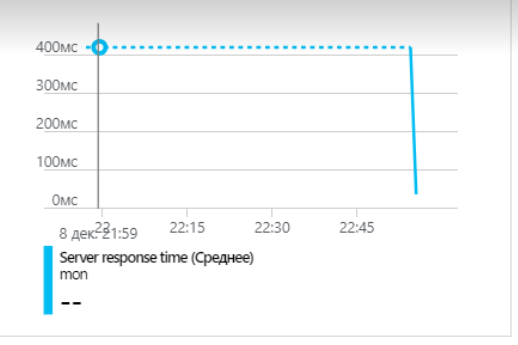
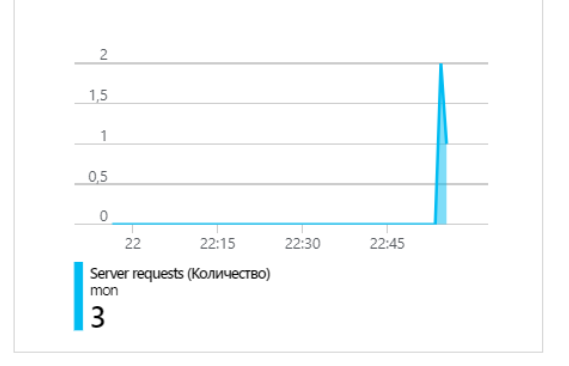
На базі діаграми формується security report.

# Monitoring

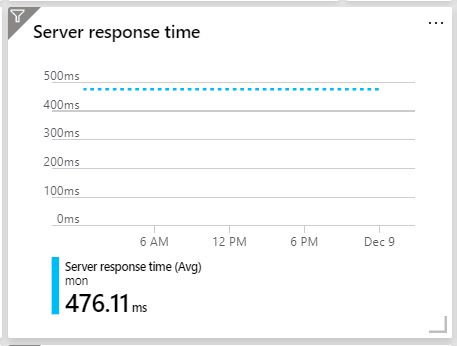
Ми включили у свій проект моніторинг. І ось такі результати ми отримали. При першому підключенні.

Це час відповіді серверу підчас підключення до сайту.

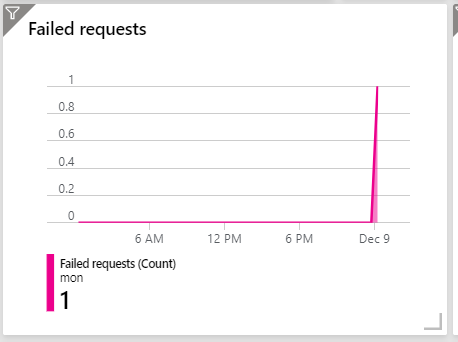
Друге це кількість запитів на сервер. У даний мент вийшло що запитів було 3.



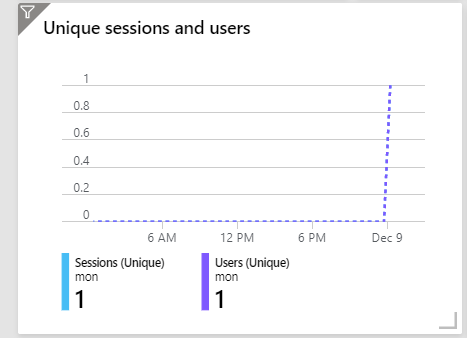
Вже після перезавантаження сайту з усіма фіксами час відклику серверу збільшився.

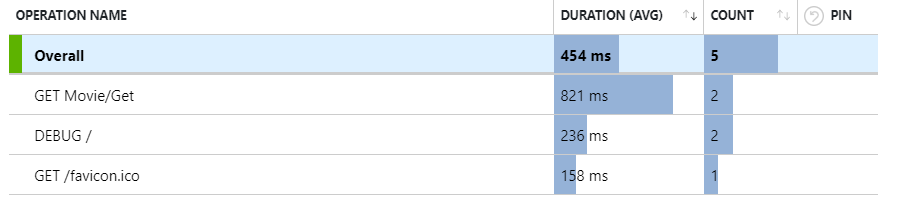


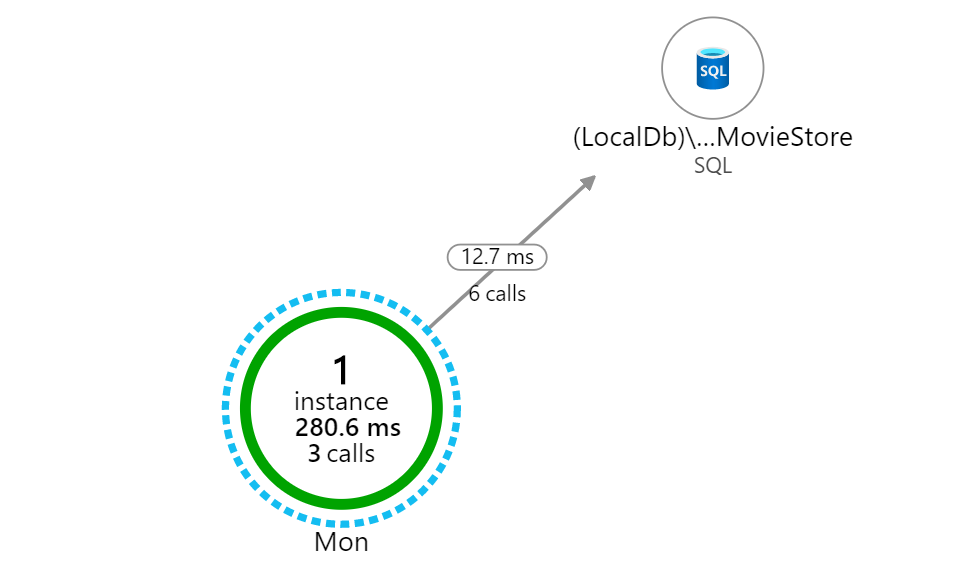
А також стався перший збій, що на діаграмі можна побачити

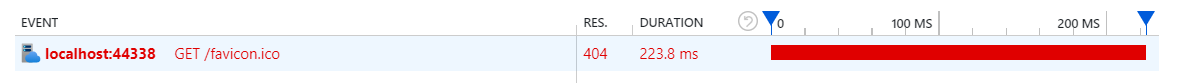


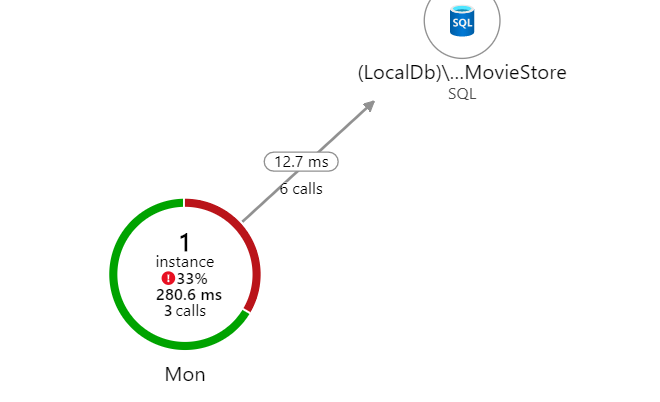
Перевірка унікальних входів.





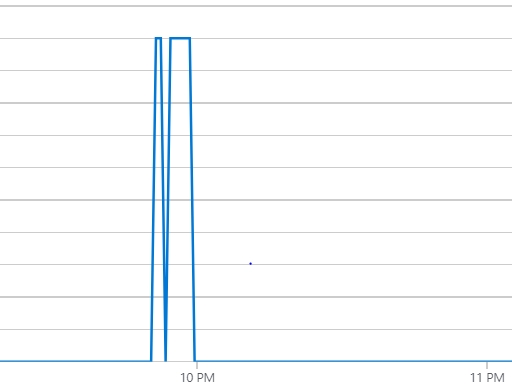




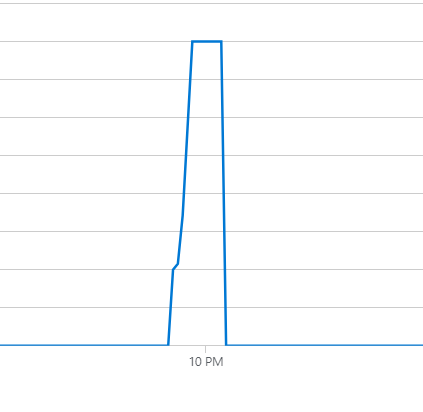


|  |  |  |
| --- | --- | --- |
| № | Помилка | Її вирішення |
| 1 | Помилка бд. | Бекенд частині спробувати здійнити повторний запит і лише тоді дати відповідь клієнтській частині  Повідомлення користувачу перезавантажити сторінку. Повідомлення розробнику на пошту про помилку і негайне її вирішення. |
| 2 | Помилка авторизації | Повідомити користувачу, що неправильність логіну та паролю. |
| 3 | Помилка Front-End | Спробувати відкрити сторінку якщо вона була раніше закешована (функція React JS роботи в офлайн режимі).  Повідомити користувачу перезавантажити сторінку. Відправити запит розробнику про помилку. |
| 4 | Помилка Back-End | Спробувати повторно обробити запит через деякий проміжок часу інакше повідомити користувача та відправити відповідний код про помилку на клієнтську частину. Клієнтській частині спробувати відкрити закешовану відповідь даного запиту.  Відправити запит розробнику на телефон(пошту) про помилку і негайне її виправлення. |
| 5 | Помилка аунтентифікації. | Повідомити користувачу про помилку автентифікації попросити перехавантаити сторінку та перелогуватися.  Повідомити про помилку і неможливість зареєстрованого користувача зайти розробнику на пошту чи телефон. |

# Telemetry



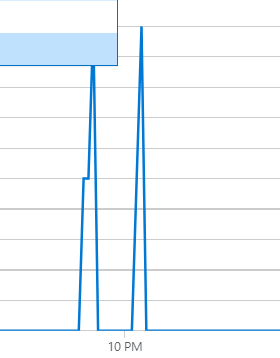
Це у нас метрика яка відповідає за те чи програма в даний момент ввімкнена і працює чи ні. Як ми бачимо 2 різні підйоми. Це означає що в проміжок в 1 хв сайт був 1 раз перезапущений. Ми можемо дивитися в момент роботи сайту. Але з періодичністю змін 1 хв.



Також ми можемо побачити кількість запитів та повідомлень під час запуску нашого сайту та звернення до цього серверу.



Тут показано самі аргументи (1 штука) який під’єднаний таким чином що під час запуску проект на сервер Ажуру подається сигнал, що такий то аргумент почав працювати.

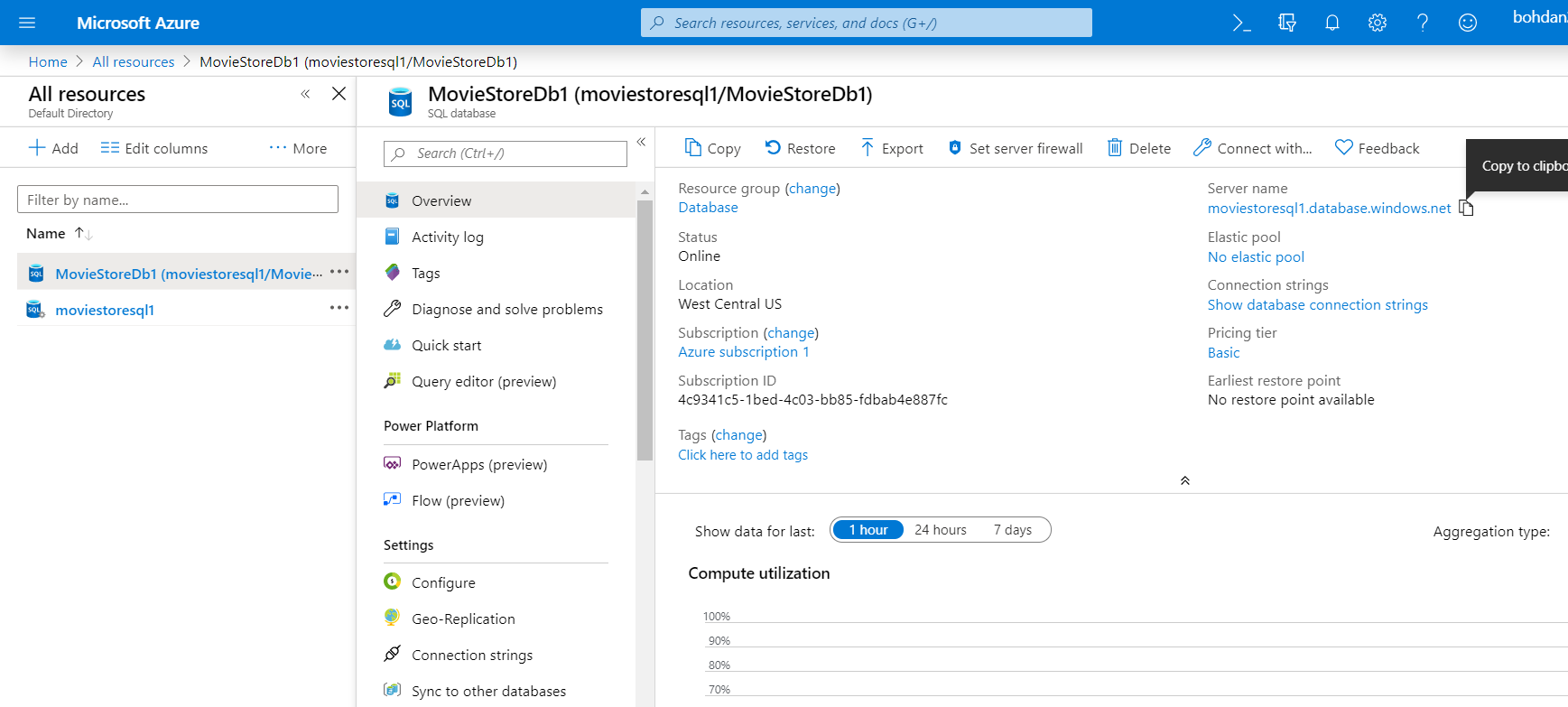


Тут показана сума всіх телеметричних запитів. Тут їх від 1 до 3.

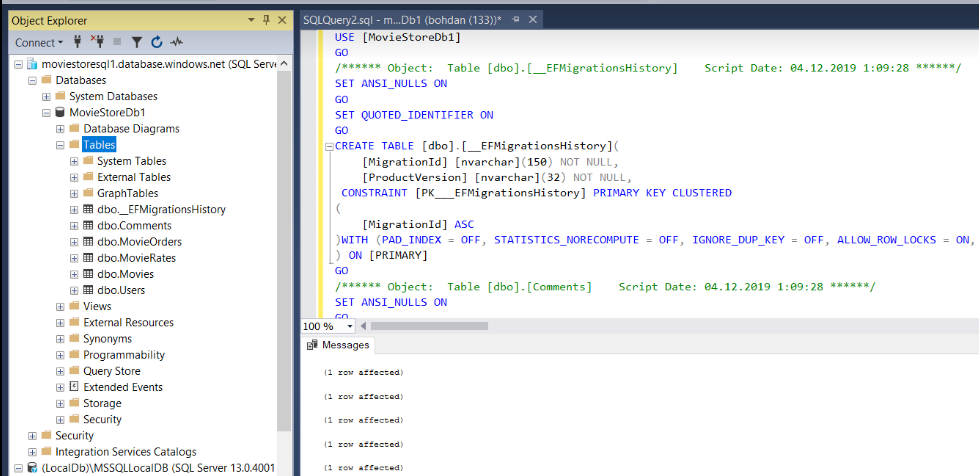
|  |  |  |
| --- | --- | --- |
| № | Помилка | Її вирішення |
| 1 | Помилка запиту до серверу. | Спробувати зробити, ще один запит. Якщо не виходить то повідомлення розробнику на пошту про помилку і негайне її вирішення. |
| 2 | Помилка запиту до сайту. | Повідомлення розробнику на пошту про помилку і негайне її вирішення. |
| 3 | Помилка запиту до БД | Спробувати повторно дати запит до Бд. Якщо не виходить повідомити про помилку адміністратору БД на пошту чи телефон. |
| 4 | Падіння сайту | Спробувати реконектнути користувача, та перезавантажити Front + Back енди. Якщо не вийде то повідомити про помилку розробнику про помилку на пошту чи телефон. |
| 5 |  |  |

# Deployment

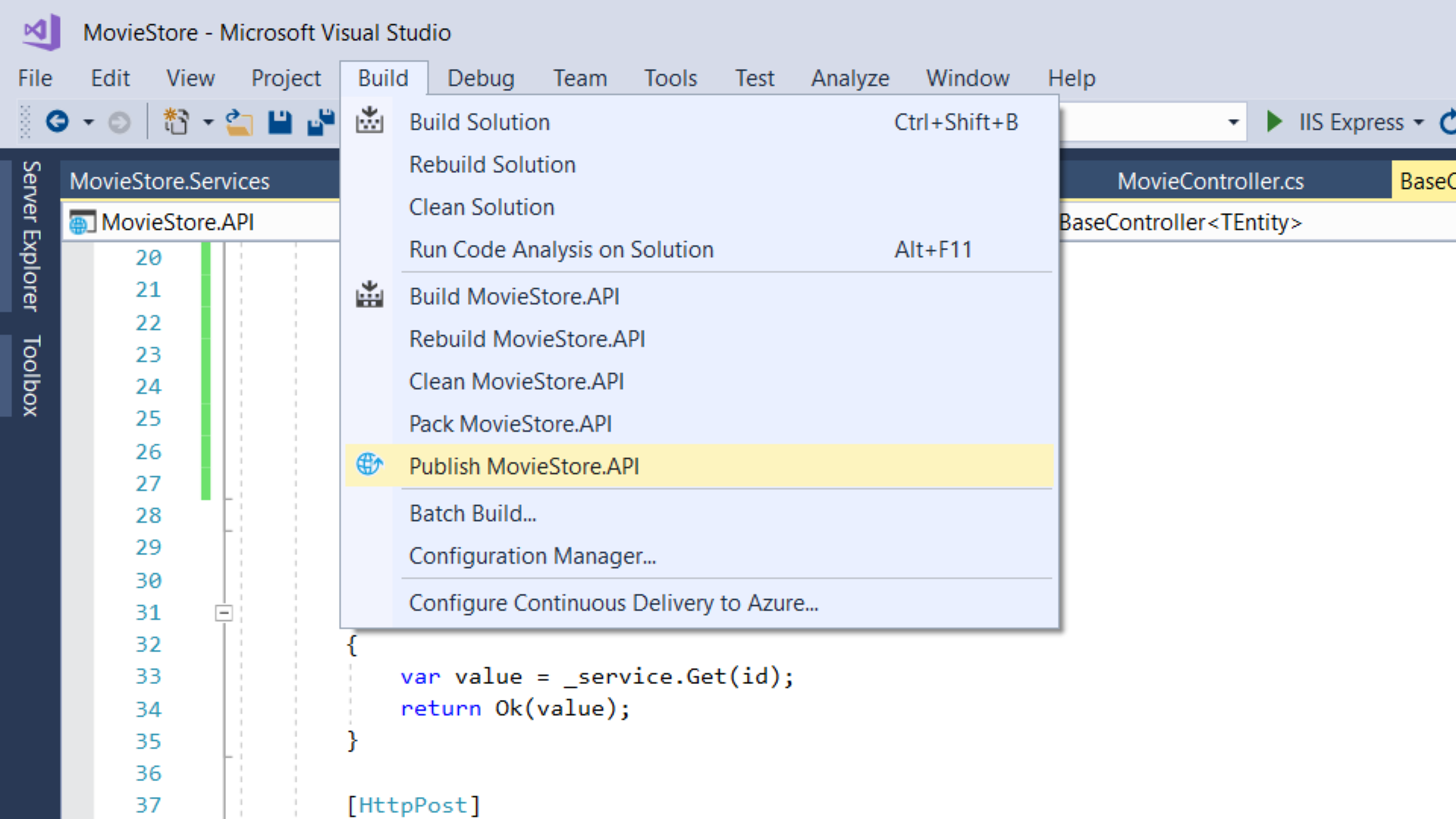
Для деплойменту проекту було використано сервіс **Microsoft Azure**. Відповідно здійснено реєстрацію на сайті <https://azure.microsoft.com/> після чого було переадресовано на портал в якому здійснюється керування сервісами, базами даних та іншими ресурсами. Спочатку було створено базу даних, для цього створено відповідну групу **DataBase** з використанням функції **Create New Group.**

Після чого скориставшись функцією **Create New Resource** вибравши тип **SQL Server Database** і задавши відповідну назву, користувача та пароль створюється база даних. 

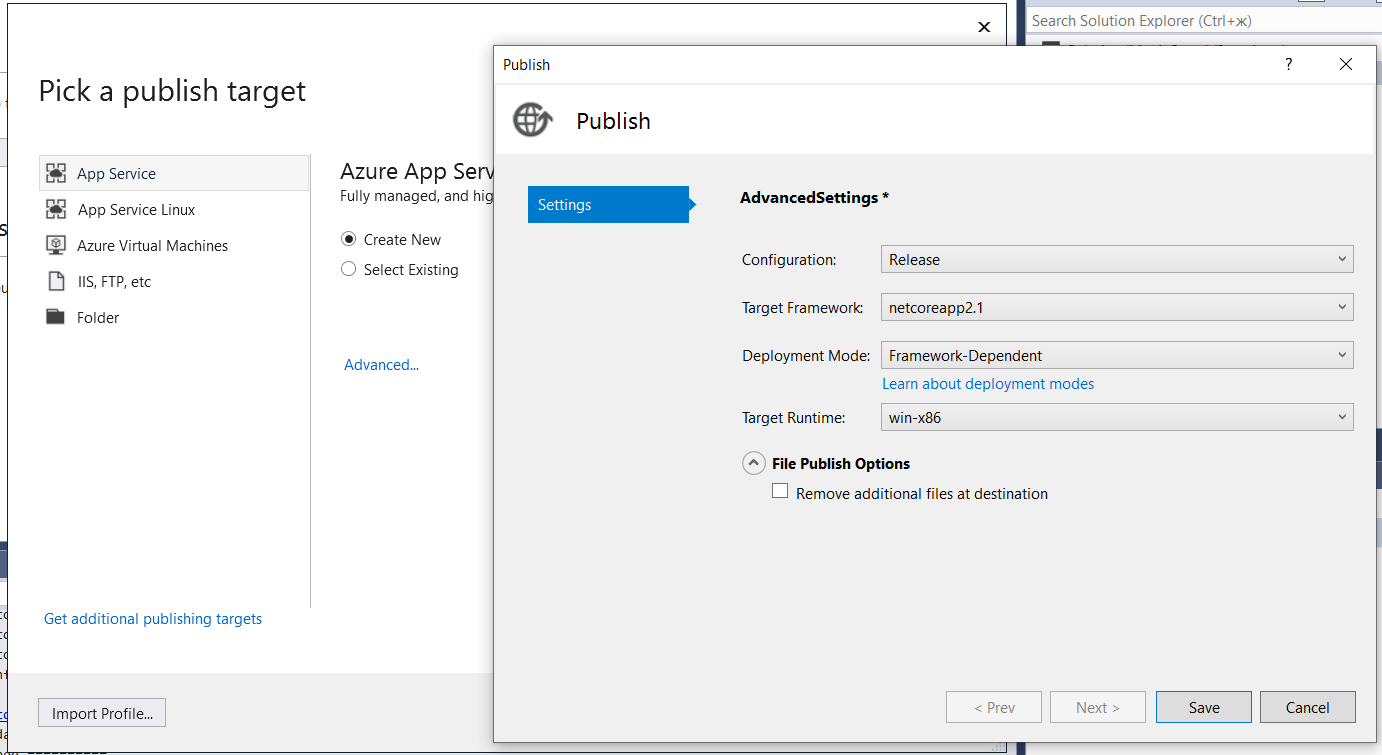
Легко можна побачити, що база даних знаходиться за адресою **moviestoresql1.database.windows.net** відповідно можна доступитися до неї з використанням **Microsoft SQL Server Management Studio**, відповідно ствоивший дамп нашої бази можна задеплоїти БД на відповідний сервер.



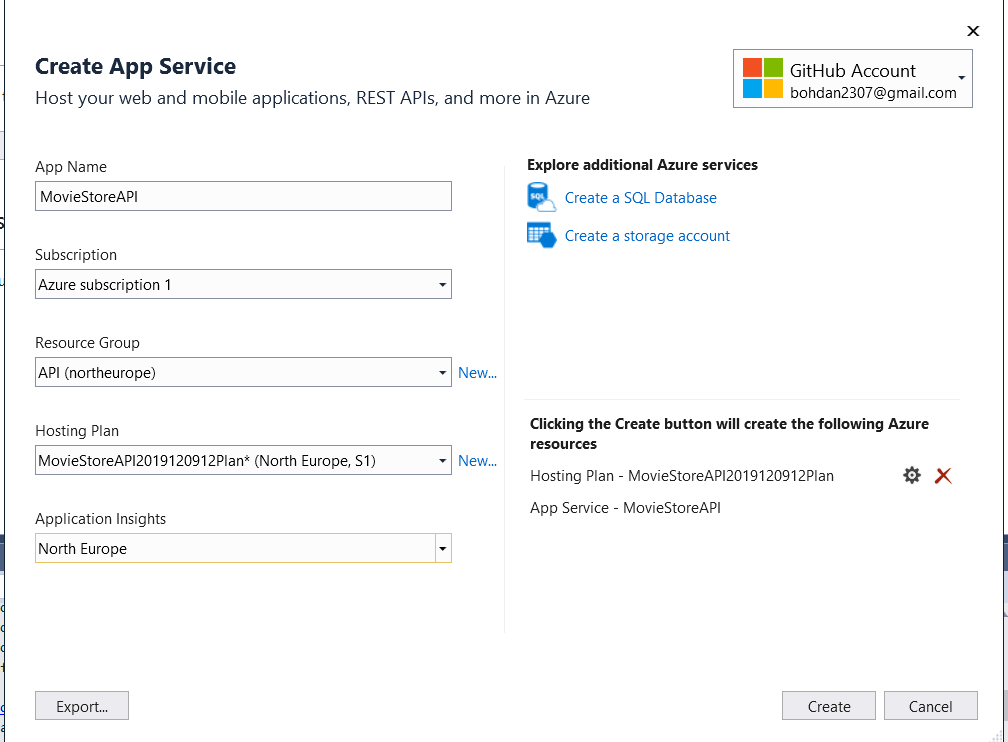
Наступним кроком є деплоймент **Web API** рівня для цього також створимо відповідну групу, а сам процес деплойменту виконаємо з використанням **Microsoft Visual Studio.** Спочатку модифікуємо конфігурацію для використання уже готової БД**.** Далі в меню **Build** вибираємо опцію **Publish.**



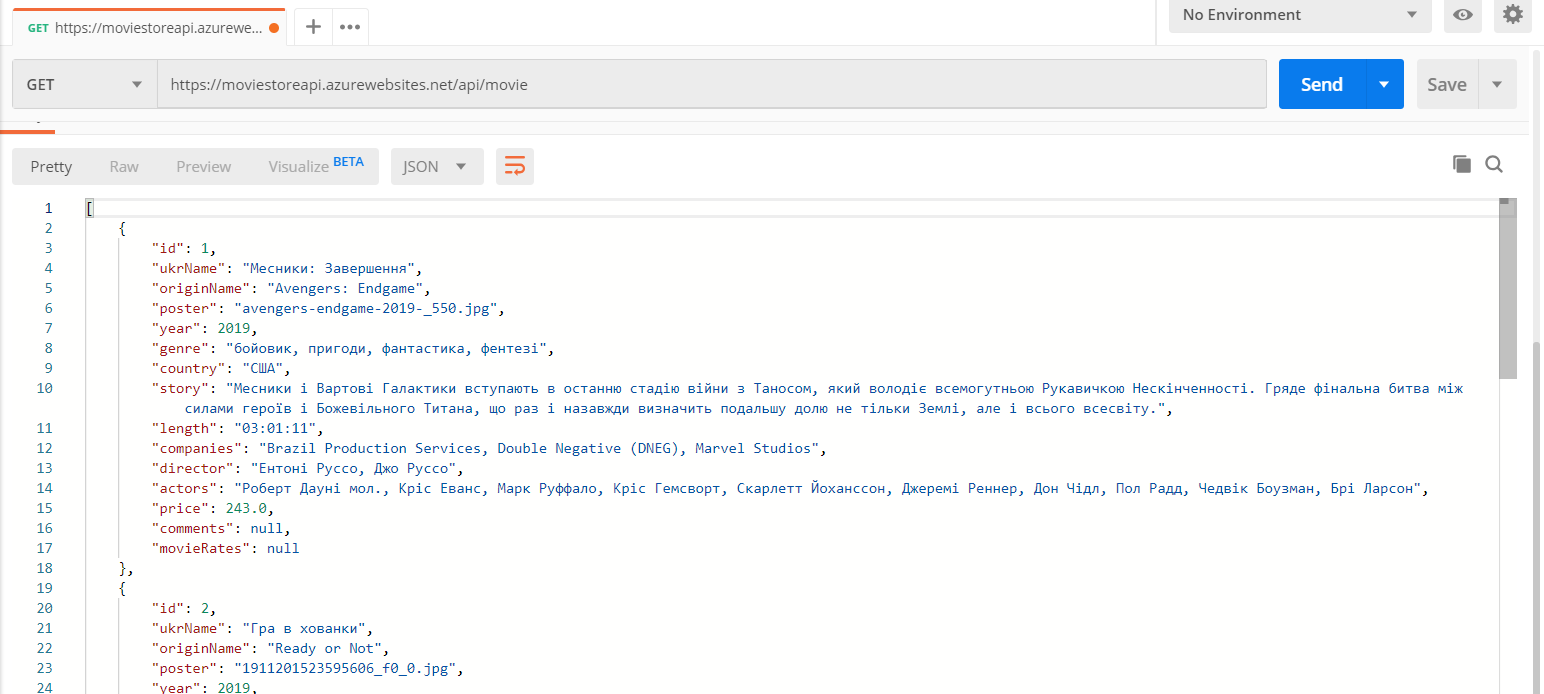
Після цього заходимо у свій **Azure** профіль, вибираємо відповідні налаштування і зберігаємо.



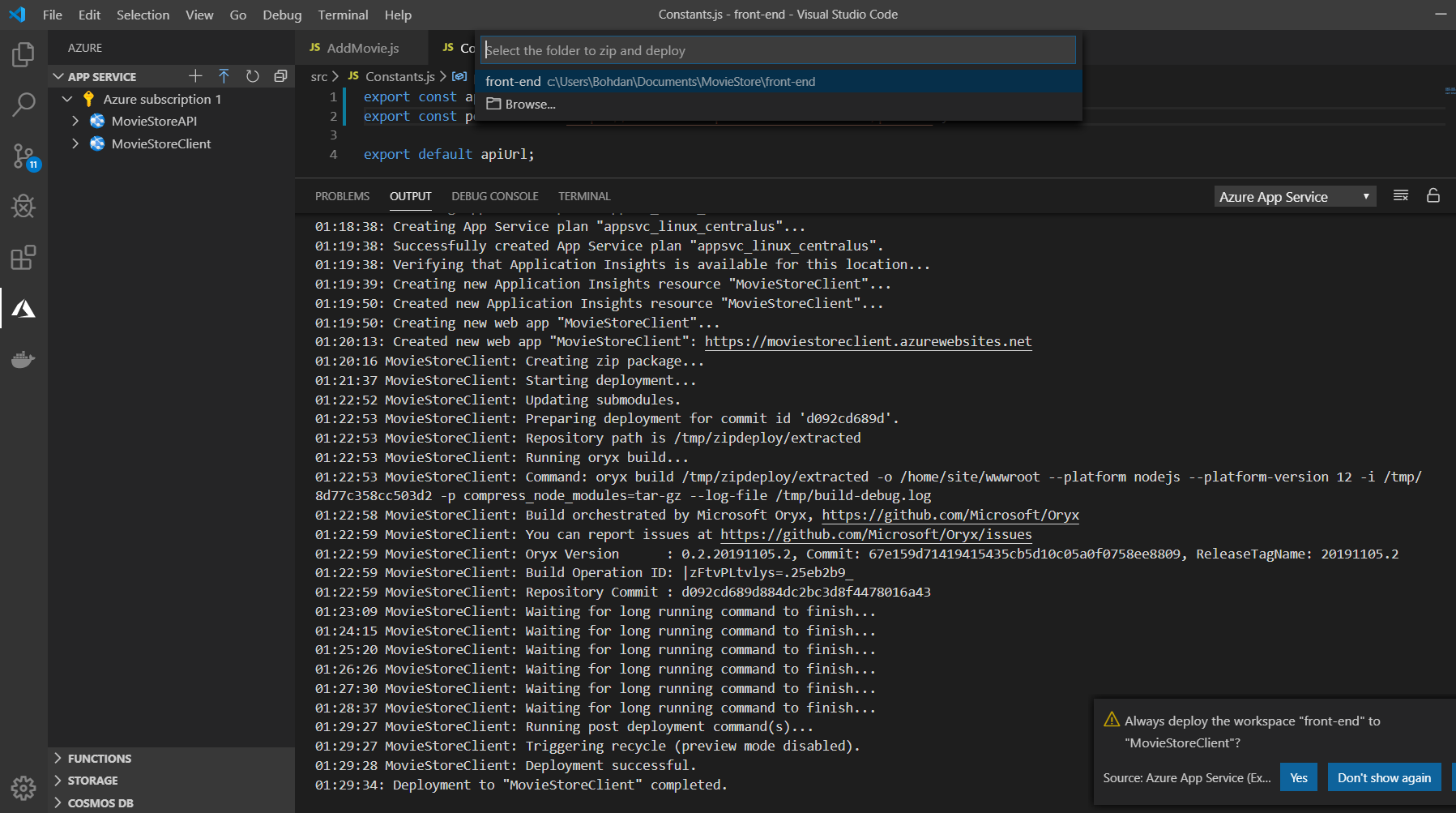
Вводимо назву аплікації, вибираємо групу та сервер на який бажаємо здійснити деплоймент.



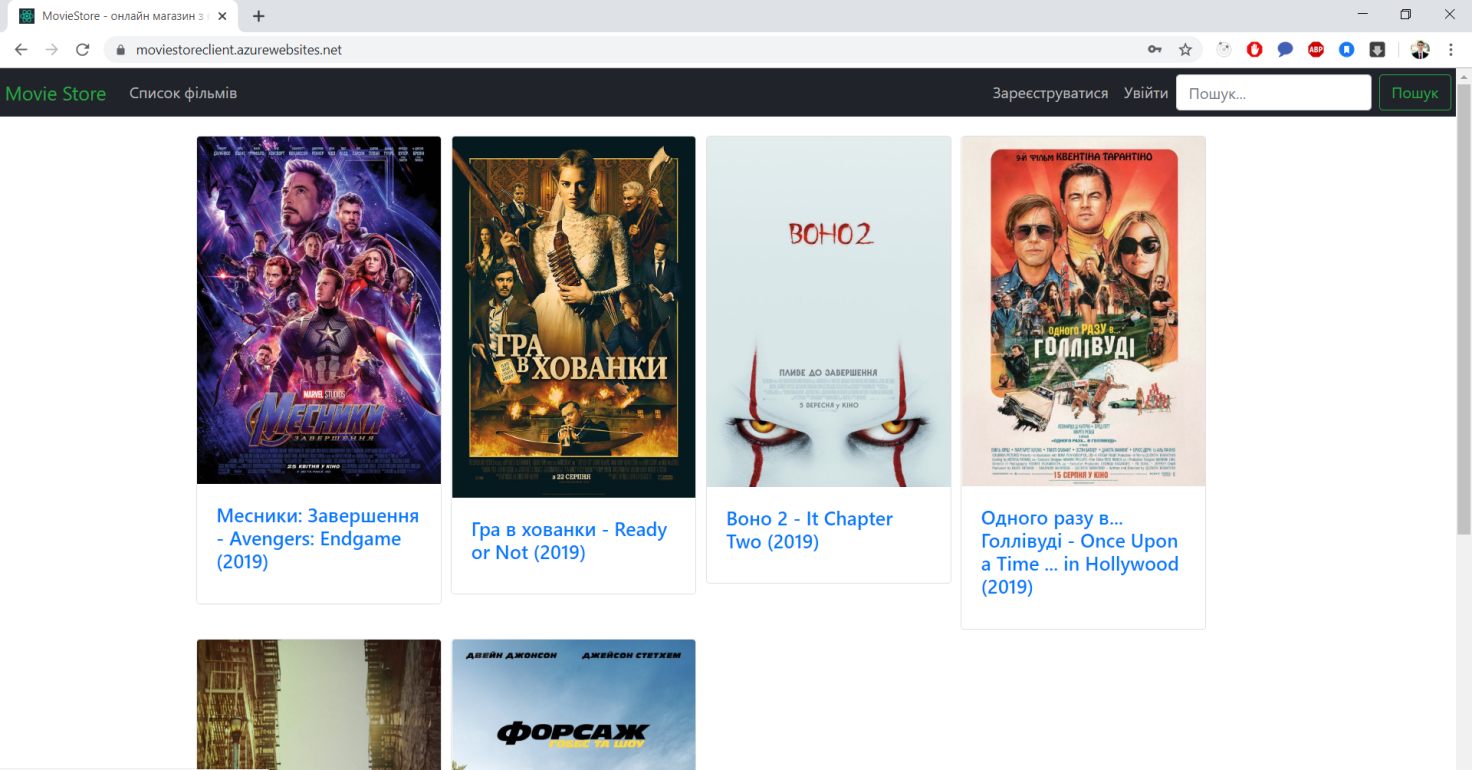
Після чого нажимаємо **Create** і чекаємо доки проект збілдиться і задеплоїться на відповідний сервіс. Посилання за яким він буде доступний – це <https://moviestoreapi.azurewebsites.net/> яке відкриється одразу по завершенню процесу деплойменту. Для перевірки правильності роботи сайту скористаємось програмою **Postman**. Як можна бачити, на відповідний ***GET*** запит **API** повернуло відповідь, одже зв’язка **API** рівня з базою даних працює правильно і все добре сконфігуровано.



Для деплойменту клієнтської частини аналогічним чином використано **Microsoft Visual Studio Code** з встановленими плагінами для роботи з **Azure** сервісами. Спочатку модифікується конфігурацію для зв’язку з уже задеплоїним API рівнем. Далі задається назва проекту, створюється новий проект і відбувається деплоймент.



Як бачимо все успішно завершено і сайт знаходиться за посиланням <https://moviestoreclient.azurewebsites.net/> для перевірки робочості якого скористаємося браузером.



Як бачимо клієнтська частина працює правильно, отримує відповідь від APІ рівня та підтягує з нього картинки. Отже усі рівні сайту працюють правильно і деплоймент успішний.