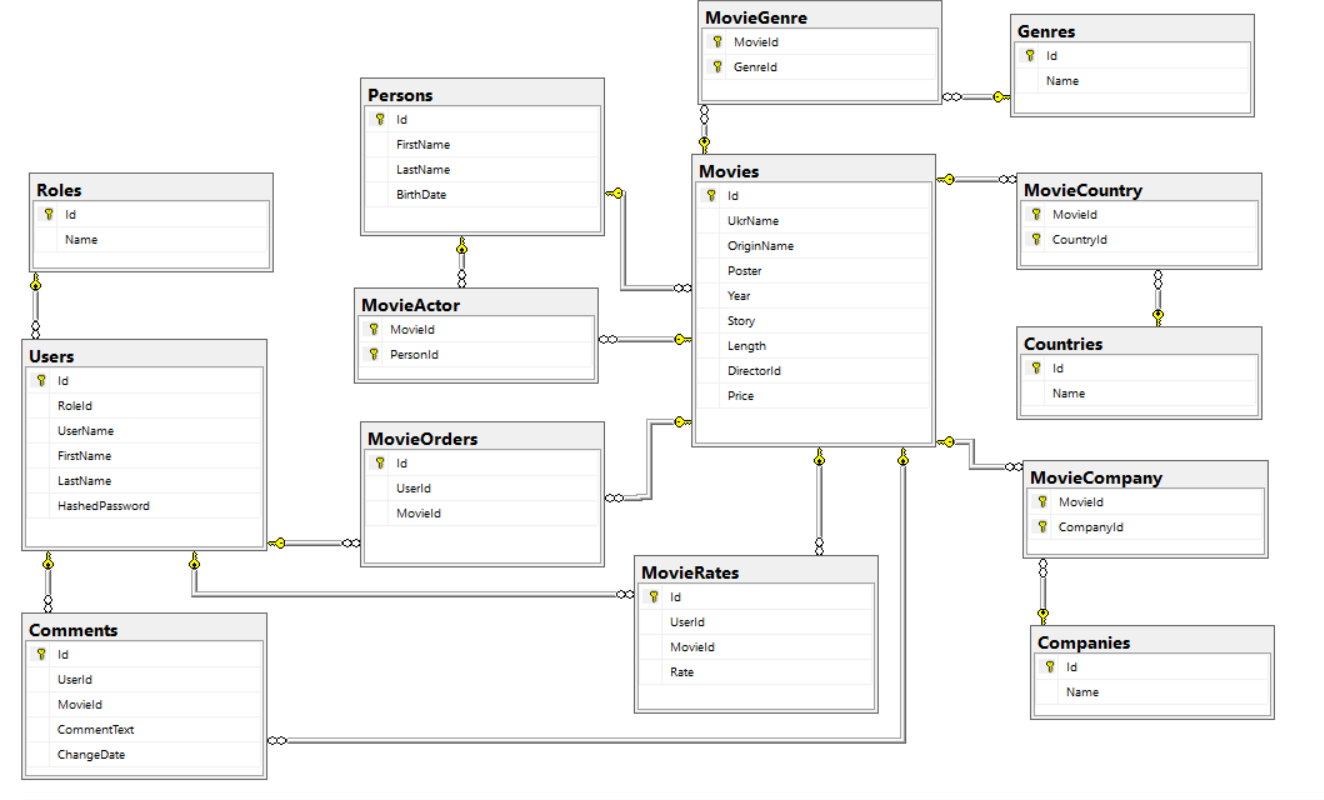
# Storage

СКБД – MS SQL Server.

Діаграма бази даних:



Цілісність бази даних забезпечується за допомогою таких концепцій:

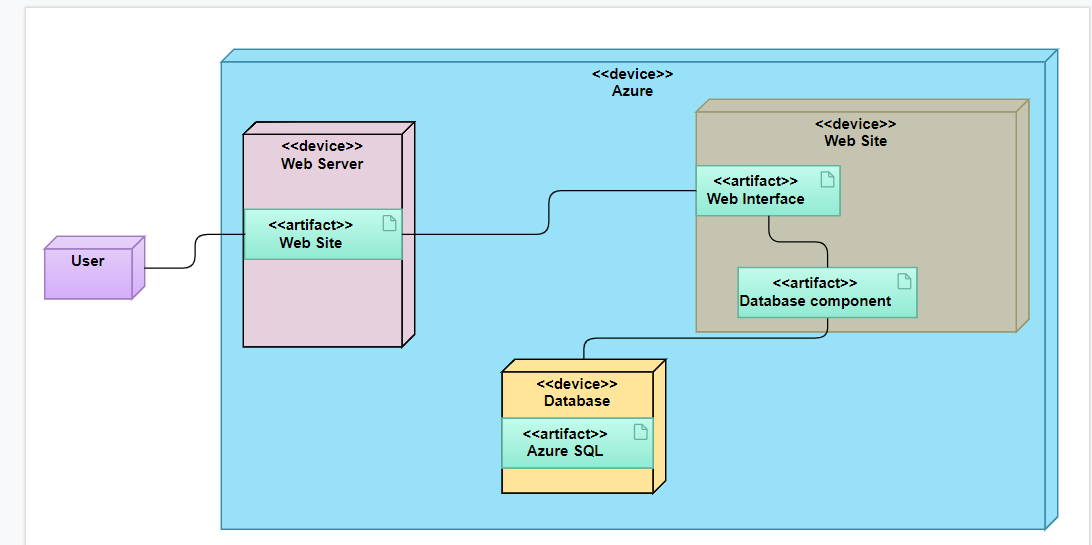
* **Атомарність даних** – кожне поле кожної таблиці може містити нероздільну у даному контексті (атомарну) одиницю інформації.
* **Первинний ключ** (Primary Key) – кожна таблиця, що представляє певну сутність, має первинний ключ – унікальний ідентифікатор (колонка Id). Таблиці, що забезпечують зв’язок 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 | | Interaction Dependency |
| 1 | | Frontend app -> API | Make GET/Post/ PATCH/PUT requests to API. |
| 2 | | API -> DB | CRUD operation |
| 3 | | API -> Cloudinary | Upload images |
|  | |  |  |

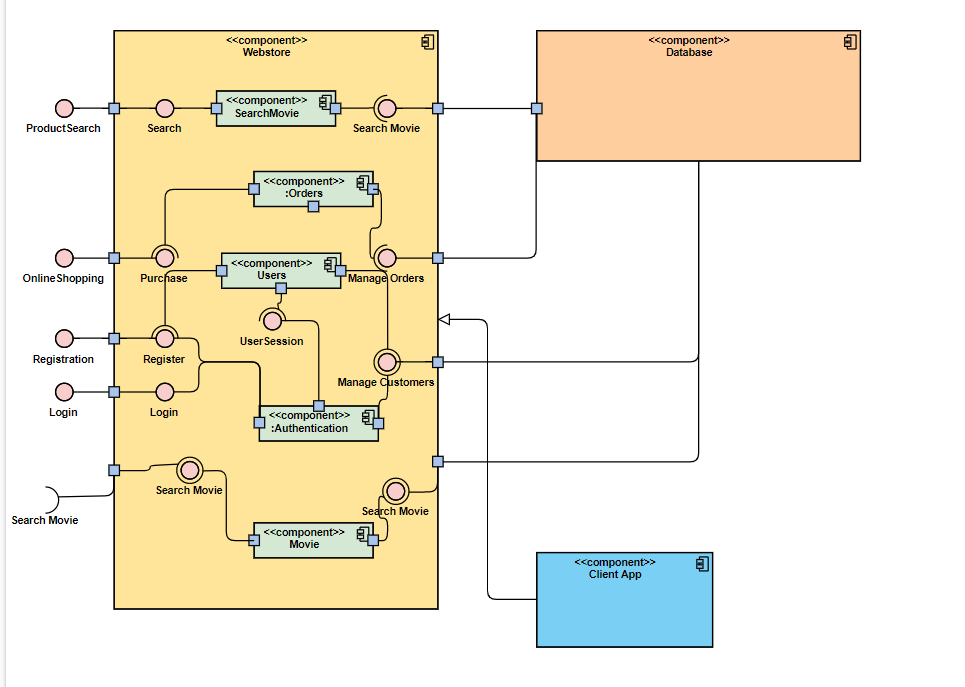
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Component/ Dependency Interaction | Failure Short Name | Failure Description | Response |
| 1 | APP -> DB | Помилка БД | БД не можу бути підтягнута, або відсутні дані у БД | Дати запит на перезавантаження БД. |
| 2 | APP-> Front-End | Помилка Фронт енду | Фронт енд не можу бути підтягнутий, або відображений. | Підтягнути закешовані дані, або спробувати повторно дати запит до фронта. |
| 3 | APP -> Back-End | Помилка Бек енду | Бек енд не можу бути запущений, у зв’язку з якоюсь помилкою. | Повторно дати запит до бек енду. Спробувати його перезавантажити. |
| 4 |  |  |  |  |
| 5 |  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Component/ Dependency Interaction | Failure Short Name | Portion Affected | Detection | Resolution | Likelihood | Risk |
| APP -> DB | Помилка БД |  |  |  |  |  |
| APP-> Front-End | Помилка Фронт енду |  |  |  |  |  |
| APP -> Back-End | Помилка Бек енду |  |  |  |  |  |
|  |  |  |  |  |  |  |

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.

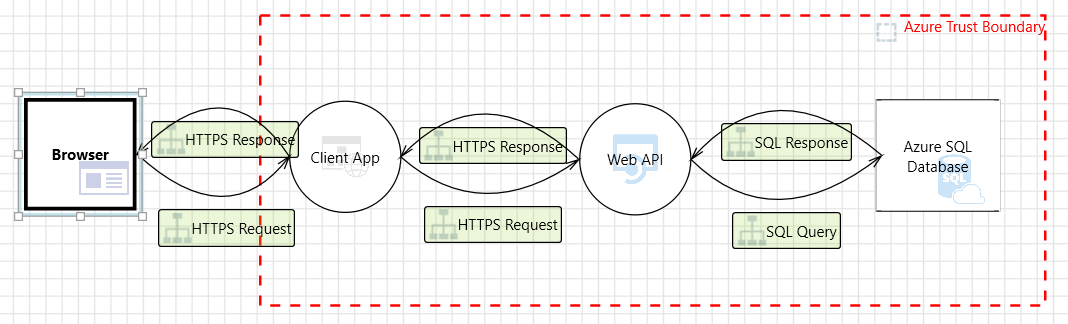
# CID

Так виглядає діаграма взаємодії:



# Security

Threat model діаграма системи:



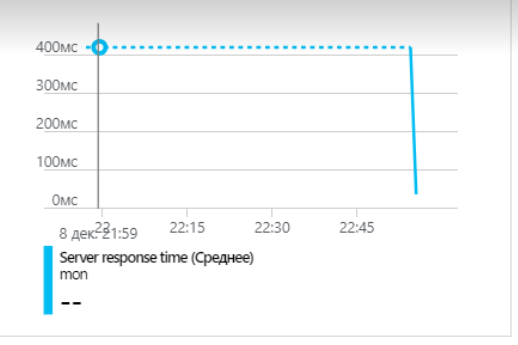
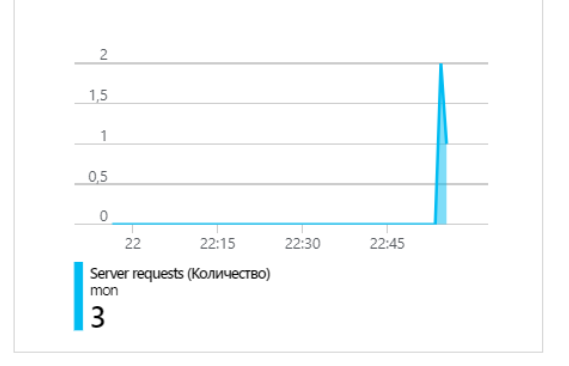
На базі діаграми формується security report.

# Monitoring

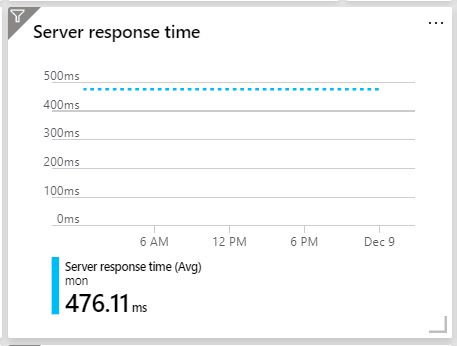
Ми включили у свій проект моніторинг. І ось такі результати ми отримали. При першому підключенні.

Це час відповіді серверу підчас підключення до сайту.

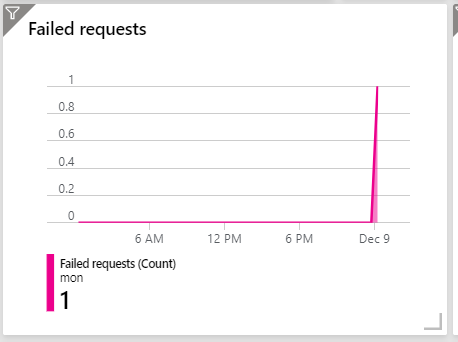
Друге це кількість запитів на сервер. У даний мент вийшло що запитів було 3.



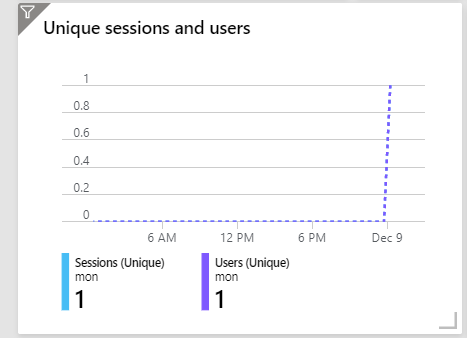
Вже після перезавантаження сайту з усіма фіксами час відклику серверу збільшився.

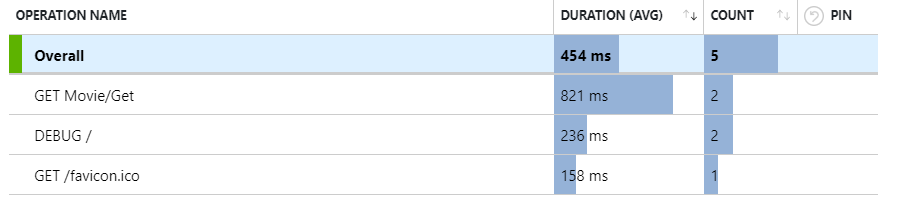


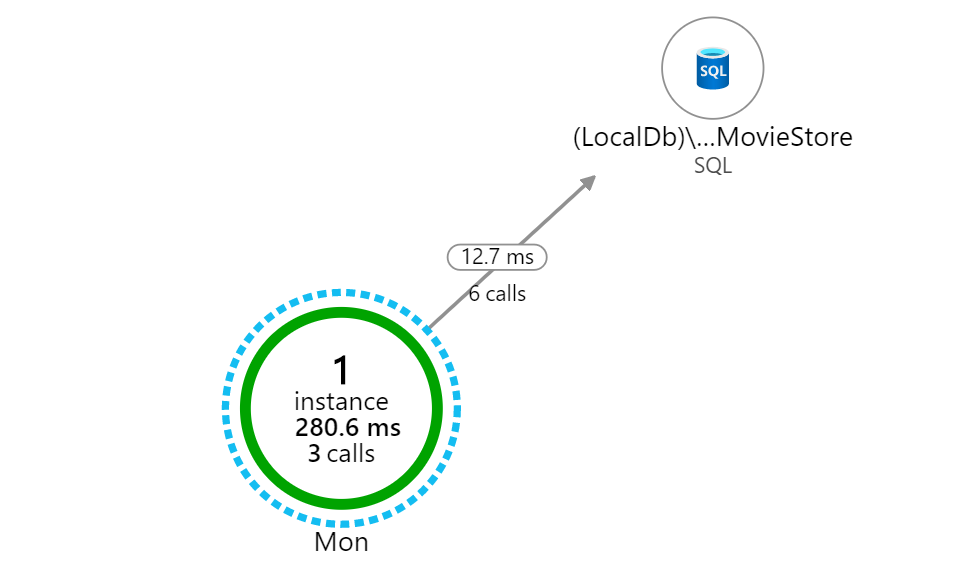
А також стався перший збій, що на діаграмі можна побачити

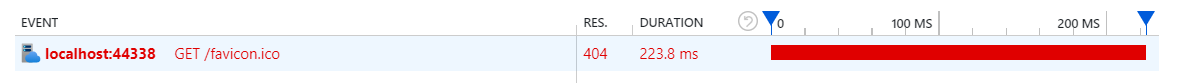


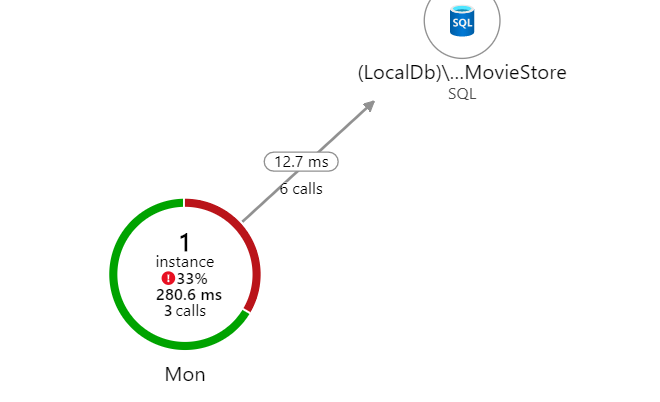
Перевірка унікальних входів.





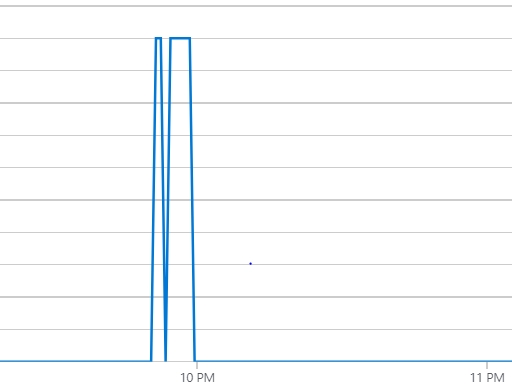




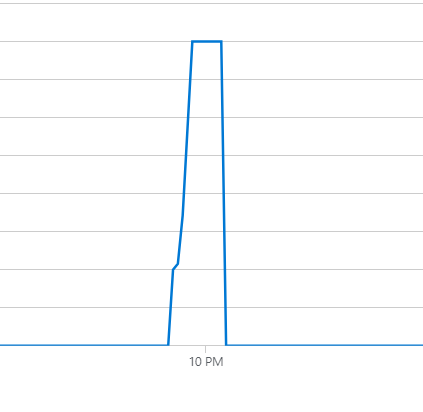


|  |  |  |
| --- | --- | --- |
| № | Помилка | Її вирішення |
| 1 | Помилка бд. | Бекенд частині спробувати здійнити повторний запит і лише тоді дати відповідь клієнтській частині  Повідомлення користувачу перезавантажити сторінку. Повідомлення розробнику на пошту про помилку і негайне її вирішення. |
| 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



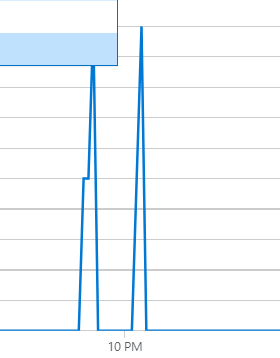
Це у нас метрика яка відповідає за те чи програма в даний момент ввімкнена і працює чи ні. Як ми бачимо 2 різні підйоми. Це означає що в проміжок в 1 хв сайт був 1 раз перезапущений. Ми можемо дивитися в момент роботи сайту. Але з періодичністю змін 1 хв.



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.

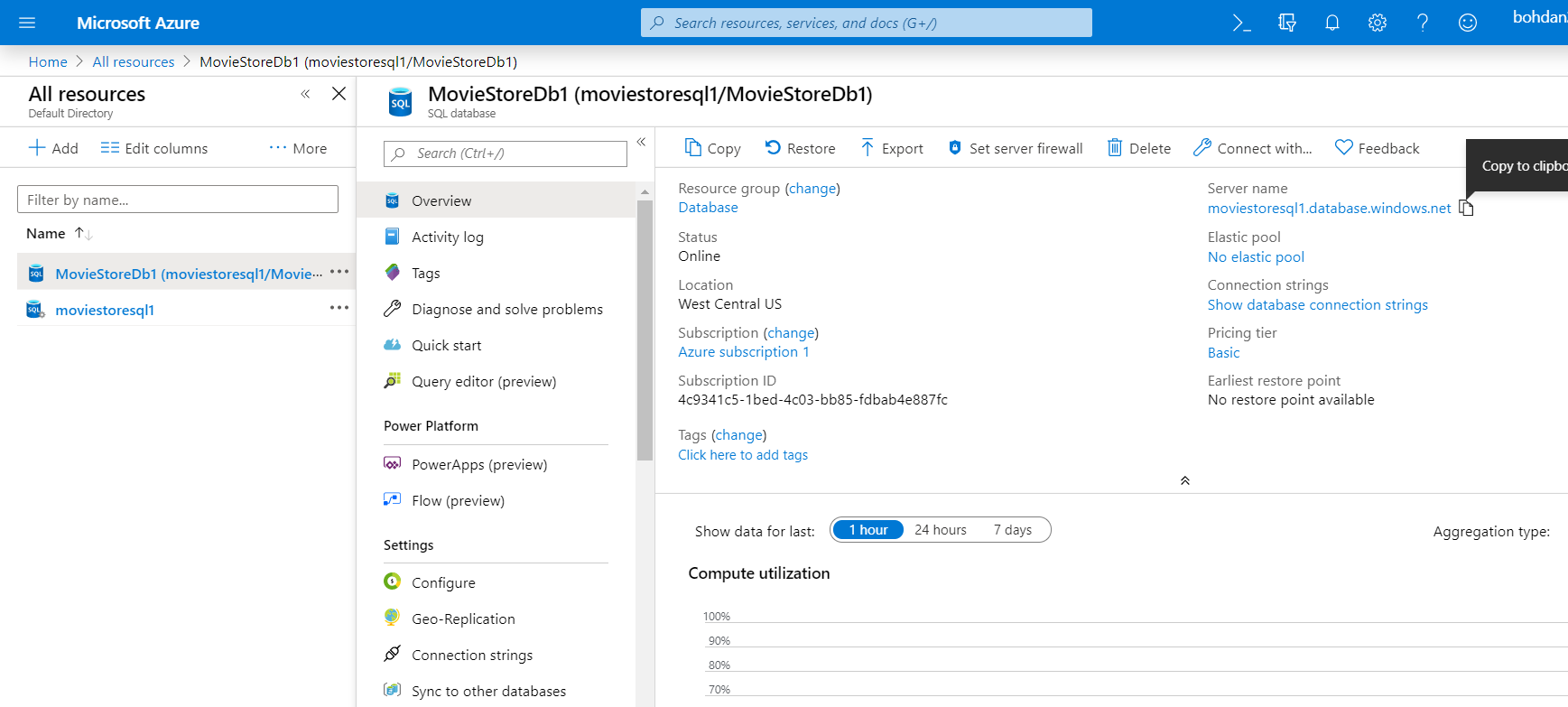


Тут показана сума всіх телеметричних запитів. Тут їх від 1 до 3.

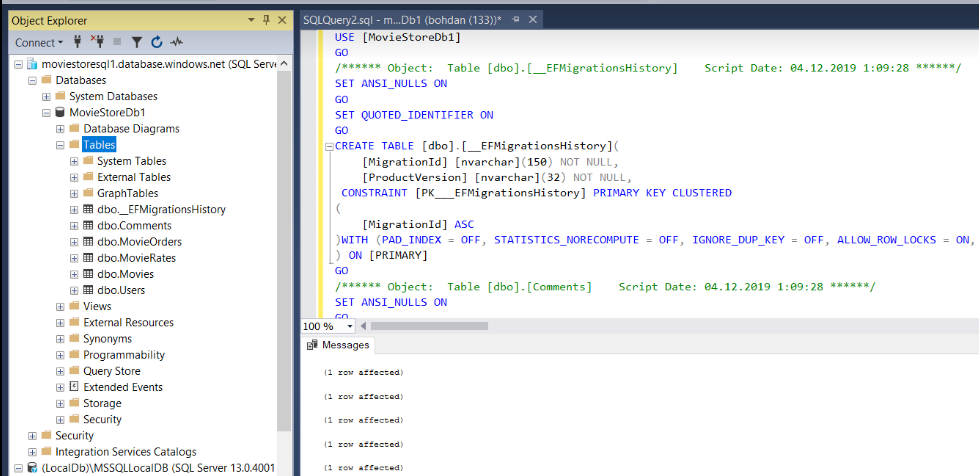
|  |  |  |
| --- | --- | --- |
| № | Помилка | Її вирішення |
| 1 | Помилка запиту до серверу. | Спробувати зробити, ще один запит. Якщо не виходить то повідомлення розробнику на пошту про помилку і негайне її вирішення. |
| 2 | Помилка запиту до сайту. | Повідомлення розробнику на пошту про помилку 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

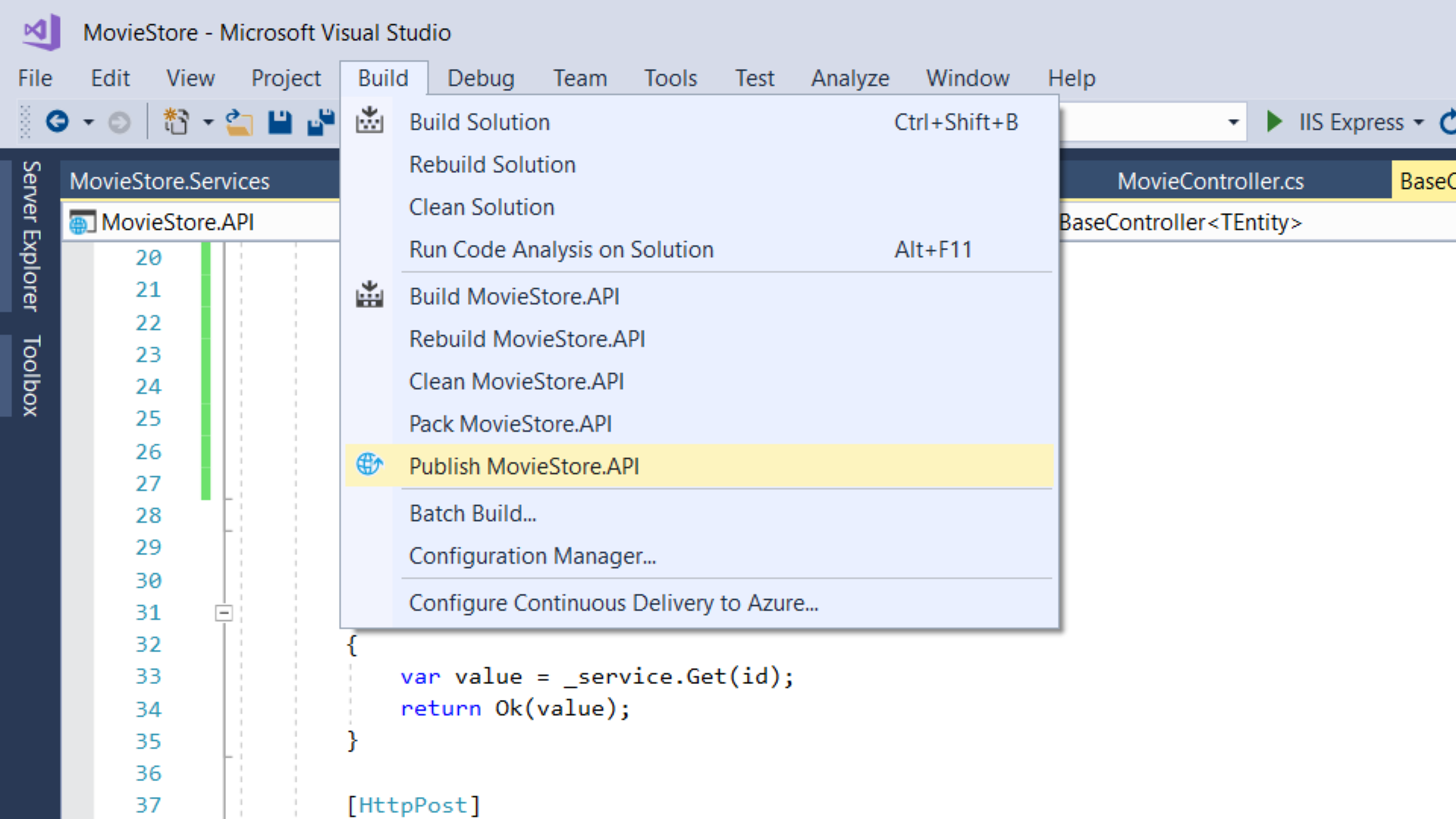
Для деплойменту проекту було використано сервіс **Microsoft Azure**. Відповідно здійснено реєстрацію на сайті <https://azure.microsoft.com/> після чого було переадресовано на портал в якому здійснюється керування сервісами, базами даних та іншими ресурсами. Спочатку було створено базу даних, для цього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

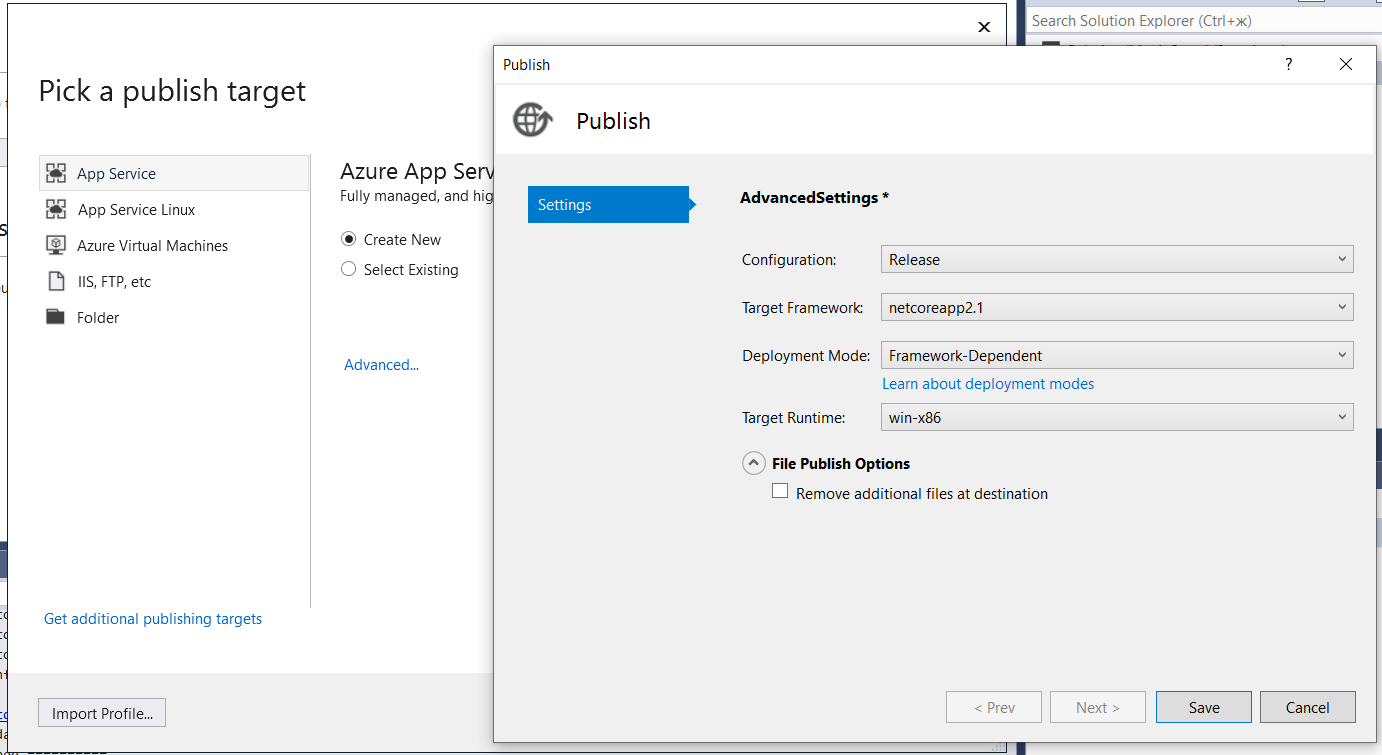
Легко можна побачити, що база даних знаходиться за адресою **moviestoresql1.database.windows.net** відповідно можна доступитися до неї з використанням **Microsoft SQL Server Management Studio**, відповідно ствоивший дамп нашої бази можна задеплоїти БД на відповідний сервер.



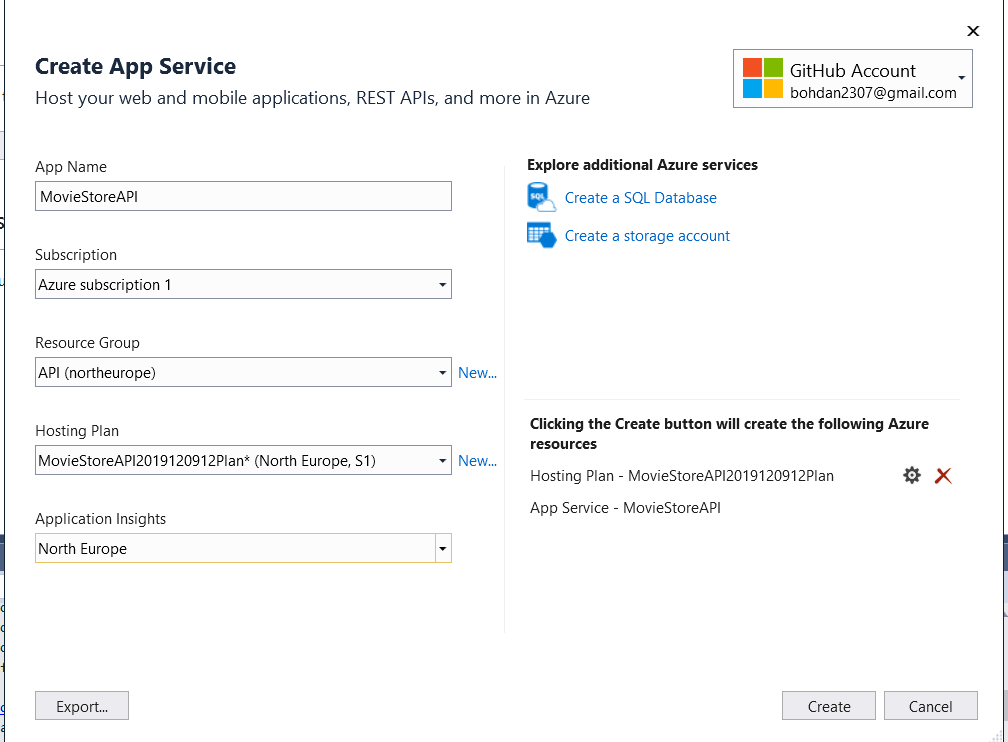
Наступним кроком є деплоймент **Web API** рівня для цього також створимо відповідну групу, а сам процес деплойменту виконаємо з використанням **Microsoft Visual Studio.** Спочатку модифікуємо конфігурацію для використання уже готової БД**.** Далі в меню **Build** вибираємо опцію **Publish.**



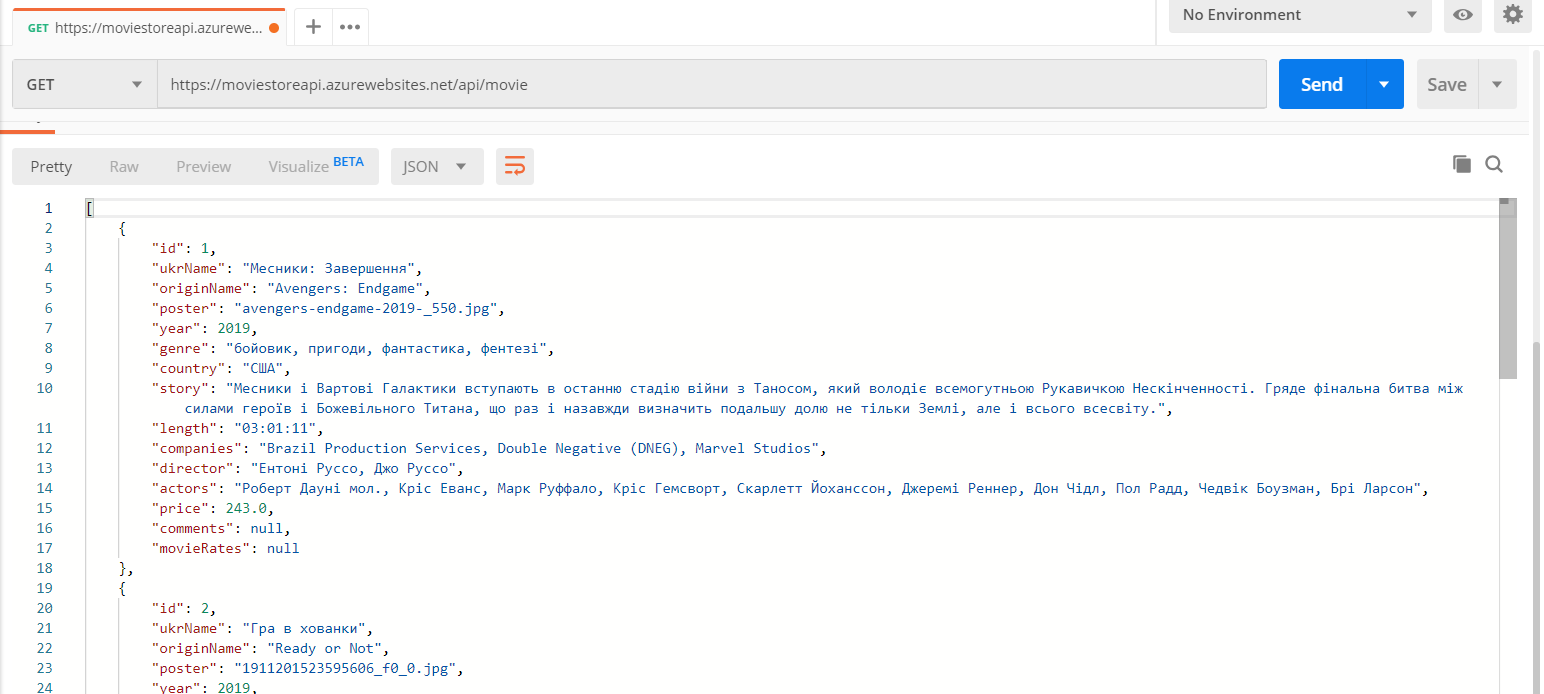
Після цього заходимо у свій **Azure** профіль, вибираємо відповідні налаштування і зберігаємо.



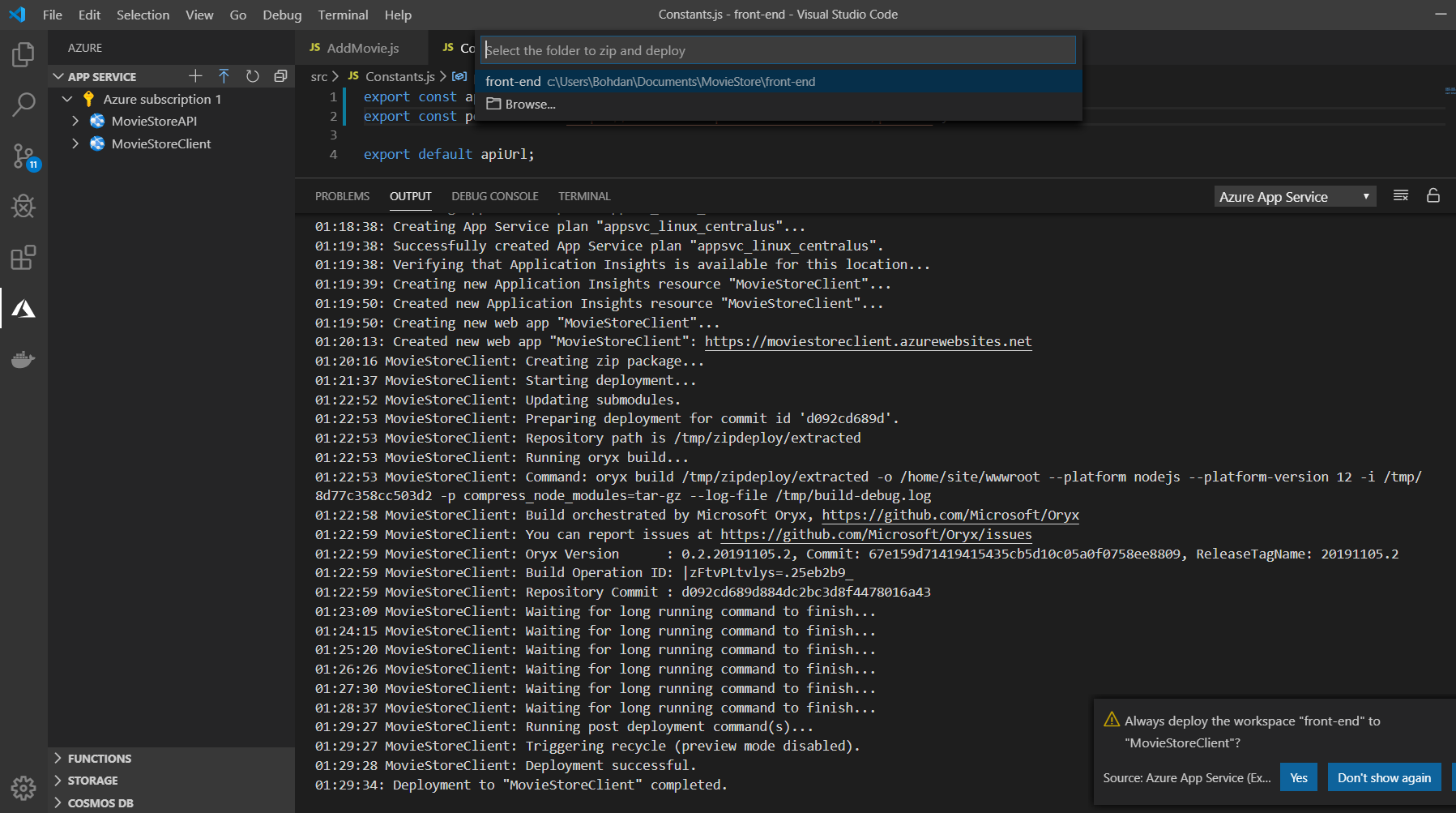
Вводимо назву аплікації, вибираємо групу та сервер на який бажаємо здійснити деплоймент.



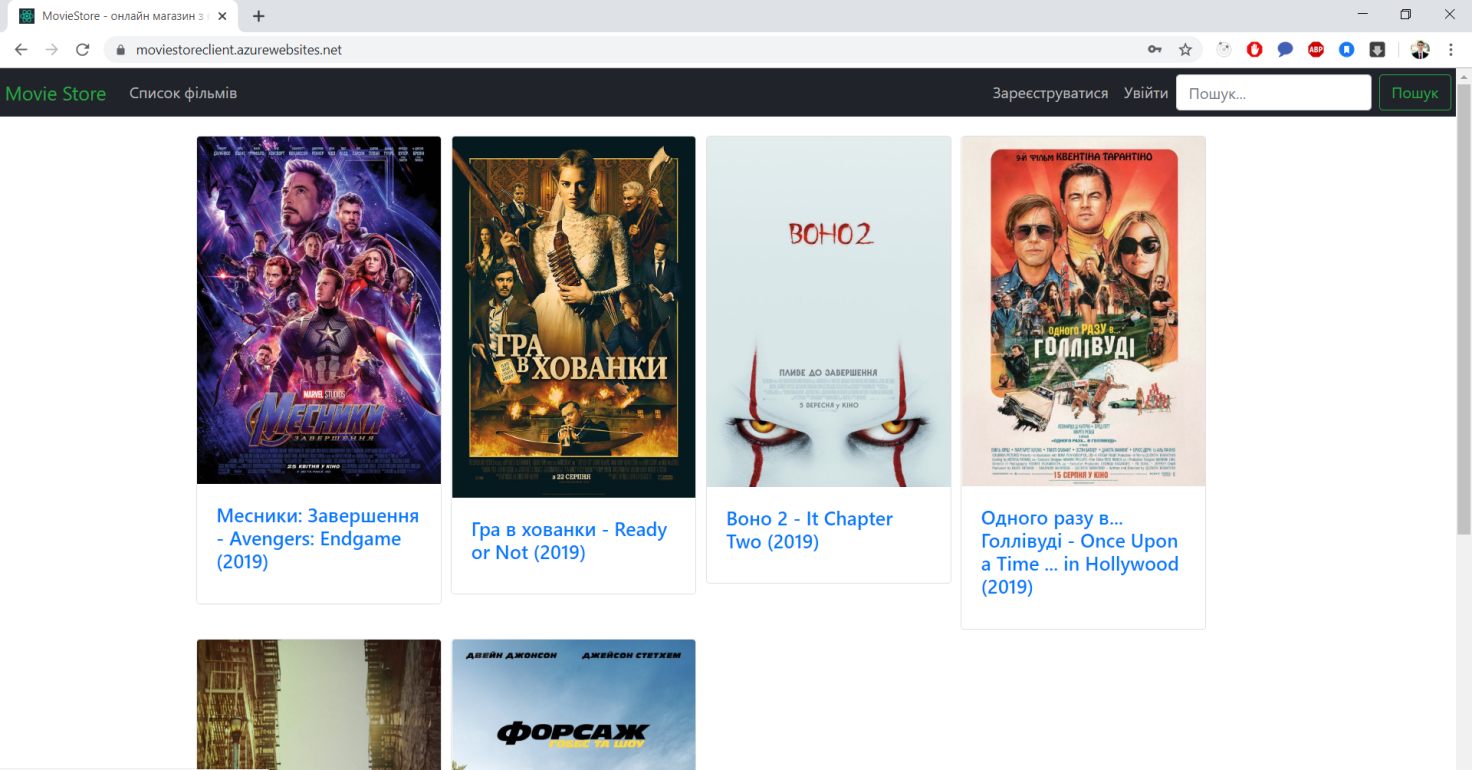
Після чого нажимаємо **Create** і чекаємо доки проект збілдиться і задеплоїться на відповідний сервіс. Посилання за яким він буде доступний – це <https://moviestoreapi.azurewebsites.net/> яке відкриється одразу по завершенню процесу деплойменту. Для перевірки правильності роботи сайту скористаємось програмою **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.