

Mariusz Dworniczak

Od
Dockera
do chmury
Azure

Aplikacje cloud-native w **Azure**

Budowa, wdrażanie i bezpieczeństwo

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz wydawca dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz wydawca nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Redaktor prowadzący: Natalia Hermansa

Projekt okładki: Studio Gravite/Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki

Grafika na okładce została wykorzystana za zgodą AdobeStock.com.

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: helion@helion.pl
WWW: helion.pl (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
helion.pl/user/opinie/azupra
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Kody źródłowe wybranych przykładów dostępne są pod adresem:
<https://ftp.helion.pl/przyklady/azupra.zip>

ISBN: 978-83-289-4025-3

Copyright © Helion S.A. 2026

Printed in Poland.

- Kup książkę
- Poleć książkę
- Oceń książkę

- Księgarnia internetowa
- Lubię to! » Nasza społeczność

Spis treści

WSTĘP	Dlaczego cloud-native i Microsoft Azure?	7
ROZDZIAŁ 1.	Architektura cloud-native — projektowanie dla nowoczesnej chmury	9
1.1.	Filozofia 3-warstwowa w praktyce cloud-native	9
1.2.	Kluczowe wzorce projektowe — fundament czystej architektury	10
1.2.1.	MVC (Model-View-Controller)	10
1.2.2.	Repository Pattern (wzorzec repozytorium)	11
1.2.3.	DTO (Data Transfer Object)	12
1.3.	Projekt — Cloud Task Manager	13
1.3.1.	Zakres funkcjonalny (co budujemy?)	13
1.3.2.	Wymagania techniczne (jak budujemy?)	14
1.3.3.	Wymagania chmurowe (gdzie wdramy?)	14
1.4.	Planowanie architektury — standard C4 i mapowanie chmurowe	15
1.4.1.	Poziom kontenerów (C4 Container Level)	15
1.4.2.	Wzorzec architektury — nasz „gold standard”	15
1.4.3.	Przykładowy plan projektu (wzorzec do dokumentacji)	16
1.5.	Lab 01 — Twój diagram architektury (artefakt nr 1)	16
1.6.	Przykładowe rozwiązanie lab 01 (artefakt nr 1)	17
1.7.	Podsumowanie rozdziału — architektura cloud-native	24
1.8.	Sprawdź swoją wiedzę — architektura i planowanie	24
ROZDZIAŁ 2.	Warsztat pracy — system kontroli wersji i konteneryzacja	26
2.1.	Wymagania — Twoje laboratorium	26
2.2.	Dlaczego Docker? Rozwiązanie problemu „u mnie nie działa”	26
2.3.	Podstawowe pojęcia i cykl życia kontenera	27
2.4.	Instalacja i rozwiązywanie problemów (Windows, Mac, Linux)	28
2.5.	Sprawdź swoją wiedzę — Docker	30
2.6.	Git — Twoja polisa ubezpieczeniowa	30
2.7.	Docker Compose — dyrygent Twojej architektury	31
2.8.	Nasz rdzeń — docker-compose.yml dla Cloud Task Managera	32
2.9.	Lab 02 — Twoje pierwsze repozytorium i kontenery (artefakt nr 2)	33
2.10.	Przykładowe rozwiązanie lab 02 (artefakt nr 2)	33
2.11.	Podsumowanie rozdziału — środowisko wielokontenerowe	42
2.12.	Sprawdź swoją wiedzę — Docker Compose	43

ROZDZIAŁ 3. Frontend w nowoczesnym wydaniu (React, Vite i komunikacja z API)	45
3.1. Teoria architektury — SPA vs MPA	45
3.2. Budowa aplikacji — komponenty i hooki	45
3.3. Planowanie struktury (podejście architektoniczne)	46
3.4. Komunikacja z API (dobre praktyki)	46
3.5. Docker dla frontendu	47
3.6. Lab 03 — działająca warstwa prezentacji (artefakt nr 3)	47
3.7. Przykładowe rozwiązanie lab 03 (artefakt nr 3)	47
3.8. Podsumowanie rozdziału — frontend jako niezależna usługa	56
3.9. Sprawdź swoją wiedzę — nowoczesny frontend	57
ROZDZIAŁ 4. Backend w architekturze aplikacji chmurowej	59
4.1. Wprowadzenie do backendu	59
4.2. Architektura Clean Architecture (struktura projektu)	59
4.3. Integracja z bazą danych (Azure SQL)	60
4.4. Konteneryzacja backendu	60
4.5. Lab 04 — działająca warstwa logiki backendu (artefakt nr 4)	61
4.6. Przykładowe rozwiązanie lab 04 (artefakt nr 4)	61
4.7. Podsumowanie rozdziału — integracja Full-Stack	79
4.8. Sprawdź swoją wiedzę — architektura i integracja	79
ROZDZIAŁ 5. Architektura Cloud-Ready i stabilność kontraktu	81
5.1. Implementacja DTO — budowa stabilnego kontraktu	81
5.2. SQL Server i Docker Volumes — nieśmiertelność danych	82
5.3. Przejście na migracje EF Core	82
5.4. Budowa produkcyjnego UI w bibliotece React	82
5.5. Lab 05 — system gotowy na chmurę (artefakt nr 5)	83
5.6. Przykładowe rozwiązanie lab 05 (artefakt nr 5)	83
5.7. Podsumowanie rozdziału — transformacja aplikacji z prototypu w stabilny system	99
5.8. Sprawdź swoją wiedzę — architektura, dane i kontenery	99
ROZDZIAŁ 6. Witaj w Azure	102
6.1. Budżet 0 zł, możliwości 100%	102
6.1.1. Ścieżka 1. Azure for Students (dla studentów)	103
6.1.2. Ścieżka 2. Bezpłatne konto Azure (dla każdego)	106
6.1.3. Ścieżka 3. Azure Dev Tools for Teaching (dla wykładowców i kadr EDU)	107
6.2. Pierwsze kroki w chmurze Azure	108
6.2.1. Subskrypcja	108
6.2.2. Regiony — dlaczego Poland Central to Twój pierwszy wybór?	108
6.2.3. Grupy zasobów	109
6.3. Azure Portal — interfejs zarządzania i administracji (Control Plane)	110
6.3.1. Global Search i Marketplace — punkt wejścia	110
6.3.2. Konfigurowalny pulpit (dashboard) i widoki współdzielone	110
6.3.3. Azure Cloud Shell — terminal wewnątrz przeglądarki	112
6.3.4. Activity Log i Resource Explorer — transparentność zmian	112
6.3.5. Monitorowanie i Advisor — proaktywna optymalizacja	113
6.4. Podsumowanie rozdziału — Twoje konto w Azure	113
6.5. Sprawdź swoją wiedzę — fundamenty Azure	114

ROZDZIAŁ 7. Wdrożenie aplikacji w Azure i zarządzanie dostępem	116
7.1. Lab 06 — wdrożenie aplikacji w Azure (artefakt nr 6)	116
7.2. Przykładowe rozwiązanie lab 06 (artefakt nr 6)	116
7.3. Podsumowanie rozdziału — architektura cloud-native w praktyce	140
7.4. Sprawdź swoją wiedzę — architektura i wdrożenie w Azure	140
ROZDZIAŁ 8. Skalowanie, bezpieczeństwo i usługi zewnętrzne	142
8.1. Lab 07 — zabezpieczona i skalowalna aplikacja (artefakt nr 7)	142
8.2. Przykładowe rozwiązanie lab 07 (artefakt nr 7)	143
8.3. Podsumowanie rozdziału — aplikacja klasy enterprise	155
8.4. Sprawdź swoją wiedzę — skalowanie, bezpieczeństwo i usługi zewnętrzne	155
ROZDZIAŁ 9. Testowanie, dokumentacja i automatyzacja (CI/CD)	157
9.1. Lab 08 — wbudowany „bezpiecznik” i wdrożony automat CI/CD (artefakt nr 8)	158
9.2. Przykładowe rozwiązanie lab 08 (artefakt nr 8)	158
9.2.1. Testy jednostkowe — Twój automatyczny kontroler jakości	158
9.2.2. Automatyczne wdrożenie	160
9.2.3. Implementacja logiki — usuwanie i edycja	165
9.2.4. Twoja profesjonalna dokumentacja w chmurze	169
9.2.5. Finalna dokumentacja — README.md	171
9.3. Podsumowanie rozdziału — testowanie, dokumentacja, automatyzacja	171
9.4. Sprawdź swoją wiedzę — testowanie, dokumentacja i automatyzacja	172
ROZDZIAŁ 10. Nowy horyzont — Twoja droga w chmurze Azure	173
10.1. Gratulacje!	173
10.2. Co dalej? Wskazówki	173
10.3. Słowo na pożegnanie	174
10.4. Ostatnia rada od autora	174
Bibliografia i materiały źródłowe	175
1. Oficjalne źródła Microsoftu (zawsze aktualne)	175
2. Książki (głęboka wiedza praktyczna)	175
3. Społeczność i narzędzia	175
4. Spis listingów	176

WSTĘP

Dlaczego cloud-native i Microsoft Azure?

Witaj w świecie, w którym pojęcie „serwera pod biurkiem” odeszło do lamusa. Witaj w erze, w której Twoja aplikacja nie jest tylko zbiorem plików, ale żywym organizmem, który skaluje się, zabezpiecza i naprawia niemal automatycznie. Witaj w świecie **cloud-native**.

Jeśli kiedykolwiek męczyłeś się z ręcznym kopiowaniem plików na FTP, jeśli konfiguracja bazy danych na produkcji zajęła Ci całą noc albo jeśli boisz się, że Twój system padnie przy większym ruchu — ta książka jest dla Ciebie.

Co zyskasz dzięki tej lekturze? To nie jest teoretyczna rozprawa o wyższości chmury nad własnym serwerem. To kompletny, praktyczny warsztat. Budujemy wspólnie jedną, zaawansowaną aplikację — od pierwszego wiersza kodu, przez konteneryzację, aż po w pełni zautomatyzowane wdrożenie w Microsoft Azure.

- **Standardy rynkowe.** Nie uczymy się „klikania”. Uczymy się architektury 3-warstwowej, wzorców DTO i Repository — czyli tego, czego wymagają najlepsi pracodawcy.
- **Niezależność dzięki Dockerowi.** Nieważne, co masz zainstalowane na komputerze. Dzięki Dockerowi Twoje środowisko będzie identyczne jak to, które finalnie trafi do chmury. Zero niespodzianek typu „u mnie nie działa”.
- **Bezpłatne konto Azure.** Chmura Azure jest dostępna dla każdego, a ja pokażę Ci, jak zacząć w niej budować zupełnie za darmo.
 - **Jeśli jesteś studentem.** Wykorzystasz *Azure for Students* — bez karty kredytowej otrzymasz 100 USD kredytu i darmowe limity na rok.
 - **Jeśli jesteś profesjonalistą/hobbystą.** Pokażę Ci, jak aktywować *Bezpłatne konto platformy Azure*, które daje 200 USD na start i dostęp do najpopularniejszych usług za zero złotych przez pierwsze 12 miesięcy.
- **Inwestycja w przyszłość.** Umiejętność budowy aplikacji od razu pod chmurę (*cloud-native*) to dziś najkrótsza droga do roli senior developera lub cloud architecta.

Czego nauczysz się w praktyce? Moim celem jest przekazanie Ci „pamięci mięśniowej” nowoczesnego inżyniera. Nie będziemy tylko pisać kodu — będziemy budować kompletny ekosystem.

Dzięki tej lekturze:

- **Opanujesz cykl życia aplikacji.** Od lokalnego `docker-compose up`, do publicznego adresu URL w domenie `azurewebsites.net`.
- **Zrozumiesz separację warstw.** Nauczysz się, dlaczego frontend nie powinien wiedzieć o bazie danych i jak bezpiecznie połączyć je przez API.

- **Zabezpieczysz swoje dane.** Dowiesz się, jak używać Azure Key Vault, by Twoje hasła nigdy nie trafiły do repozytorium kodu.
- **Wdrożysz CI/CD.** Skonfigurujesz GitHub Actions tak, aby każdy Twój commit automatycznie testował i odświeżał aplikację na produkcji.

Dla kogo jest ta książka? Niezależnie od tego, czy jesteś studentem stawiającym pierwsze kroki w IT, programistą chcącym przeskoczyć na poziom *cloud-native*, czy administratorem, który chce zrozumieć świat deweloperów — ten materiał jest dla Ciebie.

Jak korzystać z tej lektury? To nie jest powieść do poduszki. To instrukcja obsługi Twojej nowej kariery. Każdy rozdział to kolejny artefakt — kolejna cegiełka Twojego projektu. Nie bój się psuć, nie bój się błędów w terminalu. Naprawianie ich to najsukcesowniejsza lekcja, jaką możesz odebrać.

Twój kod zasługuje na coś więcej niż Twój dysk twardy. Zasługuje na chmurę. **Zaczynamy wdrożenie Twojego sukcesu!**



Kody źródłowe wybranych przykładów dostępne są pod adresem:

<https://ftp.helion.pl/przyklady/azupra.zip>

ROZDZIAŁ 1.

Architektura cloud-native — projektowanie dla nowoczesnej chmury

Zanim napiszemy pierwszą linię kodu i zanim uruchomimy jakikolwiek serwer, musimy zdefiniować fundamenty. W nowoczesnym IT nie budujemy już aplikacji „na serwer” — budujemy je **dla chmury**.

Cloud-native to podejście do projektowania, budowania i uruchamiania aplikacji, które w pełni wykorzystują zalety modelu obliczeń w chmurze. Nie chodzi tylko o to, *gdzie* aplikacja działa, ale *jak* została stworzona. Systemy *cloud-native* charakteryzują się:

- **Modułowością** — rozbięciem na niezależne komponenty.
- **Odpornością** — zdolnością do przetrwania awarii pojedynczych usług.
- **Skalowalnością** — łatwym dopasowaniem do rosnącej liczby użytkowników.
- **Przenośnością** — niezależnością od konkretnego sprzętu czy dostawcy.

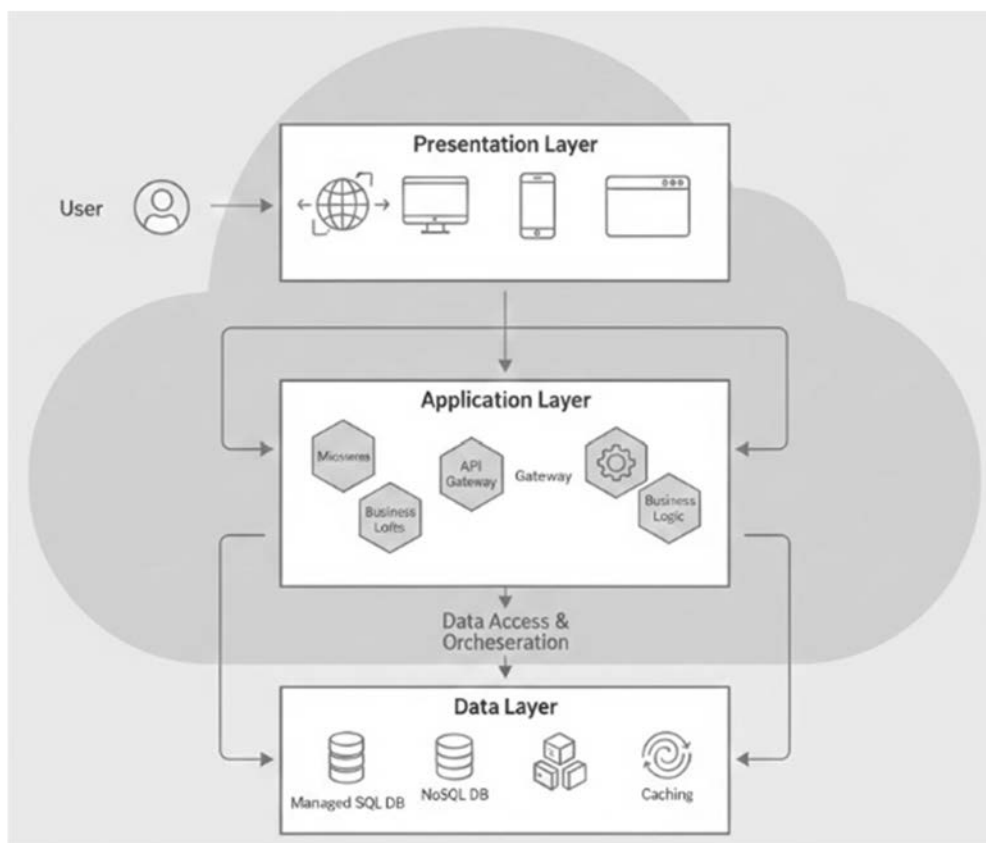
W tym rozdziale przejdziemy ścieżkę od teorii do praktycznego dowodu na to, że dobrze zaprojektowany system jest wolny od ograniczeń Twojego komputera.

1.1. Filozofia 3-warstwowa w praktyce cloud-native

W świecie cloud-native najskuteczniejszym sposobem na zachowanie porządku i skalowalności jest rygorystyczny podział na trzy warstwy (przykładową implementację przedstawia rysunek 1.1). Każda z nich pełni inną funkcję i może być rozwijana przez różne zespoły lub w różnych technologiach.

- Warstwa prezentacji (*Presentation Layer*):
 - **Technologia** — np. React.
 - **Rola** — to „twarz” aplikacji. Odpowiada wyłącznie za interfejs użytkownika i komunikację z API. W podejściu chmurowym traktujemy ją jako statyczny zasób, który można serwować globalnie z punktów brzegowych (*Edge*).
- Warstwa aplikacji (*Application Layer*):
 - **Technologia** — np. .NET Web API lub Node.js.
 - **Rola** — logika biznesowa, procesowanie danych, autoryzacja. To serce systemu, które w chmurze będzie skalować się w zależności od obciążenia.
- Warstwa danych (*Data Layer*):
 - **Technologia** — np. Azure SQL/Cosmos DB.

- o **Rola** — trwałe składowanie informacji. W architekturze *cloud-native* dbamy o to, by baza była oddzielona od logiki, co pozwala na jej niezależne backupowanie i replikację.



RYSUNEK 1.1. Schemat trzyczarstwowej aplikacji

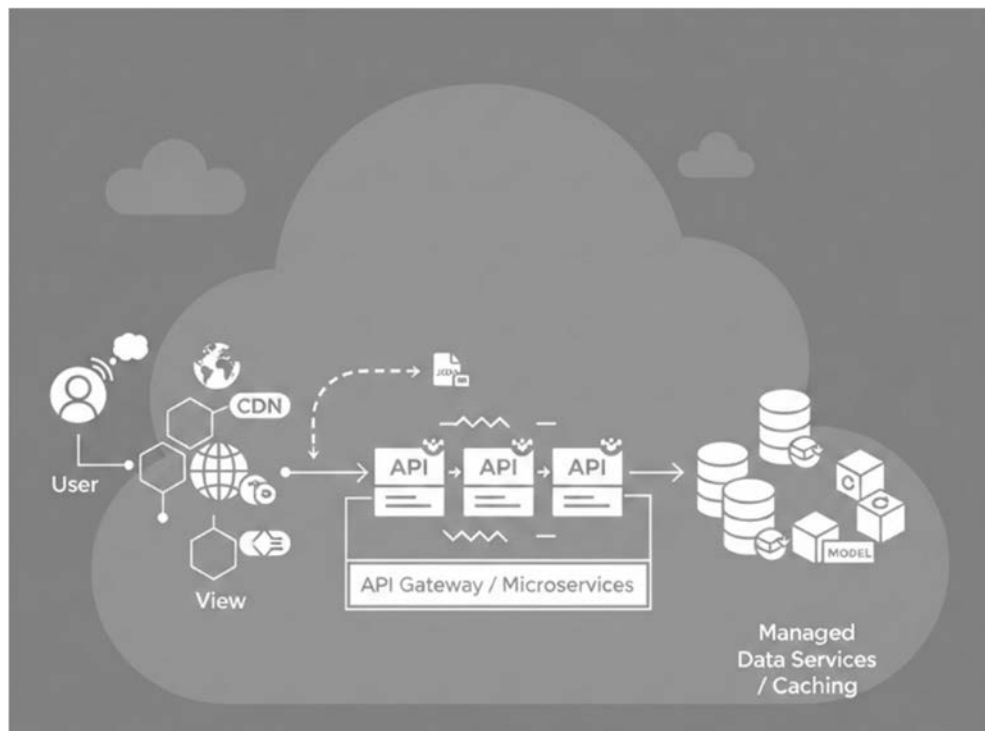
1.2. Kluczowe wzorce projektowe — fundament czystej architektury

W architekturze *cloud-native* kod musi nie tylko działać, ale przede wszystkim być łatwy do testowania, rozbudowy i wymiany. Stosujemy trzy fundamentalne wzorce, które dbają o to, by nasza aplikacja nie stała się „nienaruszalnym monolitem”.

1.2.1. MVC (Model-View-Controller)

Jest to wzorec strukturalny, który dzieli aplikację na trzy główne komponenty. W naszym projekcie *cloud-native* wzorec ten ewoluje w stronę **Web API** ze względu na podział na osobny frontend i backend.

- **Co to jest?** Mechanizm oddzielający dane (*Model*) i logikę sterowania (*Controller*) od sposobu ich prezentacji (*View*).
- **Po co to stosujemy?** Aby zmiany w wyglądzie aplikacji (np. zmiana przycisku w aplikacji korzystającej z biblioteki React) nie wymagały modyfikacji logiki na serwerze.
- **Przykład.** Użytkownik klika *Zapisz zadanie*. **Controller** odbiera to żądanie, instruuje **Model**, aby zaktualizował dane, a następnie zwraca informację o sukcesie, którą warstwa prezentacji wyświetla użytkownikowi. Przykładowy diagram wzorca MVC pokazuje rysunek 1.2.



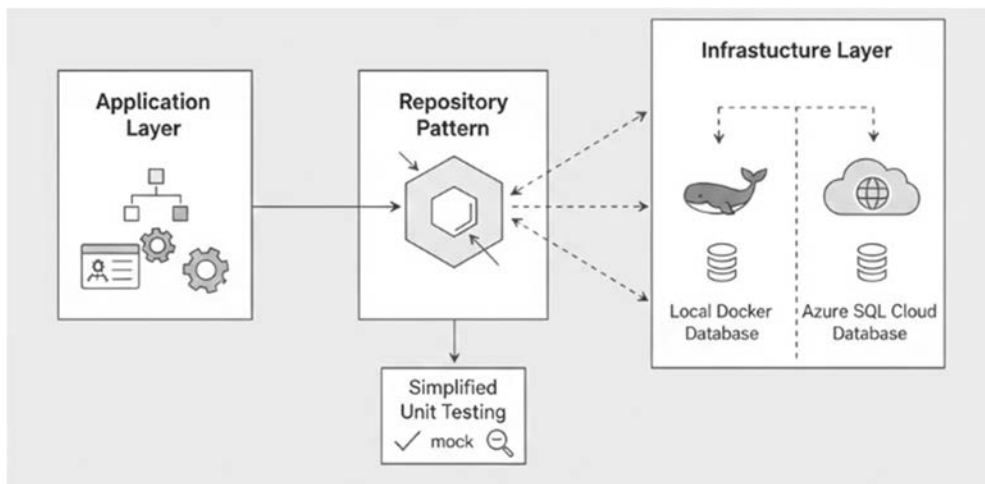
RYSUNEK 1.2. Diagram wzorca MVC

1.2.2. Repository Pattern (wzorzec repozytorium)

To warstwa pośrednicząca między logiką biznesową aplikacji a źródłem danych.

- **Co to jest?** Abstrakcja, która udaje kolekcję obiektów w pamięci, ukrywając przed resztą systemu fakt, że pod spodem znajduje się baza danych SQL, plik JSON czy zewnętrzne API.
- **Po co to stosujemy?** Aby zapewnić **niezależność od dostawcy bazy danych**. Dzięki temu Twoja logika biznesowa (*Application Layer*) nie może stwierdzić, czy rozmawiasz z lokalnym kontenerem Docker, czy z usługą Azure SQL w chmurze. Znacznie ułatwia to pisanie testów jednostkowych (mockowanie bazy).

- **Przykład.** Zamiast pisać zapytanie SQL bezpośrednio w kontrolerze, wywołujesz metodę `_taskRepository.GetAllActive()`. Jeśli w przyszłości zmienisz bazę z SQL na NoSQL (Cosmos DB), zmienisz tylko kod wewnątrz repozytorium — reszta aplikacji pozostanie nietknięta. Na rysunku 1.3 przedstawiono diagram wzorca repozytorium.



RYSUNEK 1.3. Diagram wzorca repozytorium

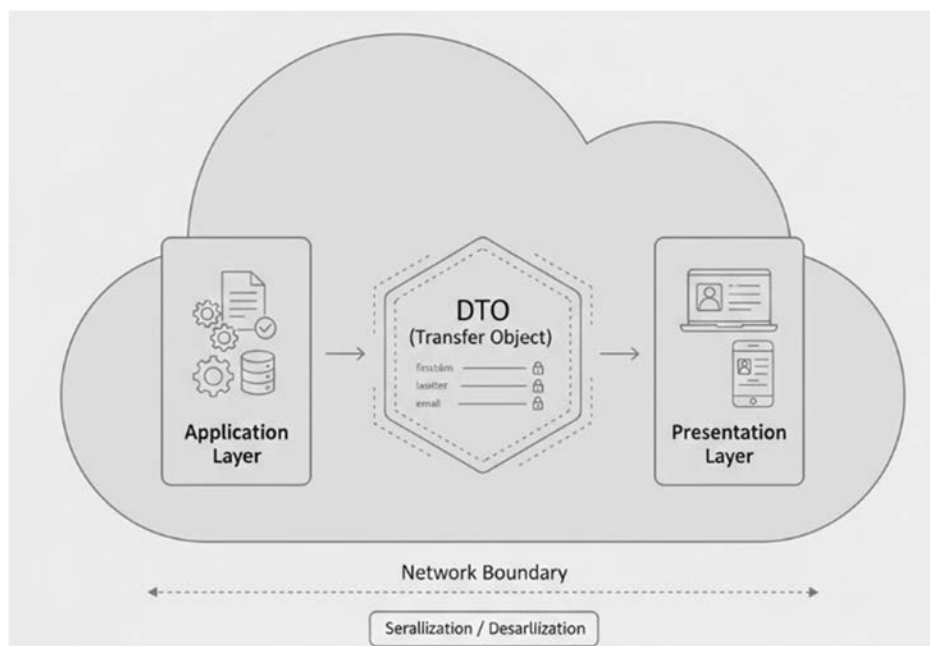
1.2.3. DTO (Data Transfer Object)

Obiekty DTO to proste kontenery służące wyłącznie do przesyłania danych między procesami (np. backend ↔ frontend). Diagram DTO (Data Transfer Object) przedstawiono na rysunku 1.4.

- **Co to jest?** To „paszporty dla danych”. Są to klasy, które nie zawierają żadnej logiki biznesowej, a jedynie zestaw pól.
- Po co to stosujemy?
 - **Bezpieczeństwo.** Nie chcemy wysyłać do przeglądarki całego obiektu bazy danych (który może zawierać np. hash hasła lub ukryte ID). Wysyłamy tylko to, co UI musi wyświetlić.
 - **Wydajność.** Ograniczamy rozmiar przesyłanych paczek danych (tzw. *payload*).
 - **Odporność na zmiany.** Możesz zmienić strukturę tabeli w bazie danych, nie psując kontraktu z frontendem, o ile zmapujesz nowe dane do starego formatu DTO.
- **Przykład.** Twoja encja w bazie User ma 20 pól, w tym datę utworzenia i tajne tokeny. Twoje UserDTO wysyłane do biblioteki React zawiera tylko `FirstName`, `LastName` i `AvatarUrl`.

Uzasadnienie techniczne zastosowania wzorców:

Zastosowanie tych wzorców sprawia, że Twoja aplikacja staje się **luźno powiązana (loosely coupled)**. To kluczowa cecha cloud-native: każdy element systemu może ewoluować niezależnie, co jest niezbędne w dynamicznym środowisku chmurowym Azure.



RYSUNEK 1.4. Diagram DTO (Data Transfer Object)

1.3. Projekt — Cloud Task Manager

W tej książce nie ograniczamy się do prostych przykładów typu „Hello, World”. Budujemy rozbudowany system, który posłuży nam jako kompletna ilustracja procesu tworzenia natywnej aplikacji chmurowej. Przechodząc przez kolejne rozdziały, nauczysz się krok po kroku, jak przekuć wymagania biznesowe w działający, zabezpieczony i skalowalny produkt w Azure.

Cel projektu

Stworzenie platformy do zarządzania zadaniami, która demonstruje pełny cykl życia aplikacji: od lokalnego kontenera, przez automatyzację testów, aż po wdrożenie w chmurze Azure.

1.3.1. Zakres funkcjonalny (co budujemy?)

Nasza aplikacja, niezależnie od Twojej finalnej domeny (może to być system rezerwacji lub LMS), musi implementować następujące mechanizmy:

- **Zarządzanie tożsamością** — implementacja rejestracji i logowania użytkowników. Wykorzystamy lokalną bazę dla fazy dev, by docelowo zintegrować się z **Azure Entra ID** (standard korporacyjny).
- **Kompletny cykl CRUD** — pełna obsługa encji biznesowych (tworzenie, odczyt, edycja, usuwanie) z zachowaniem integralności danych.

- **Profesjonalny interfejs (UI/UX)** — wykorzystanie biblioteki React do stworzenia widoków list, filtrów oraz szczegółowych formularzy edycji.

W naszych przykładowych rozwiązaniach zrobimy Cloud Tasks Managera, ale Ty możesz zmienić nazwy pól i wykorzystać to jako szablon do Twoich przyszłych projektów programistycznych.

1.3.2. Wymagania techniczne (jak budujemy?)

Aby projekt można było uznać za zgodny z duchem cloud-native, musi on spełniać rygorystyczne warunki techniczne:

- **Wersjonowanie i historia** — repozytorium GitHub z jasną historią commitów (każdy etap to osobny artefakt).
- **Struktura monorepo** — czysty podział na foldery */frontend* i */backend*, co odzwierciedla architekturę 3-warstwową.
- **Orkiestracja lokalna (*Docker Compose*)** — cały system (React, API, SQL) musi „wstawać” po wpisaniu jednej komendy w terminalu. To gwarancja, że Twoje środowisko jest niezależne od systemu operacyjnego.
- **Konfiguracja 12-Factor App** — wykorzystanie zmiennych środowiskowych (*.env*) do konfiguracji połączeń, co umożliwi płynne przejście między bazą lokalną a chmurową bez zmiany kodu.

1.3.3. Wymagania chmurowe (gdzie wdrażamy?)

Finałem Twojej pracy będzie publicznie dostępny adres URL, pod którym aplikacja będzie działać w środowisku Microsoft Azure, podzielonym na:

- **Frontend** — hostowany w usłudze Azure Static Web Apps.
- **Backend** — uruchomiony w usłudze Azure App Service.
- **Database** — zarządzana baza Azure SQL w regionie Poland Central.
(Uwaga: Jeśli Twoja subskrypcja blokuje ten region, wybierz np. Germany West Central).

Dlaczego nasze podejście jest lepsze?

Większość kursów uczy technologii w izolacji. My uczymy **procesu**. Widząc, jak wzorce MVC, Repository i DTO współpracują ze sobą w Dockerze, a potem idealnie mapują się na usługi Azure, zyskasz kompetencje, których poszukują architekci rozwiązań chmurowych. Nie uczysz się teorii o chmurze — uczysz się, jak do niej wejść z gotowym produktem.

ROZDZIAŁ 7.

Wdrożenie aplikacji w Azure i zarządzanie dostępem

Lokalny Docker służył nam do symulacji środowiska. W chmurze przechodzimy na model **PaaS (Platform as a Service)**. Oznacza to, że nie zarządzamy już systemem operacyjnym ani aktualizacjami silnika SQL — zajmuje się tym Microsoft Azure. Proces wdrażania (ang. *deployment*) to końcowy etap publikacji, polegający na przeniesieniu skompilowanego kodu źródłowego z lokalnego środowiska programistycznego do infrastruktury chmurowej. My skupiamy się wyłącznie na kodzie i bezpiecznym połączeniu usług.

Co zrobimy w tym rozdziale?

1. Przenieśmy bazę danych do **Azure SQL** i zmigramy jej schemat.
2. Uruchomimy logikę biznesową w **Azure App Service**.
3. Opublikujemy interfejs w **Azure Static Web Apps**.
4. Zabezpieczymy całość za pomocą **firewalla** i ról **RBAC**.

7.1. Lab 06 — wdrożenie aplikacji w Azure (artefakt nr 6)

Checklista do wykonania:

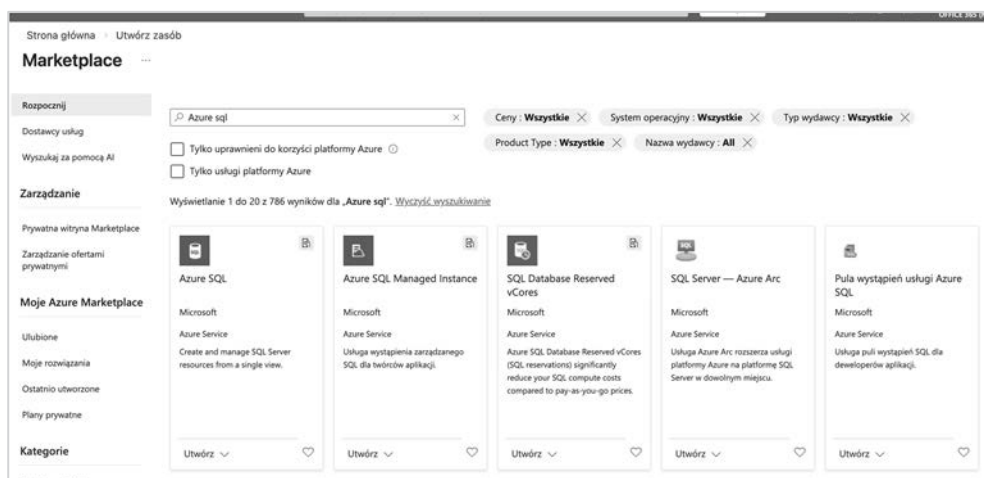
- ☐ 6.1. Azure SQL Database utworzona i dostępna przez firewall.
- ☐ 6.2. Schemat bazy danych zmigrowany przy użyciu `dotnet ef`.
- ☐ 6.3. Backend (.NET) działający pod publicznym adresem HTTPS.
- ☐ 6.4. Frontend (React) zintegrowany z produkcyjnym adresem API.
 - Działająca aplikacja z Azure.
- ☐ 6.5. Zaktualizowana dokumentacja i kod wysłany do GitHuba.

7.2. Przykładowe rozwiązanie lab 06 (artefakt nr 6)

Krok 1. Azure SQL — Twoja baza w chmurze

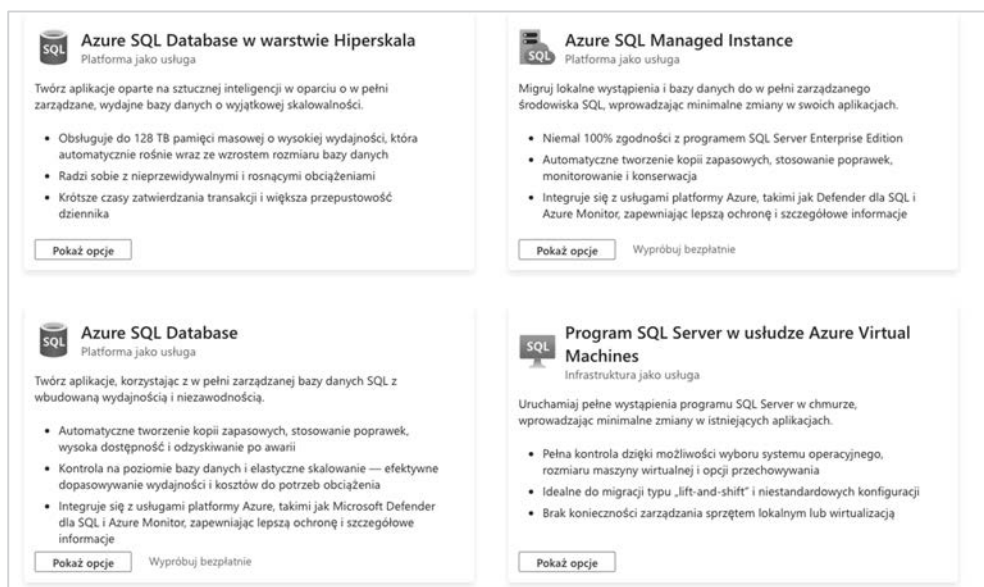
Zaczynamy od bazy, ponieważ backend już w momencie startu musi wiedzieć, gdzie się połączyć.

1. W portalu Azure wybierz *Utwórz zasób/Azure SQL* (rysunek 7.1).



RYSUNEK 7.1. Wybór bazy Azure SQL

- Wybierz typ bazy. Azure zapyta, który typ bazy chcesz wybrać. Wybierz *Azure SQL Database*. Platforma jako usługa. Kliknij *Wypróbuj bezpłatnie* (rysunek 7.2).



RYSUNEK 7.2. Wybór typu bazy

- Skonfiguruj bazę. Po wyborze typu bazy dostaniesz informację podobną do tej: *Zastosowano bezpłatną ofertę Otrzymujesz 100 000 sekund rdzenia wirtualnego, 32 GB danych i 32 GB bezpłatnej przestrzeni na kopie zapasowe miesięcznie przez cały czas trwania subskrypcji* (rysunek 7.3).

Weryfikowanie nie powiodło się. Brak wymaganych informacji lub są one nieprawidłowe.

Podstawowe informacje Przejrzyj i utwórz

Utwórz bezpłatną bazę danych Azure SQL Database. Aby dostosować ustawienia, wybierz Konfiguracja zaawansowana poniżej.

● Zastosowano bezpłatną ofertę. Otrzymujesz 100 000 sekund rdzenia wirtualnego, 32 GB danych i 32 GB bezpłatnej przestrzeni na kopie zapasowe miesięcznie przez cały czas trwania subskrypcji. Dowiedz się więcej [?]

Aby usunąć ofertę, wybierz Konfiguracja zaawansowana. [Konfiguracja zaawansowana](#)

Szczegóły projektu

Wybierz subskrypcję, aby zarządzać wdrożonymi zasobami i kosztami. Użyj grup zasobów jak folderów, aby organizować wszystkie Twoje zasoby i zarządzać nimi.

Subskrypcja *

Grupa zasobów * [Utwórz nowy](#)

Szczegóły bazy danych

Wprowadź nazwę bazy danych i wybierz serwer logiczny.

Nazwa bazy danych *

Serwer * [Utwórz nowy](#)

Podsumowanie kosztów

Środowisko obliczeniowe 0,00 USD
Wirtualny rdzeń na sekundę¹ 0,00 USD

Magazyn 0,00 USD
Koszt na GB 0,00 USD
Wybrano maksymalną ilość miejsca do 32 magazynowania (w GB)

Rozdzielanie nadwyżek **Wylączone**
Nie będą naliczane opłaty za użycie w ramach bezpłatnych limitów. Baza danych zostanie wstrzymana automatycznie po osiągnięciu bezpłatnych limitów.

Szacowana suma **Bezpłatne¹**

Ceny mają charakter szacunkowy. Wyświetl kalkulator cen platformy Azure [?]. Odstawione opłaty będą wyświetlane w walucie lokalnej w widokach analizy kosztów i rozliczeń.

¹ Bezpłatne bazy danych są rozliczane w sekundach rdzenia wirtualnego na podstawie kombinacji wykorzystania procesora i pamięci. Dowiedz się więcej na temat rozliczeń w przypadku rozłążeń bezpłatnych [?].

RYSUNEK 7.3. Konfiguracja bazy Azure SQL

Konfiguracja bazy:

- o **Grupa zasobów.** Utwórz nową grupę, np. **rg-cloud-task-manager**. Później będziesz używał tej samej grupy dla reszty projektu.



Dlaczego to ważne? Grupa zasobów pozwala jednym kliknięciem usunąć cały projekt (bazę, serwer, API), gdy skończysz naukę, co chroni Cię przed niechcianymi kosztami w przyszłości.

- o Polecenie **Serwer/Utwórz nowy** — przeniesie Cię do nowego okna (rysunek 7.4).

Utwórz serwer usługi SQL Database Microsoft

Szczegóły serwera

Wprowadź wymagane ustawienia dla tego serwera, w tym podając nazwę i lokalizację. Ten serwer zostanie utworzony w tej samej subskrypcji i grupie zasobów co baza danych.

Nazwa serwera * [.database.windows.net](#)

Lokalizacja *

Uwierzytelnianie

● Nazwa serwera nie powinna zawierać słów zastrzeżonych.
● Podana nazwa serwera jest dostępna.
● Nazwa serwera nie może zaczynać się ani kończyć łącznikami „-”, ani nie może zawierać dwóch łączników „-” w trzecim i czwartym miejscu nazwy.

RYSUNEK 7.4. Konfigurowanie nazwy serwera SQL



Uwaga: w lutym 2026 region Poland Central nie był dozwolony w ramach subskrypcji studenckiej **Azure for Students** w Polsce. Dozwolone regiony: "germanywestcentral", "francecentral", "swedencentral", "norwayeast", "spaincentral". W takim wypadku wybierz: Germany West Central.

- o **Nazwa serwera: cloud-app-db.**



Uwaga: nazwa powinna być unikalna w skali całego Azure, więc jeśli wybrana przez Ciebie nazwa okaże się zajęta, dodaj coś unikalnego, np. datę utworzenia tej bazy).

- o **Uwierzytelnienie.** Użyj *SQL Authentication* (np. login: dbadmin, hasło: Silne ↗Hasło123!). Bezpieczeństwem haseł zajmiemy się w dalszych rozdziałach. Na razie zapisz gdzieś te dane, nazwę serwera, identyfikator administratora i hasło (rysunek 7.5).

RYSUNEK 7.5. Uwierzytelnianie bazy

4. Aby zakończyć, kliknij *Przejrzyj i utwórz* (rysunek 7.6).

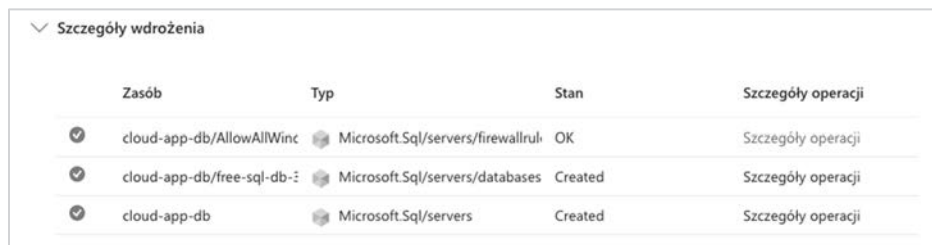
RYSUNEK 7.6. Finalizacja tworzenia bazy danych

Potwierdzamy przyciskiem *Utwórz*.

Poczekaj, aż baza zostanie utworzona. Dostaniesz komunikat:

Wdrożenie zostało ukończone

W szczegółach wdrożenia będą widoczne trzy zaznaczenia zasobów (rysunek 7.7).



Szczegóły wdrożenia				
	Zasób	Typ	Stan	Szczegóły operacji
✓	cloud-app-db/AllowAllWindowsFirewallRule	Microsoft.Sql/servers/firewallrules	OK	Szczegóły operacji
✓	cloud-app-db/free-sql-db-3460555	Microsoft.Sql/servers/databases	Created	Szczegóły operacji
✓	cloud-app-db	Microsoft.Sql/servers	Created	Szczegóły operacji

RYSUNEK 7.7. Szczegóły wdrożenia bazy Azure SQL

Krok 2. Połączenie aplikacji z Azure SQL

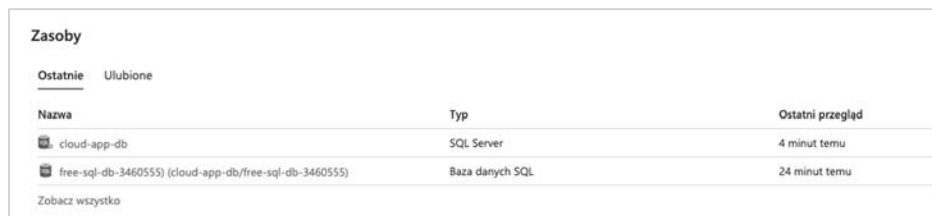
Baza danych już działa, ale jest niczym sejf w innym mieście — musisz mieć klucz i adres, żeby się do niego dostać.

Krok 2.1. Pobranie Connection String (parametry połączenia)

Baza danych już działa, ale musimy pobrać „adres i klucz”, aby backend mógł się z nią połączyć.

1. Znajdź bazę (nie serwer!).

- o Na stronie głównej portalu Azure lub wewnątrz swojej grupy zasobów znajdź zasób, który ma ikonę niebieskiego walca i typ *Baza danych SQL (SQL database)*.
- o Najprawdopodobniej nazywa się ona *free-sql-db-xxxxxxx* (lub podobnie). Kliknij ją (rysunek 7.8).



Zasoby		
Ostatnie Ulubione		
Nazwa	Typ	Ostatni przegląd
cloud-app-db	SQL Server	4 minut temu
free-sql-db-3460555 (cloud-app-db/free-sql-db-3460555)	Baza danych SQL	24 minut temu
Zobacz wszystko		

RYSUNEK 7.8. Wybór bazy danych

2. Otwórz parametry połączenia:

- o Będąc już w widoku bazy, spójrz na menu po lewej stronie.
- o W sekcji *Ustawienia (Settings)* kliknij *Parametry połączenia (Connection strings)*.

3. Skopiuj ADO.NET:

- o Upewnij się, że wybrana jest pierwsza zakładka — *ADO.NET*.

PROGRAM PARTNERSKI

— GRUPY HELION —



1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ

Zmień swoją stronę WWW w działający bankomat!

Dowiedz się więcej i dołącz już dzisiaj!

<http://program-partnerski.helion.pl>

GRUPA
Helion 

Zbuduj swoją pierwszą aplikację specjalnie dla chmury

Aplikacje typu *cloud-native*, na których budowie i implementacji skupia się ta książka, są tworzone specjalnie dla środowisk chmurowych. Tego typu rozwiązanie, korzystające z elastyczności chmury, opiera się na architekturze mikroservisów, konteneryzacji, interfejsach API i automatyzacji. Dzięki temu jest szybkie we wdrażaniu, odporne na awarie i łatwe do rozbudowy.

Ten poradnik pomoże Ci stworzyć od zera zaawansowaną aplikację: od pierwszego wiersza kodu, przez konteneryzację (Docker), aż po pełną automatyzację (CI/CD) w chmurze Microsoft Azure. W trakcie pracy nad projektem dowiesz się, jak pozyskać odpowiedni budżet i darmowe limity od Microsoftu, poznasz standardy rynkowe (architektura 3-warstwowa, wzorce DTO i Repository), zaznajomisz się również z zagadnieniami dotyczącymi bezpieczeństwa, takimi jak korzystanie z Azure Key Vault, by hasła nigdy nie trafiły do repozytoriów kodu.

Od
Dockera
do chmury
Azure

Mariusz Dworniczak

Doświadczony menedżer techniczny i certyfikowany architekt chmurowy (AWS, Azure, Google Cloud). Od ponad dwudziestu lat zarządza złożonymi programami IT i migracjami infrastruktury w globalnych organizacjach, takich jak IBM, BOX czy Toyota. Wykładowca na Uniwersytecie WSB Merito oraz doświadczony mentor techniczny. Specjalizuje się w automatyzacji infrastruktury (IaC), a także łączeniu twardych kompetencji technicznych z nowoczesnymi metodykami zarządzania projektami. Prywatnie pasjonat technologii blockchain, AI i gry w szachy.

Helion



helion.pl



HELION S.A.
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej! ▶



ISBN 978-83-289-4025-3



Cena: 67,00 zł