

Wyższa Szkoła Informatyki Stosowanej i Zarządzania

pod auspicjami Polskiej Akademii Nauk



WYDZIAŁ INFORMATYKI

Kierunek INFORMATYKA

Studia II stopnia (dyplom magistra inżyniera)



PRACA DYPLOMOWA

Piotr Fogiel

AUTOMATYZACJA WDRAŻANIA ŚRODOWISK W CHMURZE AZURE Z WYKORZYSTANIEM AUTORSKIEJ APLIKACJI WEBOWEJ

Promotor pracy:

Dr inż. Jarosław Sikorski

WARSZAWA, rok akademicki 2022/2023

Autor:	Piotr Fogiel
Tytuł:	Automatyzacja wdrażania środowisk w chmurze Azure z wykorzystaniem autorskiej aplikacji webowej.
Wydział:	Informatyki
Kierunek:	Informatyka
Specjalność:	Architektura i Bezpieczeństwo Chmury Obliczeniowej
Studia:	II Stopnia (Dyplom Magistra Inżyniera)
Dziekan Wydziału:	dr inż. Jarosław Sikorski
Promotor Pracy:	dr inż. Jarosław Sikorski
Konsultant pracy:	mgr. inż. Robert Przybylski
Rok akademicki:	2022/2023



Wyższa Szkoła Informatyki Stosowanej i Zarządzania

pod auspicjami Polskiej Akademii Nauk
ul. Nowelska 6, 01-447 Warszawa

Wydział Informatyki

Kierunek: Informatyka

OŚWIADCZENIE AUTORA PRACY

Temat pracy: *Automatyzacja wdrażania środowisk w chmurze Azure z wykorzystaniem autorskiej aplikacji webowej.*

Świadom(a) odpowiedzialności prawnej oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny z obowiązującymi przepisami prawa, w szczególności z ustawą z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (Dz. U. z 1994 r. Nr 24, poz. 83 – tekst pierwotny i Dz. U. z 2000 r. Nr 80, poz. 904 – tekst jednolity, z późn. zm.). Oświadczam również, że przedstawiona praca nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu zawodowego w szkole wyższej. Oświadczam także, że wersja pracy składana w dziekanacie, pliki elektroniczne pracy zawierają identyczną treść. Oświadczam również, że wszystkie narzędzia informatyczne zastosowane do wykonania niniejszej pracy wykorzystałem(am) zgodnie z obowiązującymi przepisami prawa w zakresie ochrony własności intelektualnej i przemysłowej. Wyrażam zgodę na przetwarzanie następujących danych osobowych: imiona, nazwisko, nr albumu, temat pracy dyplomowej oraz oceny na dyplomie, ukończony kierunek oraz rok, system i rodzaj studiów, adres do korespondencji (w tym e-mail i telefon kontaktowy) dla celów badania losu absolwenta zgodnie z Rozporządzeniem Parlamentu Europejskiego i Rady Unii Europejskiej 2016/679 z dnia 27 kwietnia 2016 r. w sprawie ochrony osób fizycznych w związku z przetwarzaniem danych osobowych i w sprawie swobodnego przepływu takich danych oraz uchylenia dyrektywy 95/46/WE (RODO).

Wyrażam zgodę na publikację pracy.

Wyrażam zgodę na przetwarzanie następujących danych osobowych: imiona, nazwisko, nr albumu, ukończony kierunek oraz rok, system i rodzaj studiów, adres do korespondencji (w tym e-mail i telefon kontaktowy) dla celów promocji uczelni zgodnie z Rozporządzeniem Parlamentu Europejskiego i Rady Unii Europejskiej 2016/679 z dnia 27 kwietnia 2016 r. w sprawie ochrony osób fizycznych w związku z przetwarzaniem danych osobowych i w sprawie swobodnego przepływu takich danych oraz uchylenia dyrektywy 95/46/WE (RODO).

Wyrażam zgodę na przetwarzanie następujących danych osobowych: tematu pracy dyplomowej oraz oceny na dyplomie dla celów promocji uczelni zgodnie z Rozporządzeniem Parlamentu Europejskiego i Rady Unii Europejskiej 2016/679 z dnia 27 kwietnia 2016 r. w sprawie ochrony osób fizycznych w związku z przetwarzaniem danych osobowych i w sprawie swobodnego przepływu takich danych oraz uchylenia dyrektywy 95/46/WE (RODO).

Wyrażam zgodę na przetwarzanie następujących danych osobowych: aktualnego miejsca pracy, miejsca pracy po ukończeniu studiów oraz obecnie zajmowanego stanowiska dla celów promocji uczelni zgodnie z Rozporządzeniem Parlamentu Europejskiego i Rady Unii Europejskiej 2016/679 z dnia 27 kwietnia 2016 r. w sprawie ochrony osób fizycznych w związku z przetwarzaniem danych osobowych i w sprawie swobodnego przepływu takich danych oraz uchylenia dyrektywy 95/46/WE (RODO).

Wyrażam zgodę na przesyłanie mi informacji o ofertach uczelni i ankiet dla celów promocji uczelni zgodnie z Rozporządzeniem Parlamentu Europejskiego i Rady Unii Europejskiej 2016/679 z dnia 27 kwietnia 2016 r. w sprawie ochrony osób fizycznych w związku z przetwarzaniem danych osobowych i w sprawie swobodnego przepływu takich danych oraz uchylenia dyrektywy 95/46/WE (RODO).

Wyrażam zgodę na przetwarzanie następujących danych osobowych: imiona, nazwisko, nr albumu, ukończony kierunek, specjalność oraz rok, system i rodzaj studiów, wynik ukończenia, miejsce urodzenia w celu wydrukowania dyplomu na zlecenie WSISiZ przez drukarnię ZPW Bogucin k/Poznań z siedzibą w Kobylnicy, ul. Gnieźnieńska 127 zgodnie z Rozporządzeniem Parlamentu Europejskiego i Rady Unii Europejskiej 2016/679 z dnia 27 kwietnia 2016 r. w sprawie ochrony osób fizycznych w związku z przetwarzaniem danych osobowych i w sprawie swobodnego przepływu takich danych oraz uchylenia dyrektywy 95/46/WE (RODO).

16/03/2023

(data)

Piotr Fogiel, 19576

(student)

.....
(podpis studenta)

Niniejszy dokument został wygenerowany na podstawie danych zgromadzonych w systemie UBI.

SPIS TREŚCI

WSTĘP	9
ROZDZIAŁ 1. OPIS TECHNOLOGII	11
1.1 Definicja chmury obliczeniowej	11
1.2 Microsoft Azure	13
1.3 Azure DevOps.....	16
1.4 Szablony ARM	17
1.5 .NET 6.....	18
1.6 .NET Azure SDK	19
1.7 Angular	19
ROZDZIAŁ 2. OPIS PROBLEMU	21
2.1 Rejestrowanie aplikacji w usłudze Azure Active Directory	21
2.2 Proces tworzenia zasobów w chmurze Azure.....	24
2.3 Proces tworzenia środowiska programistycznego	27
ROZDZIAŁ 3. PROJEKT APLIKACJI.....	29
3.1 Architektura aplikacji serwerowej	29
3.2 Wzorzec projektowy CQRS.....	31
3.3 Architektura aplikacji klienckiej.....	32
3.4 Infrastruktura aplikacji.....	33
3.5 Uprawnienia aplikacji	34
3.6 Inicjalizacja aplikacji	36
ROZDZIAŁ 4. ZASADA DZIAŁANIA APLIKACJI.....	39
4.1 Tworzenie projektu	39
4.2 Tworzenie diagramu wdrożeniowego.....	40
4.3 Okno dialogowe wdrożenia	48
4.4 Proces wdrożenia	51
4.5 Wynik wdrożenia.....	53
ZAKOŃCZENIE.....	58
BIBLIOGRAFIA	60
WYKAZ ILUSTRACJI.....	62
WYKAZ TABEL	64

WSTĘP

W dzisiejszych czasach coraz większe znaczenie w dziedzinie IT zyskuje chmura obliczeniowa, która umożliwia elastyczne i skalowalne zarządzanie infrastrukturą informatyczną. Chmura ta pozwala na przechowywanie, przetwarzanie i udostępnianie danych z dowolnego miejsca na świecie, co wpływa na rozwój przedsiębiorstw oraz wspiera innowacyjność branży IT.

Jednym z kluczowych wyzwań związanych z zarządzaniem infrastrukturą w chmurze jest zapewnienie bezpieczeństwa, skalowalności oraz łatwości wdrażania nowych aplikacji. Dlatego coraz większą popularnością cieszą się narzędzia automatyzujące, które umożliwiają efektywne zarządzanie środowiskami chmurowymi oraz usprawniają procesy ich wdrożenia. Celem pracy jest zbudowanie narzędzia, które dzięki swojemu intuicyjnemu interfejsowi w postaci interaktywnego diagramu, zautomatyzuje proces tworzenia zaawansowanych systemów informatycznych wdrażanych w chmurze Azure.

Praca składa się z czterech części. Rozdział pierwszy skupia się na opisie technologii, które zostały wykorzystane w tytułowej pracy. W pierwszej części rozdziału omówione zostały podstawowe definicje i pojęcia związane z chmurą obliczeniową. Następnie opisano jedną z najpopularniejszych chmur obliczeniowych, to jest Microsoft Azure wraz z platformą Azure DevOps. Kolejno przedstawiono główne technologie użyte do wytworzenia aplikacji będącej przedmiotem niniejszej pracy.

Rozdział drugi dotyczy tematyki związanej z problemami napotykanymi przez programistów w ich codziennej pracy. Opisano w nim etapy procesu wytwarzania systemów informatycznych, zorientowanych na funkcjonowanie w chmurze Microsoft Azure z uwzględnieniem trudności związanych z ich konfiguracją.

Kolejna część pracy przedstawia założenia przyświecające tworzeniu tytułowej aplikacji. Opisano w niej architekturę systemu oraz wzorce projektowe użyte podczas jego budowania. Nadto, zaprezentowano infrastrukturę oraz uprawnienia systemu, a także działania prowadzące do jego inicjalizacji.

Przedmiotem czwartego rozdziału jest dokładny opis działania tytułowej aplikacji, poczynwszy od etapu tworzenia projektów, przez ich wdrażanie, kończąc na prezentacji wyników działania systemu.

ROZDZIAŁ 1. OPIS TECHNOLOGII

W niniejszym rozdziale została zaprezentowana definicja chmury obliczeniowej. Ponadto, wyszczególniono główne technologie użyte w ramach pracy. Pierwszoplanowym dostawcą technologii jest firma Microsoft, która jest twórcą chmury obliczeniowej Microsoft Azure, platformy Azure DevOps oraz środowiska programistycznego .NET. W projekcie aplikacji został użyty również framework¹ Angular, którego twórcą jest firma Google.

1.1 Definicja chmury obliczeniowej

Chmura obliczeniowa to model umożliwiający powszechny, wygodny i elastyczny dostęp do współdzielonych zasobów informatycznych, które mogą być szybko wdrażane i usuwane z minimalnym wysiłkiem zarządzania lub interakcji z dostawcą usług. Definicja ta została opracowana przez NIST² w publikacji „The NIST Definition of Cloud Computing” z 2011 roku³. Według tej definicji, chmura obliczeniowa musi spełniać pięć podstawowych cech:

- **samoobsługa na żądanie:** użytkownicy mogą samodzielnie wdrażać zasoby i je konfigurować bez potrzeby interakcji z ich dostawcą;
- **dostępność przez Internet:** użytkownicy mogą zarządzać zasobami chmurowymi wykorzystując Internet;
- **współdzielone zasoby:** wszystkie zasoby dostępne w chmurze są udostępniane dla wielu użytkowników ze wspólnej puli zasobów. Oznacza to, że wielu klientów może korzystać z usług pochodzących z tego samego fizycznego serwera;
- **elastyczność zasobów:** użytkownicy mogą w łatwy i szybki sposób skalować swoje zasoby według indywidualnych potrzeb;
- **mierzalność zasobów:** system chmury obliczeniowej automatycznie kontroluje i optymalizuje wykorzystanie zasobów oraz udostępnia konsumentom narzędzia do ich monitorowania, kontrolowania i raportowania.

¹ Framework – platforma programistyczna służąca do budowy aplikacji w narzucony odgórnie sposób.

² National Institute of Standards and Technology – Narodowy Instytut Standardów i Technologii.

³ P.Mell, T.Grance, , *The NIST Definition of Cloud Computing, 2011*, w:
<https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf>, [dostęp 19.02.2023].

W definicji przygotowanej przez NIST można znaleźć również opis trzech modeli usług:

- **oprogramowanie jako usługa, SaaS** (z ang. Software as a Service): model ten polega na korzystaniu z aplikacji dostarczanych przez usługodawcę, uruchamianych na infrastrukturze chmury obliczeniowej. Aplikacje te są dostępne z różnych urządzeń klienckich przy wykorzystaniu Internetu. Konsument nie zarządza ani nie kontroluje infrastruktury (np. serwery, sieci, systemy operacyjne) ani nawet poszczególnych możliwości aplikacji z wyjątkiem specyficznych ustawień konfiguracyjnych dla konkretnych użytkowników;
- **platforma jako usługa, PaaS** (z ang. Platform as a Service): w tym modelu dostawca umożliwia klientowi wdrażanie w infrastrukturze chmurowej własnych lub nabytych przez niego aplikacji. Klient nie zarządza ani nie kontroluje infrastruktury chmurowej, natomiast ma kontrolę nad wdrożonymi aplikacjami i ustawieniami konfiguracyjnymi środowiska hostingowego dla aplikacji;
- **infrastruktura jako usługa, IaaS** (z ang. Infrastructure as a Service): jest to model, który pozwala konsumentowi wdrażać zasoby odpowiedzialne za przetwarzanie, przechowywanie, sieci oraz inne podstawowe zasoby obliczeniowe. Konsument nie zarządza infrastrukturą bazową chmury, natomiast ma kontrolę nad systemami operacyjnymi, pamięcią masową, wdrożonymi aplikacjami i poszczególnymi komponentami sieciowymi.

Kolejnym aspektem poruszonym w definicji chmury jest model wdrażania. NIST wyszczególnił cztery modele:

- **chmura prywatna** – jest udostępniona wyłącznie jednej organizacji, która może składać się z wielu konsumentów (np. różne departamenty przedsiębiorstwa). Chmura ta jest zarządzana i obsługiwana przez samą organizację lub firmy zewnętrzne. Może znajdować się na terenie organizacji lub poza nią;
- **chmura społecznościowa** – przeznaczona jest dla określonej grupy użytkowników z różnych organizacji, którzy mają wspólne potrzeby. Chmura zarządzana jest przez jedną lub kilka organizacji z tej społeczności lub przez zewnętrznego dostawcę usług. Może znajdować się na terenie organizacji lub poza nią;

- **chmura publiczna** – jest przeznaczona do ogólnego użytku przez wszystkich konsumentów. Może być zarządzana i obsługiwana przez prywatne, akademickie lub rządowe organizacje. Znajduje się ona na terenie dostawcy;
- **chmura hybrydowa** – jest połączeniem dwóch lub więcej różnych infrastruktur chmurowych (prywatnych, społecznościowych lub publicznych), które pozostają odrębnymi jednostkami, ale są ze sobą powiązane za pomocą technologii umożliwiających przenoszenie danych i aplikacji.

1.2 Microsoft Azure

Microsoft Azure to jedna z największych i najpopularniejszych platform chmury obliczeniowej na rynku, oferująca wiele różnorodnych usług i narzędzi dla organizacji i użytkowników indywidualnych. Chmura Microsoft Azure została oficjalnie wprowadzona na rynek w lutym 2010 roku⁴, a jej wydajność i innowacyjność w krótkim czasie przyczyniły się do zdobycia znaczącej pozycji w branży chmurowej. Według raportu Canalys, Microsoft Azure zajmuje drugie miejsce w rankingu największych dostawców usług chmurowych na świecie, z udziałem w rynku wynoszącym około 22% (stan na ostatni kwartał roku 2021)⁵.

Microsoft Azure oferuje szeroki zakres usług i narzędzi dla organizacji, takich jak przetwarzanie danych w chmurze, przechowywanie danych, sztuczną inteligencję i uczenie maszynowe, usługi analityczne, wirtualizację, rozwiązania IoT⁶ oraz usługi sieciowe i bezpieczeństwa.

W tytułowej pracy użyto niewielką ilość usług chmury Microsoft Azure. Opis każdego z nich przedstawia tabela Tab. 1.1.

Tab. 1.1 Opis użytych usług chmury Microsoft Azure

Usługa	Opis
Azure Active Directory	Usługa zarządzania tożsamością i dostępem w chmurze. Azure Active Directory umożliwia zarządzanie

⁴ Windows Azure Platform Now Generally Available in 21 Countries, w : <https://azure.microsoft.com/en-us/blog/windows-azure-platform-now-generally-available-in-21-countries/>, [dostęp: 09.03.2023].

⁵ Canalys, *Global cloud services spend exceeds US\$50 billion in Q4 2021*, w: <https://www.canalys.com/newsroom/global-cloud-services-Q4-2021>, [dostęp: 08.03.2023].

⁶ IoT – Internet of things – sieć połączonych ze sobą różnych urządzeń za pomocą Internetu.

	użytkownikami, grupami i aplikacjami, zapewniając bezpieczny sposób uwierzytelniania oraz autoryzacji ⁷ .
Resource Group	Jest to kontener, która umożliwia organizowanie i zarządzanie powiązаныmi ze sobą zasobami ⁸ .
App Service	Usługa hostingowa dla różnych rodzajów aplikacji webowych opartych o protokół HTTP ⁹ napisanych w różnych językach programowania (np. .NET, Node.js, Python, Java) ¹⁰ .
App Service Plan	Usługa, która jest wymagana do utworzenia zasobów <i>App Service</i> i jest odpowiedzialna za ich skalowanie, wysoką dostępność, system operacyjny i zasoby obliczeniowe ¹¹ .
Application Insights	Usługa, która umożliwia śledzenie i analizowanie zachowania aplikacji internetowych. Dostarcza informacji o wydajności, użyteczności i awariach aplikacji ¹² .
Azure SignalR	Usługa, która umożliwia komunikację w czasie rzeczywistym pomiędzy aplikacjami internetowymi za pośrednictwem protokołu HTTP. Usługa zapewnia skalowalność, niezawodność i niskie opóźnienia w przesyłaniu komunikatów, co umożliwia tworzenie zaawansowanych aplikacji ¹³ .
<i>Azure Service Bus</i>	Usługa, która umożliwia asynchroniczną komunikację między aplikacjami i usługami w chmurze poprzez

⁷ Microsoft, *Co to jest usługa Azure Active Directory*, w: <https://learn.microsoft.com/pl-PL/azure/active-directory/fundamentals/active-directory-what-is>, [dostęp: 11.02.2023].

⁸Microsoft, *Zarządzanie grupami zasobów platformy Azure przy użyciu witryny Azure Portal*, w: <https://learn.microsoft.com/pl-pl/azure/azure-resource-manager/management/manage-resource-groups-portal>, [dostęp: 15.02.2023].

⁹ HTTP- Hypertext Transfer Protocol.

¹⁰ Microsoft, *Omówienie usługi App Service*, w: <https://learn.microsoft.com/pl-pl/azure/app-service/overview>, [dostęp: 01.03.2023].

¹¹Microsoft, *Plan usługi Azure App Service — omówienie*, w: <https://learn.microsoft.com/pl-pl/azure/app-service/overview-hosting-plans>, [dostęp: 26.02.2023].

¹² Microsoft, *Omówienie usługi Application Insights*, w: <https://learn.microsoft.com/pl-pl/azure/azure-monitor/app/app-insights-overview>, [dostęp: 26.02.2023].

¹³ Microsoft, *Co to jest usługa Azure SignalR Service?*, w: <https://learn.microsoft.com/en-us/azure/azure-signalr/signalr-overview>, [dostęp: 15.03.2023].

	wysyłanie komunikatów na kolejki (queues) lub tematy (topics) ¹⁴ .
<i>Azure Cosmos DB</i>	Usługa nierelacyjnych baz danych, która umożliwia przechowywanie, skalowanie oraz szybki dostęp do różnych typów danych w aplikacjach chmurowych ¹⁵ .
<i>Azure Key Vault</i>	Usługa, która umożliwia bezpieczne przechowywanie i zarządzanie kluczami, hasłami, certyfikatami używanymi przez aplikacje oraz usługi w chmurze ¹⁶ .

Oprócz szerokiego zakresu usług i narzędzi, które oferuje Microsoft Azure, platforma ta posiada również wiele sposobów na wdrażanie zasobów do chmury. Poniżej przedstawiono kilka z nich:

- **Azure Portal** - narzędzie webowe, które umożliwia zarządzanie zasobami przy użyciu interfejsu graficznego. Portal umożliwia łatwe tworzenie i konfigurowanie zasobów. Umożliwia również monitorowanie zasobów w czasie rzeczywistym¹⁷. Rys. 1.1 przedstawia sposób tworzenia Resource Group z poziomu Azure Portal.
- **Azure Command-Line Interface (Azure CLI)** - jest to interfejs wiersza poleceń, działający na różnych platformach, takich jak Windows, Linux i macOS¹⁸. Przykładowa komenda tworząca Resource Group wygląda następująco:

```
az group create --name "Resource-Group-Name" --location "westeurope" .
```

- **Azure PowerShell** - jest kolejnym po Azure CLI interfejsem wiersza poleceń przy czym oparty jest na języku PowerShell¹⁹. Przykładowa komenda tworząca Resource Group wygląda następująco:

```
New-AzResourceGroup -Name "Resource-Group-Name" -Location "westus" .
```

¹⁴ Microsoft, *Co to jest Azure Service Bus*, w: <https://learn.microsoft.com/pl-pl/azure/service-bus-messaging/service-bus-messaging-overview>, [dostęp: 01.03.2023].

¹⁵ Microsoft, *Azure Cosmos DB — Zapraszamy!*, w: <https://learn.microsoft.com/pl-pl/azure/cosmos-db/introduction>, [dostęp: 01.03.2023].

¹⁶ Microsoft, *Informacje o usłudze Azure Key Vault*, w: <https://learn.microsoft.com/pl-pl/azure/key-vault/general/overview>, [dostęp: 01.03.2023].

¹⁷ Microsoft, *Co to jest Azure Portal?*, w: <https://learn.microsoft.com/pl-pl/azure/azure-portal/azure-portal-overview>, [dostęp: 01.03.2023].

¹⁸ Microsoft, *Dokumentacja interfejsu wiersza poleceń platformy Azure*, w: <https://learn.microsoft.com/pl-pl/cli/azure/>, [dostęp: 01.02.2023].

¹⁹ Microsoft, *Rozpoczynanie pracy z programem Azure PowerShell*, w: <https://learn.microsoft.com/en-us/powershell/azure/get-started-azureps?view=azps-9.4.0>, [dostęp: 01.02.2023].

- **Azure REST API** - interfejs, który pozwala na zarządzanie zasobami bezpośrednio poprzez protokół HTTP²⁰.

The screenshot shows the 'Create a resource group' page in the Microsoft Azure portal. The 'Basics' tab is active. The 'Project details' section includes a 'Subscription' dropdown set to 'Azure dla studentów' and a 'Resource group' text box containing 'TestResourceGroup'. The 'Resource details' section includes a 'Region' dropdown set to '(Europe) West Europe'. At the bottom, there are three buttons: 'Review + create', '< Previous', and 'Next: Tags >'.

Rys. 1.1 Tworzenie Resource Group z poziomu Portalu Azure

1.3 Azure DevOps

Azure DevOps to platforma służąca do zarządzania projektami, procesami tworzenia oraz testowania oprogramowania oferowana przez firmę Microsoft. Wiodącą ideą całej platformy jest zwiększenie efektywności zespołów programistycznych podczas tworzenia, rozwijania i wdrażania oprogramowania²¹. Głównymi częściami składowymi tej platformy są:

- **Azure Boards** – służący do organizacji projektów, zadań oraz planowania przebiegu procesu tworzenia aplikacji;
- **Azure Pipelines** – mający za zadanie budowanie oraz wdrażanie oprogramowania przy wykorzystaniu procesu CI/CD (z ang. Continuous Integration/Continuous Deployment)²²;
- **Azure Repos** – przestrzeń, w której tworzone są repozytoria kodu Git²³;

²⁰ Microsoft, *Azure REST API reference*, w: <https://learn.microsoft.com/en-us/rest/api/azure/>, [dostęp: 01.03.2023]

²¹ Microsoft, *Azure DevOps*, w: <https://azure.microsoft.com/pl-pl/products/devops/>, [dostęp 16.01.2023].

²² Proces tworzenia oprogramowania, którego głównym celem jest jego ciągła integracja z centralnym repozytorium kodu oraz jego automatyczne kompilowanie, testowanie i wdrażanie.

²³ Git – rozproszony system kontroli wersji.

- **Azure Test Plans** – służące do przygotowywania oraz wykonywania scenariuszy testowych aplikacji;
- **Azure Artifacts** – mające na celu zarządzanie pakietami, bibliotekami i zależnościami pomiędzy aplikacjami.

1.4 Szablony ARM

W tytułowej pracy automatyzacje wdrożeń oparto na szablonach usługi ARM (Azure Resource Manager). Szablony ARM to pliki JSON²⁴, które pozwalają na zdefiniowanie i wdrażanie zasobów w sposób powtarzalny. Pozwalają one na określenie wszystkich parametrów potrzebnych do utworzenia zasobów, dzięki czemu można łatwo tworzyć i zarządzać środowiskami chmury w sposób zautomatyzowany²⁵.

Szablony ARM składają się z kilku sekcji, w tym:

- **Parameters** - w tej sekcji określa się wartości parametrów, które mogą być przekazane do szablonu podczas jego wdrażania. Parametry te mogą określać między innymi takie elementy jak nazwy zasobów, konfiguracje sieciowe, czy poziomy usług;
- **Variables** – sekcja służy do zdefiniowania zmiennych, które będą używane w szablonie np. wartości konfiguracyjne, adresy IP, numery portów;
- **Resources** – w tym fragmencie opisuje się zasoby, które mają być utworzone lub zaktualizowane. Zasoby te mogą być powiązane z innymi zasobami, co pozwala na określenie ich hierarchii i zależności;
- **Outputs** - w tej sekcji można zdefiniować wartości, które mają być zwrócone po wdrożeniu szablonu. Może to być np. adres URL serwera, klucz dostępu czy adres IP.

²⁴ JSON – JavaScript Object Notification.

²⁵ Microsoft, Co to są szablony usługi ARM?, w: <https://learn.microsoft.com/pl-pl/azure/azure-resource-manager/templates/overview>, [dostęp: 15.03.2023].

```

{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "name": {
      "type": "String"
    },
    "sku": {
      "type": "String",
      "allowedValues": ["Standard", "Basic"]
    }
  },
  "variables": {},
  "resources": [
    {
      "type": "Microsoft.ServiceBus/namespaces",
      "apiVersion": "2021-06-01-preview",
      "name": "[parameters('name')]",
      "location": "[resourceGroup().location]",
      "sku": {
        "name": "[parameters('sku')]",
        "tier": "[parameters('sku')]"
      },
      "properties": {
        "disableLocalAuth": false,
        "zoneRedundant": false
      }
    }
  ]
}

```

Rys. 1.2 Przykładowy szablon ARM

Szablon ARM przedstawiony na Rys.1.2 służy do wdrożenia zasobu *Azure Service Bus*. W szablonie wyszczególniono dwa parametry „name” oraz „sku”, które muszą być dostarczone w oddzielnym pliku i służą do określenia nazwy zasobu oraz jego warstwy cenowej.

1.5 .NET 6

W tytułowej pracy do stworzenia aplikacji serwerowych zdecydowano się na użycie darmowej platformy programistycznej .NET 6, która umożliwia tworzenie aplikacji webowych, desktopowych oraz mobilnych działających na wielu systemach operacyjnych takich jak Windows, Linux, macOS. Platforma .NET 6 umożliwia programowanie w wielu językach takich jak C#, F# czy Visual Basic²⁶. Platforma udostępnia wiele gotowych szablonów kodu, które przeznaczone są do wykonywania określonych zadań. W przedmiotowej pracy zostały użyte dwa z nich:

- **ASP.NET Core Web API** - jest szablonem kodu, służącym do budowania aplikacji webowych, które komunikują się z klientami za pomocą protokołu HTTP. W przedmiotowej pracy szablon ten został użyty jako warstwa komunikacyjna pomiędzy aplikacją kliencką a serwerową;

²⁶ Microsoft, *Co to jest platforma .NET? Wprowadzenie i omówienie*, w: <https://learn.microsoft.com/pl-pl/dotnet/core/introduction>, [dostęp: 15.03.2023].

- **Worker Service** – szablon umożliwiający tworzenie aplikacji działających w tle, które wykonują określone, często długotrwałe zadania niewymagające interakcji z użytkownikiem. W pracy szablon ten został użyty do wykonywania wdrożeń zasobów oraz kodu do chmury Azure.

1.6 .NET Azure SDK

.NET Azure SDK to zestaw bibliotek programistycznych służących do zarządzania zasobami chmury Azure z poziomu platformy .NET. Biblioteki mają za zadanie uprościć pracę programistom, którzy dzięki ich użyciu nie muszą wykonywać bezpośrednich zapytań HTTP ze swoich aplikacji aby zarządzać chmurą Azure. Spośród wielu możliwości udostępnionych przez .NET Azure SDK w tytułowej pracy skorzystano z funkcji pozwalających na wdrożenie szablonów ARM, tworzenie grupy zasobów, tworzenie kluczy w usłudze Azure Key Vault czy odczytywanie poświadczeń uwierzytelniania dla wdrożonych usług²⁷. Na Rys. 1.3 przedstawiono wdrożenie szablonu ARM do chmury Azure za pomocą .NET Azure SDK. Funkcja wdrożeniowa przyjmuje jako parametr obiekt typu `ArmDeploymentProperties`, który posiada właściwości opisujące szablon oraz parametry. Konstruktor obiektu przyjmuje parametr określający tryb wdrożenia.

```
var deploymentContent = new ArmDeploymentContent(new ArmDeploymentProperties(ArmDeploymentMode.Incremental)
{
    Template = BinaryData.FromString(resourceDeployment.Template),
    Parameters = BinaryData.FromString(JObject.FromObject(resourceDeployment.Parameters).ToString())
});

var result = await armDeploymentCollection.CreateOrUpdateAsync(
    WaitUntil.Completed,
    resourceDeployment.Name,
    deploymentContent, ct);
```

Rys. 1.3 Wdrażanie szablonu ARM z poziomu .NET Azure SDK

1.7 Angular

Do zbudowania aplikacji klienckiej tytułowej pracy został wybrany framework Angular, który jest używany do budowania aplikacji webowych typu SPA²⁸ w języku TypeScript, który jest nadzbiorem języka JavaScript. Angular opiera się na konstrukcji komponentowej, co oznacza, że aplikacja budowana jest z wielu niezależnych komponentów.

²⁷ Microsoft, *Omówienie zestawu Azure SDK dla platformy .NET*, w: <https://learn.microsoft.com/pl-pl/dotnet/azure/sdk/azure-sdk-for-dotnet>, [dostęp: 15.03.2023].

²⁸ Single Page Application – aplikacje webowe, które nie wymagają przeładowania całej strony po każdej interakcji użytkownika.

Każdy z komponentów składa się z kodu TypeScript, szablonu HTML i stylów CSS, a także posiada swój własny stan i cykl życia. Angular posiada mechanizm dwukierunkowego wiązania danych, który umożliwia automatyczną synchronizację danych pomiędzy modelem zapisanym w TypeScript a widokiem napisanych w HTML²⁹.

²⁹ Google, *What is Angular?*, w: <https://angular.io/guide/what-is-angular>, [dostęp: 13.03.2023].

ROZDZIAŁ 2. OPIS PROBLEMU

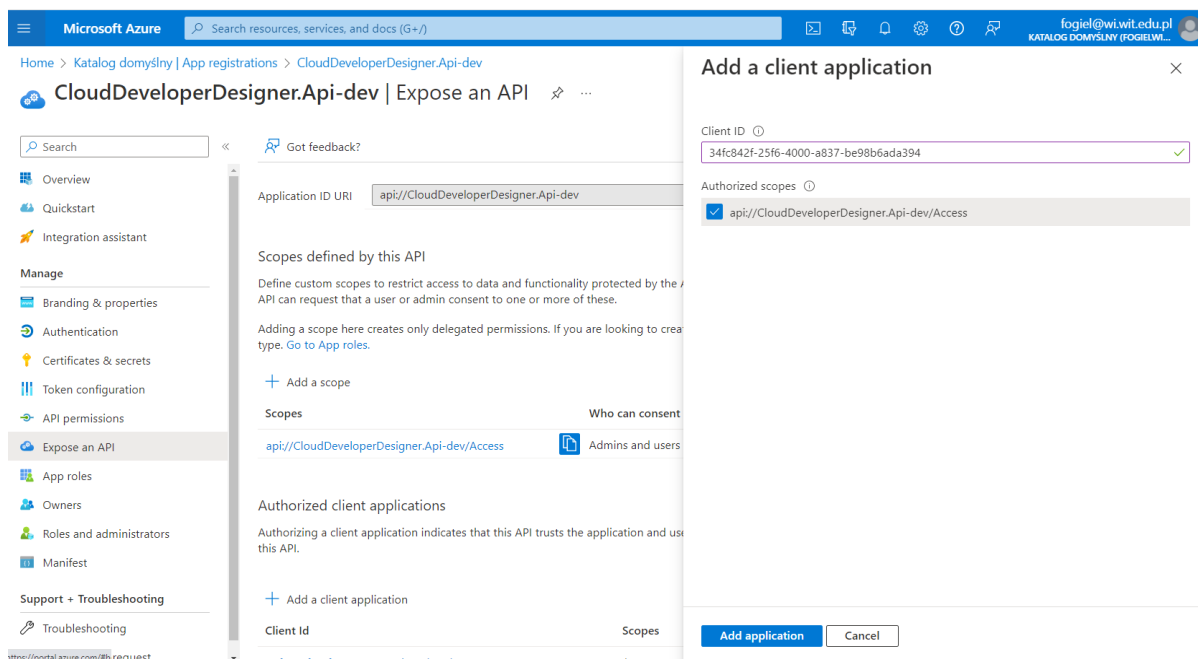
Firmy, które za cel obrały sobie cyfryzację swoich procesów biznesowych potrzebują dużej ilości spersonalizowanych aplikacji, które proces ten przyspieszą i dadzą pracownikom możliwość wykonywania powierzonych zadań w prosty sposób. Wytworzenie oprogramowania, które ma wspomagać pracę biznesu, oprócz zaimplementowania odpowiednich algorytmów działania, wymaga również użycia usług udostępnionych przez chmurę obliczeniową. Usługi te potrzebne są między innymi do zapisywania danych wprowadzanych przez użytkowników, kolejkowania zadań czy wysyłania powiadomień. Aplikacje te korzystają również z usług chmurowych, które udostępniają procesy uwierzytelniania oraz autoryzacji potrzebnych do ochrony danych przed osobami niepożądanymi. Czas potrzebny na wytworzenie takich aplikacji wydłuża się ze względu na dużą ilość często powtarzających się czynności związanych z tworzeniem ich tożsamości, wdrażaniem odpowiednich usług chmurowych i zapisaniem ich kluczy dostępowych. Nadto, należy również uwzględnić czynności związane z tworzeniem odpowiedniego środowiska programistycznego składającego się z repozytorium kodu i procesami jego budowania. W tym rozdziale przedstawiono problemy, z jakimi spotykają się programiści podczas wykonywania wyżej wymienionych procesów, które zostaną zautomatyzowane w przedmiotowej pracy.

2.1 Rejestrowanie aplikacji w usłudze Azure Active Directory

Proces tworzenia aplikacji, do których dostęp ma być zabezpieczony poprzez usługę AAD³⁰ wymaga od programistów wykonania wielu nieskomplikowanych lecz powtarzalnych kroków. Po pierwsze, dla dużych systemów wytworzonych w architekturze mikro-usług³¹ należy podać informacje na temat zależności pomiędzy modułami. Ten krok jest najbardziej czasochłonny i błędogeny, ponieważ wymaga od programisty podania unikalnego identyfikatora aplikacji, która może uzyskać dostęp do powiązanego modułu. Na Rys. 2.1 można zauważyć iż programista musi ręcznie dodać identyfikator aplikacji (*ClientId*), który wcześniej został wyszukany i skopiowany. Niestety, aktualny interfejs Portalu Azure nie udostępnia wyszukiwania nazwy aplikacji, co znacząco wydłuża proces tworzenia zależności jak i zarządzania nimi.

³⁰ AAD – Azure Active Directory

³¹ Architektura nastawiona na dzielenie systemów informatycznych na wiele modułów, które mają swoją unikalną odpowiedzialność.



Rys. 2.1 Proces dodawania autoryzowanych aplikacji

W przypadku aplikacji serwerowych, które komunikują się z innymi aplikacjami, wymagane jest również utworzenie klucza aplikacji - *Client secrets*, potrzebnego w procesie uwierzytelniania komunikacji pomiędzy nimi. Klucz ten jest tajny dlatego rekomendowane jest zapisanie go np. w usłudze *Azure Key Vault*. Ponadto klucz ten jest widoczny tylko podczas jego tworzenia, dlatego niezapisanie go powoduje jego utratę. Proces ten, choć nie powinien sprawić nikomu problemu, jest powtarzalny i czasochłonny. Warto wspomnieć również o tym, iż w przypadku niepoprawnego zapisania klucza komunikacja pomiędzy modułami nie będzie możliwa, a proces odkrycia problemu w skrajnych przypadkach może kosztować nawet kilka godzin. Na Rys. 2.2 przedstawiono przykład wygenerowanego klucza dla aplikacji serwerowej w usłudze AAD a na Rys. 2.3 jego zapis w usłudze *Azure Key Vault*.

Microsoft Azure | Search resources, services, and docs (G+)

Home > Katalog domyślny | App registrations > CloudDeveloperDesigner.Api-dev

CloudDeveloperDesigner.Api-dev | Certificates & secrets

Search | Got feedback?

Credentials enable confidential applications to identify themselves to the authentication service when receiving tokens at a web addressable location (using an HTTPS scheme). For a higher level of assurance, we recommend using a certificate (instead of a client secret) as a credential.

Application registration certificates, secrets and federated credentials can be found in the tabs below.

Certificates (0) **Client secrets (1)** Federated credentials (0)

A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.

+ New client secret

Description	Expires	Value	Secret ID
CloudDeveloperDesigner.Api-devSecret	12/23/2024	tEO*****	1b7849a7-632c-4798-9b92-69e624f9241b

Rys. 2.2 Wygenerowany klucz aplikacji

Home > Resource groups > rg-clouddeveloperdesigner-dev > kv-clouddevdesigner-dev | Secrets > AzureAd--ClientSecret

1ec9513f814e46788be2e7267f412aeb

Secret Version

Settings

Set activation date ☐

Set expiration date ☐

Enabled Yes No

Tags 0 tags

Secret

Content type (optional)

[Hide Secret Value](#)

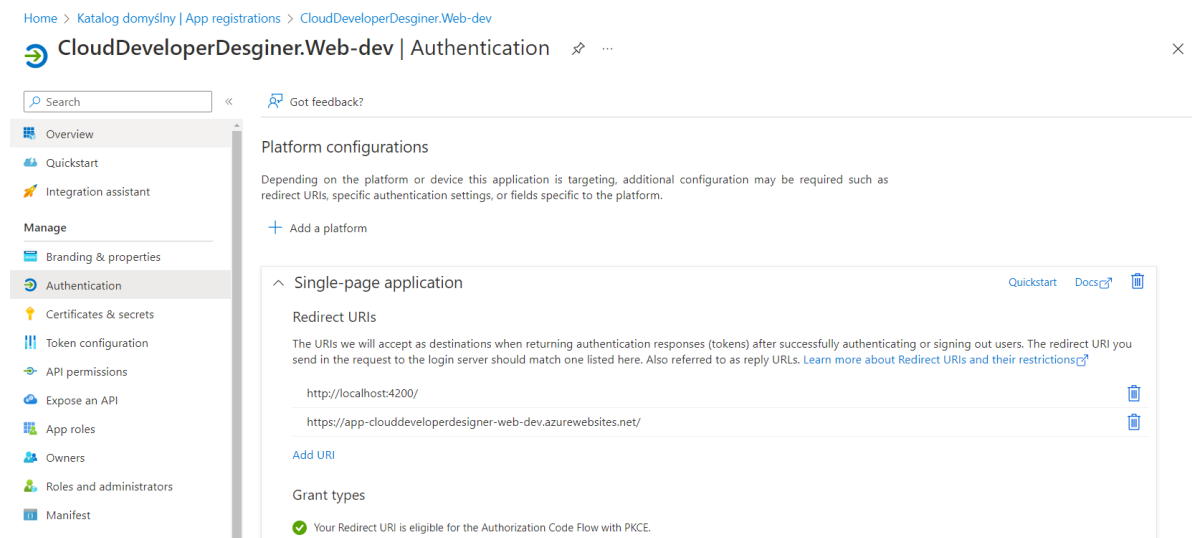
Secret value tEO8Q~iFgE4g1EMG9IABINPOrvitDuymxm_o9buj

[Apply](#) [Discard changes](#) [Close](#)

[Give feedback](#)

Rys. 2.3 Zapisany klucz aplikacji w usłudze Key Vault

W przypadku aplikacji klienckich, które wymagają logowania poprzez usługę AAD, należy również podać adresy przekierowań, to jest *Redirect URIs*, które wykorzystywane są do powrotu do aplikacji po poprawnym uwierzytelnieniu użytkownika (Rys. 2.4).



Rys. 2.4 Adresy przekierowań dla aplikacji typu SPA

Jak można zauważyć na powyższym przykładzie, aplikacja środowiska deweloperskiego posiada dwa adresy – localhost na potrzeby lokalnego uruchamiania oraz adres hosta aplikacji na potrzeby testowania po wdrożeniu na serwer. W przypadku wyższych środowisk (np. testowe, produkcyjne) podaje się tylko adres serwera. Jest to kolejny krok, który musi zostać poprawnie wykonany przez programistę, bez którego uwierzytelnianie użytkowników nie będzie działało poprawnie. Warto również zaznaczyć, iż każda z wyżej wymienionych czynności musi zostać powtórzona dla każdego środowiska. Proces tworzenia aplikacji wymaga co najmniej trzech środowisk: deweloperskie, testowe, produkcyjne.

2.2 Proces tworzenia zasobów w chmurze Azure

Kolejnym procesem, który jest powtarzalny oraz czasochłonny, jest tworzenie zasobów. Pierwszym krokiem w owym procesie jest utworzenie grupy zasobów (z ang. Resource Group). Dobrą praktyką przyjętą przez wiele firm jest tworzenie grupy zasobów dla każdego środowiska aplikacji, co implikuje utworzenie co najmniej trzech grup. W każdej grupie programista musi utworzyć zestaw prawie identycznych zasobów, które w większości przypadku różnią się od siebie tylko nazwą. Nazewnictwo zasobów jest istotną kwestią w kontekście zarządzania, wyszukiwania oraz utrzymywania chmury obliczeniowej. W dokumentacji³² udostępnionej przez firmę Microsoft można znaleźć proponowane schematy nazewnictwa. Dla przykładu dostawca chmury proponuje, by nazwa zasobu była zgodna ze schematem :

³² Microsoft, *Definiowanie konwencji nazewnictwa*, w: <https://learn.microsoft.com/pl-pl/azure/cloud-adoption-framework/ready/azure-best-practices/resource-naming>, [dostęp: 16.03.2023]

{typ-zasobu}-{nazwa-aplikacji}-{środowisko}-{region}-{instancja}

W przypadku np. usługi *Azure Service Bus* mogłoby to wyglądać następująco:

sb-clouddeveloperdesigner-prod-westeuropa-001

Problemem w tym przypadku jest jednak egzekwowanie danego schematu. Pracownik tworzący dany zasób musi znać konwencje nazewnictwa a następnie ją zastosować.

Podczas tworzenia zasobów programista nierzadko musi również wypełnić dużą ilość parametrów, które często wymagają specjalistycznej wiedzy między innymi z takich dziedzin jak bezpieczeństwo czy sieci komputerowe. W przypadku braku specjalistycznej wiedzy programista musi poprosić o wsparcie innych specjalistów lub spędzić dodatkowy czas na dokształcanie się, co znacząco wydłuża proces tworzenia zasobów. Dla przykładu na Rys. 2.5 przedstawiono parametry, które należy wypełnić aby skonfigurować sieć wirtualną dla usługi *Storage Account*.

Home > Storage accounts > Create a storage account

Basics Advanced **Networking** Data protection

Network connectivity

You can connect to your storage account either publicly, via private endpoint.

Network access *

☐ Enable public access

☒ Enable private endpoint

☐ Disable public access

Create virtual network

Location * East US

Address space

The virtual network's address space specified as one or more address prefixes in CIDR notation (e.g. 10.0.0.0/16).

Address range	Addresses	Overlap
<input type="checkbox"/> 10.0.0.0/16	10.0.0.4 - 10.0.255.254 (65531 addresses)	None
<input type="text"/>	(0 Addresses)	None

Subnets

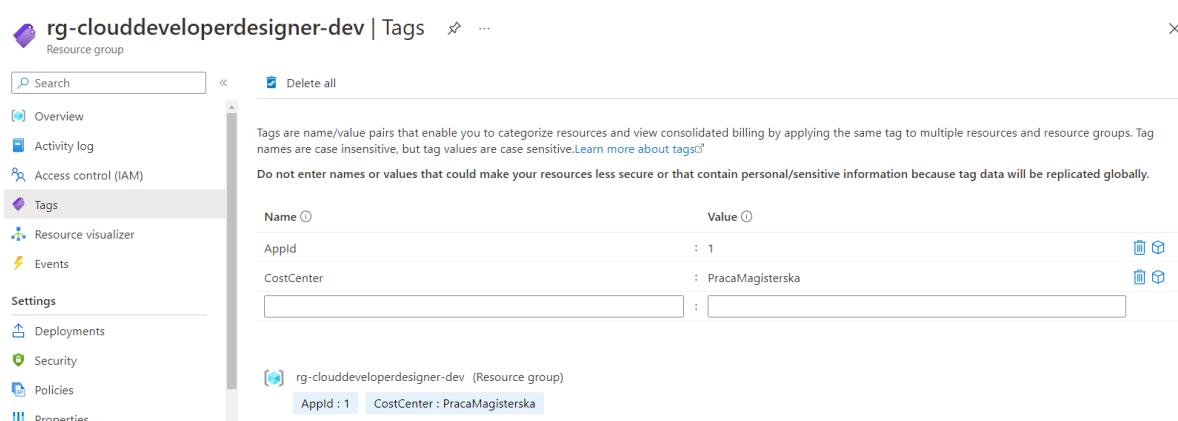
The subnet's address range in CIDR notation (e.g. 10.0.0.0/24). It must be contained by the address space of the virtual network.

Subnet name	Address range	Addresses
<input type="checkbox"/> default	10.0.0.0/24	10.0.0.4 - 10.0.0.254 (251 addresses)
<input type="text"/>	<input type="text"/>	(0 Addresses)

Review < Previous Next OK Discard

Rys. 2.5 Przykładowa konfiguracja sieci wirtualnej dla usługi *Storage account*

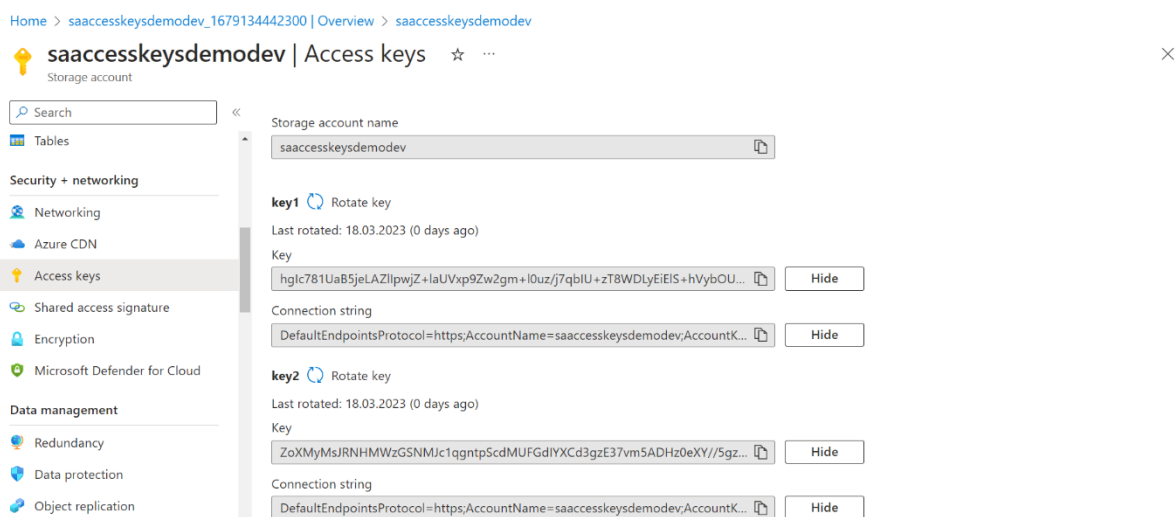
Chmura Microsoft Azure udostępnia również dodawanie metadanych do utworzonych zasobów poprzez wykorzystanie funkcji *Tags*³³. Tag to obiekt reprezentowany poprzez nazwę oraz wartość służący do organizacji zasobów poprzez na przykład przypisanie ich do danej jednostki operacyjnej organizacji lub podanie wewnętrznego kodu przypisanego do zasobu. Dodawanie odpowiednich metadanych do każdego zasobu może być wymagane przez organizację, co nakłada kolejne obowiązki na ich twórcę. Rys. 2.6 prezentuje tagi, dodane do zasobu Resource Group.



Rys. 2.6 Przykładowe tagi dla grupy zasobów

Następnym krokiem podczas tworzenia zasobów jest zapisywanie ich kluczy dostępowych do usługi *Azure Key Vault*. W tym celu programista musi odczytać dany klucz z Portalu Azure, a następnie skopiować go i zapisać. Proces ten zajmuje stosunkową dużą ilość czasu, ponieważ skomplikowane systemy mogą zawierać nawet więcej niż dziesięć zasobów w każdej grupie. Ponadto, nazwa klucza musi być identyczna w każdym środowisku oraz musi być zsynchronizowana z aktualną implementacją kodu, który te klucze pobiera. Na Rys. 2.7 przedstawiono przykładowe klucze dostępowe dla usługi *Azure Storage Account*.

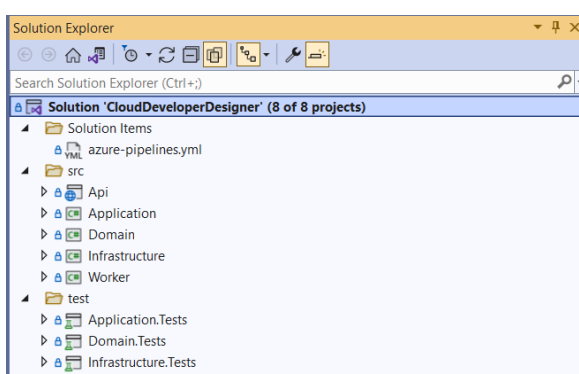
³³ Microsoft, *Organizowanie zasobów platformy Azure i hierarchii zarządzania przy użyciu tagów*, w: <https://learn.microsoft.com/pl-pl/azure/azure-resource-manager/management/tag-resources?tabs=json>, [dostęp: 21.01.2023]



Rys. 2.7 Przykładowe klucze dostępowe do usługi Azure Storage Account

2.3 Proces tworzenia środowiska programistycznego

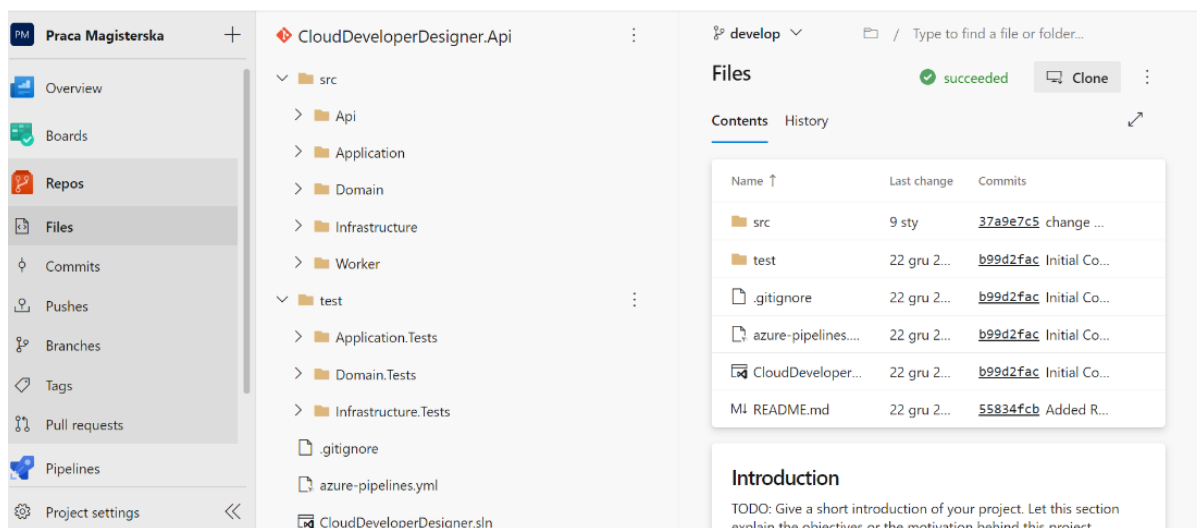
Oprócz prac typowo związanych z infrastrukturą, programista musi również rozpocząć pracę nad kodem źródłowym aplikacji. W tym celu tworzona jest solucja w wybranym przez programistę IDE (z ang. Integrated Development Environment)³⁴, która zawiera projekty. Schemat powiązania projektów w solucji jest często narzucony przez wybraną architekturę oprogramowania. Każda z solucji powinna być podzielona na projekty zawierającą implementacje funkcjonalności, jak i projekty służące do testowania napisanego kodu. Opisana powyżej czynność jest powtarzalna w zależności od ilości mikro-usług składających się na system. Rys. 2.8 prezentuje takową solucję podzieloną na poszczególne projekty.



Rys. 2.8 Przykładowa solucja kodu źródłowego

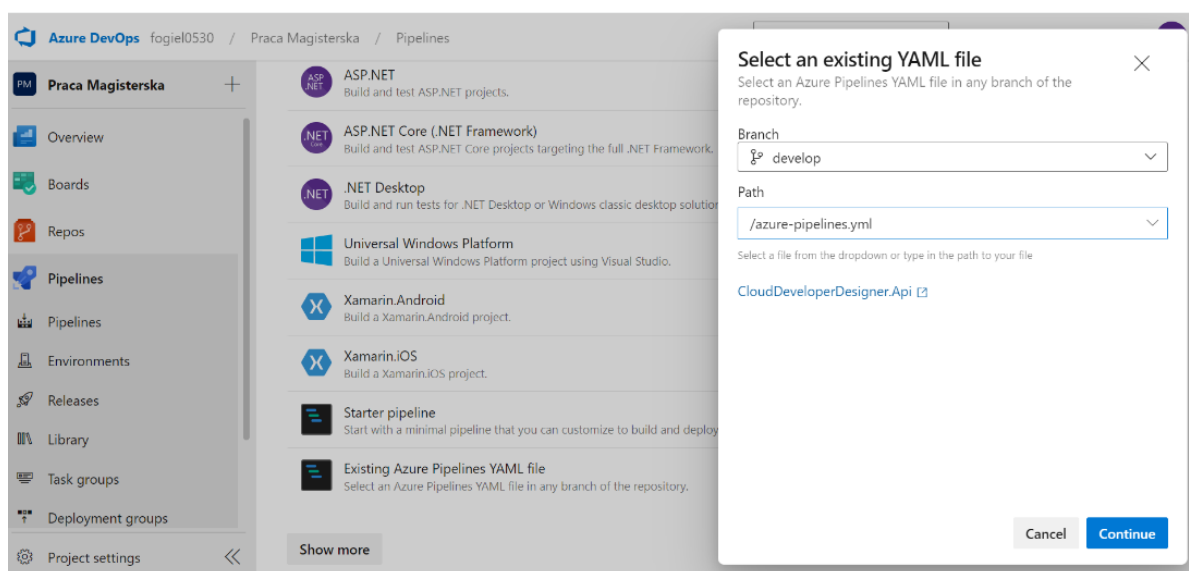
³⁴ Zintegrowane środowisko programistyczne służące do tworzenia, testowania i uruchamiania kodu źródłowego.

Po utworzeniu solucji należy ją zintegrować z repozytorium kodu, które zostało utworzone w usłudze Azure DevOps, czego wizualizację prezentuje Rys. 2.9.



Rys. 2.9 Utworzone repozytorium kodu w usłudze Azure DevOps

Aby zapewnić pełną realizację procesu CI/CD w usłudze należy również przygotować proces kompilowania, testowania oraz wdrażania kodu. Instrukcje opisujące te czynności zapisane są w plikach YAML, które są integralną częścią solucji. Proces budowania kodu w oparciu o ten plik widoczny jest na Rys. 2.10.



Rys. 2.10 Tworzenie procesu budowania kodu na podstawie pliku YAML

ROZDZIAŁ 3. PROJEKT APLIKACJI

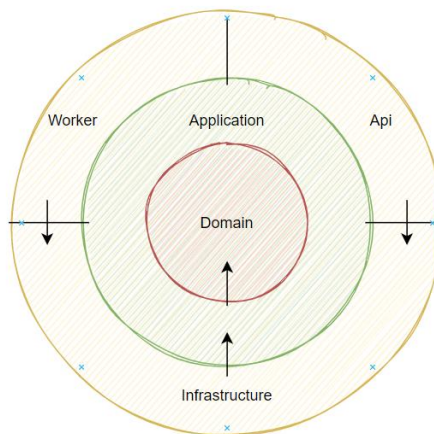
Wychodząc naprzeciw problemom opisanym w poprzednim rozdziale wykreowano aplikację *Cloud Developer Designer*, która w znaczącym sposób odciąża programistów od wykonywania powtarzalnych czynności lub od nauki specjalnych narzędzi automatyzujących. Aplikacja ma na celu przede wszystkim zaoszczędzenie jak największej ilości czasu potrzebnego na wytworzenie nowej aplikacji bez potrzeby angażowania administratorów oraz bez potrzeby szkolenia programistów w kierunku administracji zasobami. Poprzez swój intuicyjny interfejs w postaci interaktywnego diagramu programista w łatwy sposób jest w stanie utworzyć infrastrukturę aplikacji wraz z gotowym szablonem jej kodu bez potrzeby uruchamiania skryptów czy eksploracji Portal Azure lub Azure DevOps. Aplikacja wymaga jednorazowego przygotowania plików konfiguracyjnych w postaci szablonów ARM oraz dostarczenia listy dozwolonych lokalizacji i dozwolonych środowisk. Od architektów oprogramowania wymaga dostarczenia szablonów kodu, które następnie będą wdrożone do usługi Azure DevOps. Rozdział ten opisuje założenia architektoniczne przedmiotowej aplikacji, opis jej infrastruktury, uprawnień oraz procesu jej inicjalizacji.

3.1 Architektura aplikacji serwerowej

Aplikacja serwerowa została zbudowana we frameworku .NET 6.0 i składa się z *ASP.NET Core Web API*, wytworzonego zgodnie ze standardem REST³⁵ oraz *Service Workera*. Oba komponenty powiązane są z projektem o nazwie *Infrastructure*, który powiązany jest z projektem *Application* a ten - z projektem *Domain*. Taki sposób definiowania zależności pomiędzy projektami definiuje nam architektura warstwowa, która swoim wyglądem przypomina warstwy cebuli, skąd wywodzi się jej nazwa - *Onion Architecture* (architektura cebulowa). Architektura ta została stworzona w roku 2008 przez Jeffrey'a Palermo, który całą koncepcję opisał na swoim blogu³⁶. Rys. 3.1 przedstawia schematyczne przedstawienie zależności pomiędzy projektami.

³⁵ Representational state transfer – typ interfejsu programowania aplikacji, który opiera się na architekturze sieciowej umożliwiając aplikacjom wymianę danych poprzez żądania http.

³⁶ Jeffrey Palermo, *The Onion Architecture : part 1*, w: <https://jeffreypalermo.com/2008/07/the-onion-architecture-part-1/>, [dostęp 09.02.2023]



Rys. 3.1 Onion Architecture - powiązanie między projektami³⁷

Jak można zauważyć, zewnętrzna warstwa architektury składa się z projektów, które odpowiedzialne są za komunikację z zewnętrznymi zależnościami. Projekt *Api* odpowiedzialny jest za przyjmowanie żądań HTTP od aplikacji klienckiej. Projekt *Worker* przyjmuje zapytanie poprzez nasłuchiwanie wiadomości z kolejki *Azure Service Bus*.

Projekt *Infrastructure* jest również usytuowany w zewnętrznej warstwie i służy między innymi do zapisu danych w bazie danych, wysyłce wiadomości do usługi *Azure Service Bus*, czy zarządzaniem plikami w *Azure Storage Account*. W tym projekcie również znajdują się referencje do bibliotek *Azure .NET SDK*, która są odpowiedzialne za zarządzanie zasobami chmury Azure. Ogólną odpowiedzialnością tej warstwy jest komunikacja z zewnętrznymi usługami.

Następna warstwa o nazwie *Application* w głównej mierze odpowiedzialna jest za przygotowanie i przekazanie danych do warstwy domeny, wywołanie odpowiednich metod biznesowych i zwrócenie wyników do warstwy wyższej. W warstwie tej zaimplementowany jest również wzorzec CQRS³⁸, który został opisany w następnym podrozdziale.

Warstwa *Domain* jest główną warstwą, w której znajdują się reguły biznesowe, modele biznesowe oraz logika biznesowa. Warstwa ta stanowi jądro systemu i nie posiada referencji do innych projektów. Powinna być ona zaimplementowana w taki sposób, by była w jak największym stopniu odizolowana od zewnętrznych zależności.

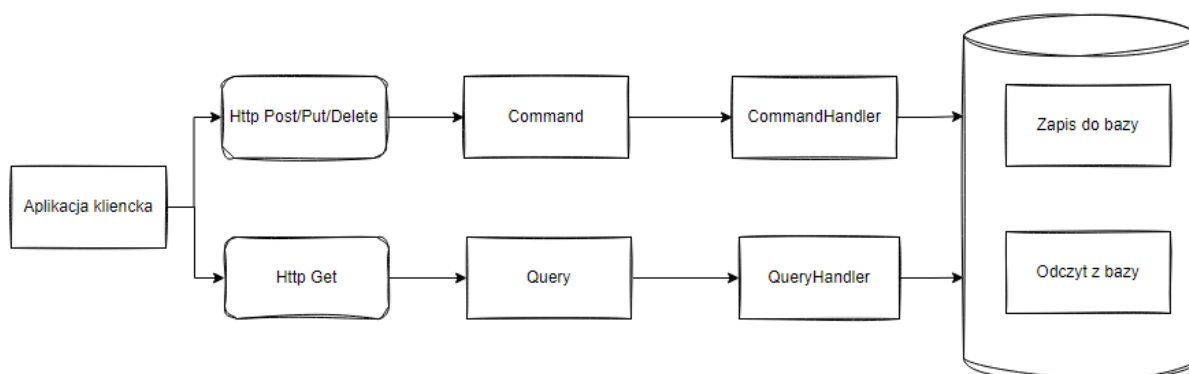
³⁷ Opracowanie własne – na podstawie: Jeffrey Palermo, The Onion Architecture : part 1, w: <https://jeffreypalermo.com/2008/07/the-onion-architecture-part-1/>, [dostęp 09.02.2023]

³⁸ CQRS (z ang. Command Query Responsibility Segregation) – wzorzec projektowy rozdzielający funkcję zapisu i odczytu.

Wybór takiej architektury motywowany jest tym, iż w przypadku chęci zaimplementowania połączenia z innymi dostawcami usług chmurowych jedyną warstwą, która zostanie zmieniona jest warstwa *Infrastructure*. Cała logika biznesowa i jej wywołanie zostanie takie samo, natomiast zmieni się jedynie implementacja połączeń z zewnętrznymi usługami.

3.2 Wzorzec projektowy CQRS

Wzorzec projektowy CQRS jest wzorcem polegającym na oddzieleniu operacji zmiany stanu aplikacji od jej odczytu. Za operacje związane ze zmianą stanu odpowiedzialne są obiekty *Command*. Posiadają one w sobie właściwości, które opisują w jaki sposób system ma się zmienić. Obiekt *Command* przekazywany jest do klasy *CommandHandler*, która odpowiedzialna jest za implementację zmiany poprzez odpowiednie modyfikacje modelu domenowego, a następnie zapisaniu go do bazy danych. Za operacje związane z odczytem odpowiedzialne są obiekty *Query*. Obiekty te posiadają właściwości opisujące, jakie dane mają zostać zwrócone z bazy danych. Przekazywane są do klasy *QueryHandler*, która odpowiedzialna jest za pobranie odpowiednich danych z bazy danych, odpowiedniej jest transformacji na model zrozumiały przez aplikacje klienckie, a następnie zwróceniu ich³⁹. Schemat opisanego powyżej wzorca w wersji graficznej prezentuje Rys. 3.2.



Rys. 3.2 Schemat wzorca CQRS⁴⁰

W przypadku aplikacji webowych zaimplementowanych zgodnie ze standardem REST zapytania HTTP typu POST/PUT/DELETE powinny być transformowane na obiekty typu *Command*, a zapytania HTTP typu GET powinno być transformowane na obiekt typu *Query*.

³⁹ Microsoft, *Wzorzec CQRS*, w: <https://learn.microsoft.com/en-us/azure/architecture/patterns/cqrs>, [dostęp: 19.02.2023]

⁴⁰ Opracowanie własne – na podstawie: Microsoft, *Wzorzec CQRS*, w: <https://learn.microsoft.com/en-us/azure/architecture/patterns/cqrs>, [dostęp: 19.02.2023]

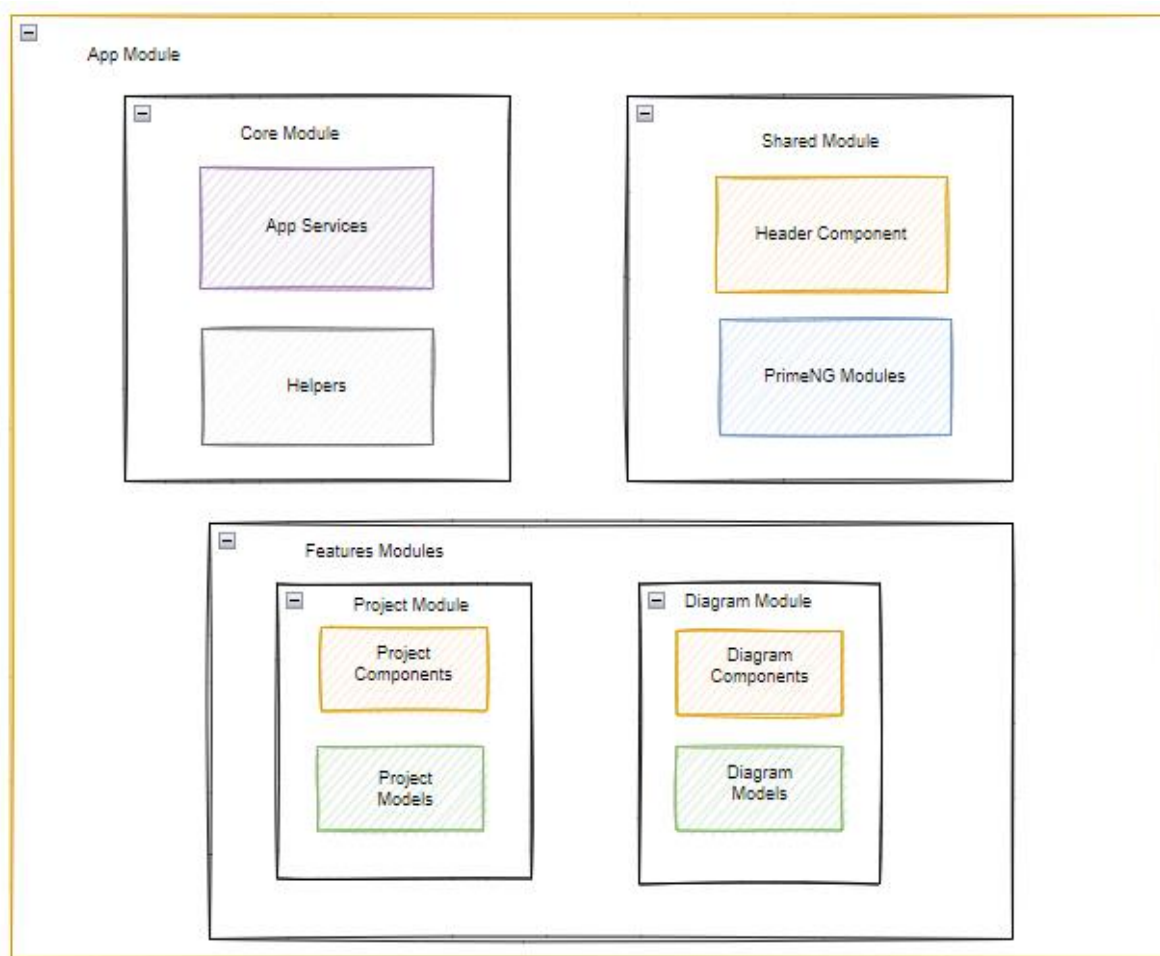
3.3 Architektura aplikacji klienckiej

Aplikacja kliencka została wytworzona we frameworku Angular i podzielona na kilka modułów. Głównym modulem jest *AppModule*, który jest modulem startowym i zawiera w sobie wszystkie inne moduły. Kolejny modulem jest *CoreModule*, który odpowiedzialny jest za komunikację z aplikacją serwerową oraz klasy pomocnicze. Zadaniem klas jest udostępnianie funkcjonalności wspólnej dla wielu klas lub funkcjonalności, które mogą być użyte w różnych miejscach w aplikacji.

Kolejnym modulem jest *SharedModul*, który zawiera w sobie wszystkie komponenty, które są dostępne dla każdej podstrony aplikacji. Przykładowym komponentem jest *HeaderComponent*, który odpowiedzialny jest za wyświetlanie nagłówka aplikacji. Moduł ten importuje również zależności z zewnętrznych bibliotek, które są używane w całej aplikacji, na przykład *DialogModule*, służący do wyświetlania okienek dialogowych i będący częścią pakietu PrimeNg⁴¹.

Następne moduły odpowiedzialne są za poszczególne części interfejsu aplikacji. *ProjectModule* odpowiada za część służącą do zarządzania projektami, natomiast *DiagramModule* - za część związaną z tworzeniem diagramu. Obrazowy podział modułów został przedstawiony na Rys. 3.3.

⁴¹ PrimeNg - Pakiet darmowych elementów interfejsu użytkownika współpracujących z Angular.

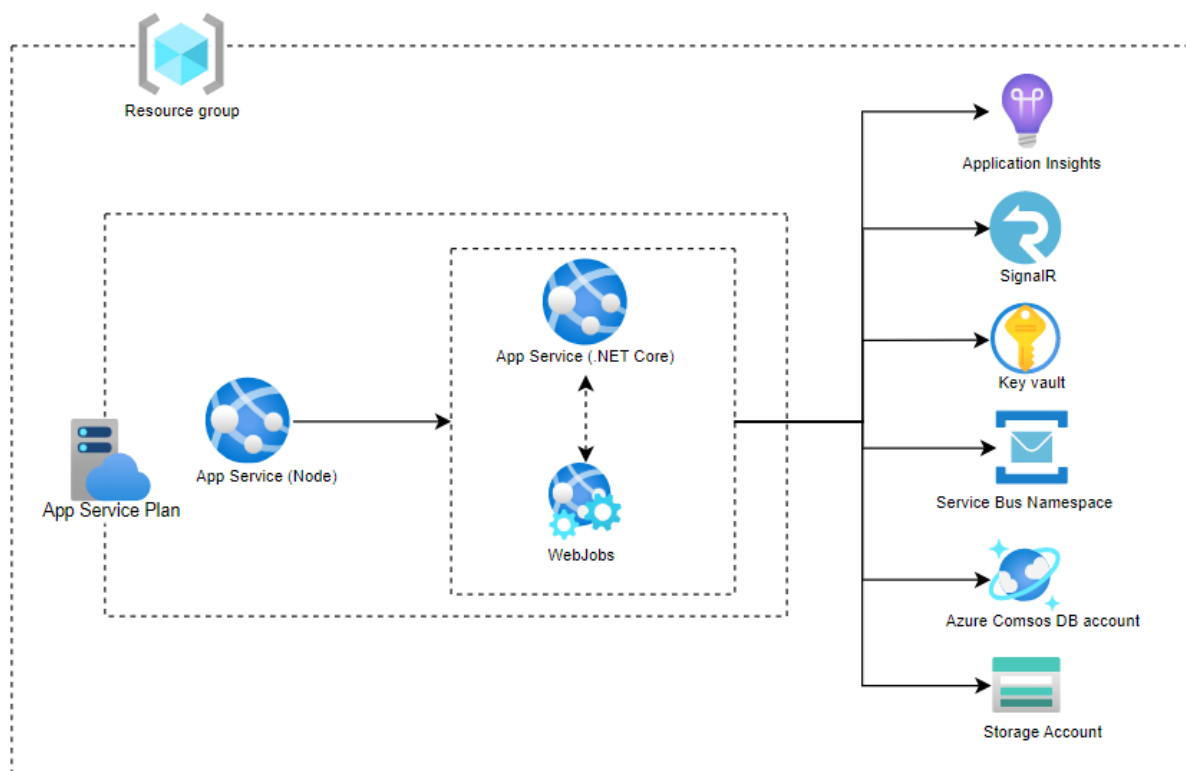


Rys. 3.3 Architektura aplikacji klienckiej⁴²

3.4 Infrastruktura aplikacji

Tytułowa aplikacja Cloud Developer Designer została wdrożona do chmury Microsoft Azure. Infrastruktura składa się z dwóch zasobów typu App Service, które są skalowane i zarządzane dzięki App Service Plan. Pierwszy zasób App Service działa na środowisku uruchomieniowym Node, co umożliwia hosting aplikacji klienckiej. Drugi zasób App Service, działa na środowisku uruchomieniowym .NET, a jego zadaniem jest hosting aplikacji serwerowej. Współpracuje on z wieloma podrzędnymi usługami, takimi jak *Application Insights*, *Azure SignalR*, *Azure Key Vault*, *Azure Service Bus*, *Azure CosmosDB* oraz *Azure Storage Account*. Cała infrastruktura aplikacji została przedstawiona na Rys. 3.4, co pozwala na łatwe zrozumienie tego, jak wszystkie składniki są ze sobą połączone.

⁴² Opracowanie własne



Rys. 3.4 Infrastruktura aplikacji⁴³

3.5 Uprawnienia aplikacji

Aplikacja do poprawnego działania potrzebuje odpowiednich uprawnień nadawanych dla chmury Microsoft Azure oraz platformy Azure DevOps. Na potrzeby tytułowej pracy zostały utworzone dwie tożsamości *CloudDeveloperDesigner.Web-dev* oraz *CloudDeveloperDesigner.Api-dev* w usłudze Azure Active Directory (Rys. 3.5).

Home > Katalog domyślny

Katalog domyślny | App registrations

Azure Active Directory

Enterprise applications

Devices

App registrations

Identity Governance

Application proxy

Custom security attributes (Preview)

Licenses

Cross-tenant synchronization (Preview)

+ New registration

Endpoints

Troubleshooting

Refresh

Download

Preview features

...

All applications

Owned applications

Deleted applications

Applications from personal account

Start typing a display name or application (client) ID to filter these r...

Add filters

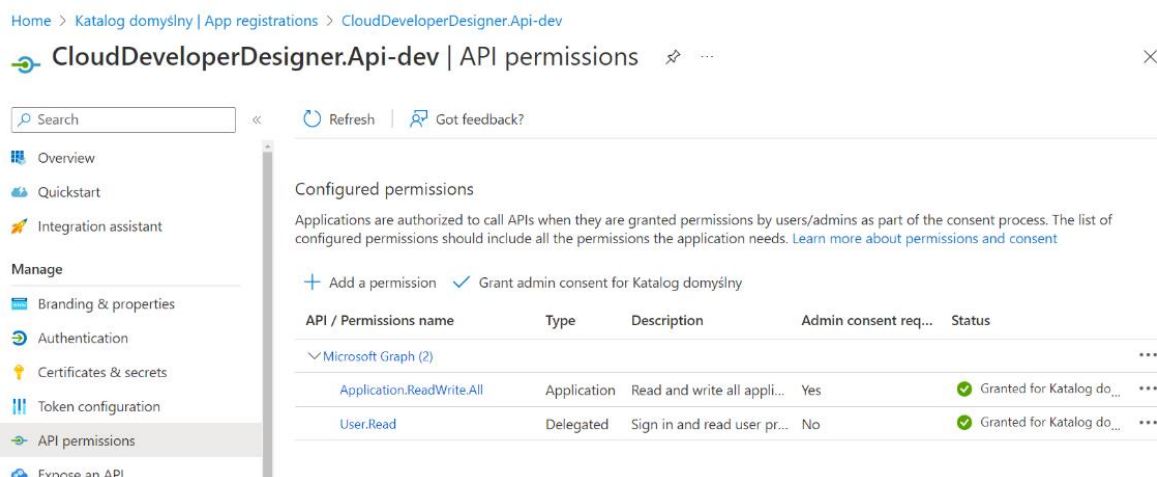
2 applications found

Display name	Application (client) ID	Created on	Certificates & secrets
CloudDeveloperDesigner.Web-dev	34fc842f-25f6-4000-a837-b...	12/23/2022	-
CloudDeveloperDesigner.Api-dev	64f73911-652d-4f15-98b2-...	12/23/2022	Current

Rys. 3.5 Utworzone tożsamości aplikacji

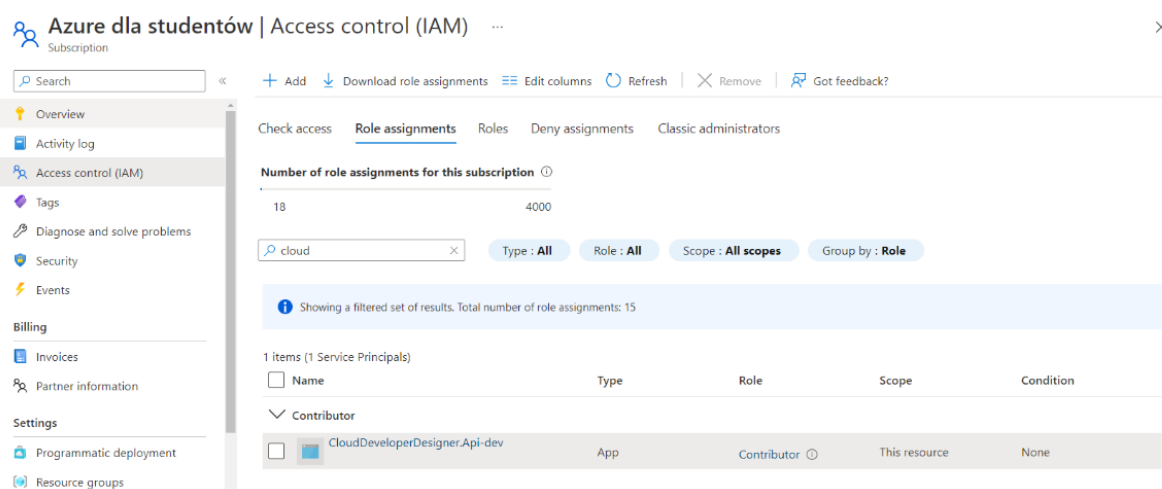
⁴³ Opracowanie własne

Tożsamość *CloudDeveloperDesigner.Web-dev* przypisana jest do aplikacji klienckiej i posiada uprawnienia do czytania informacji o zalogowanym użytkowniku (*User.Read*). *CloudDeveloperDesigner.Api-dev* przypisana jest do aplikacji serwerowej i oprócz uprawnień dotyczących nadanych aplikacji klienckiej posiada również uprawnienia do pobierania informacji o innych tożsamościach aplikacji oraz uprawnienia do ich edycji (Rys. 3.6).



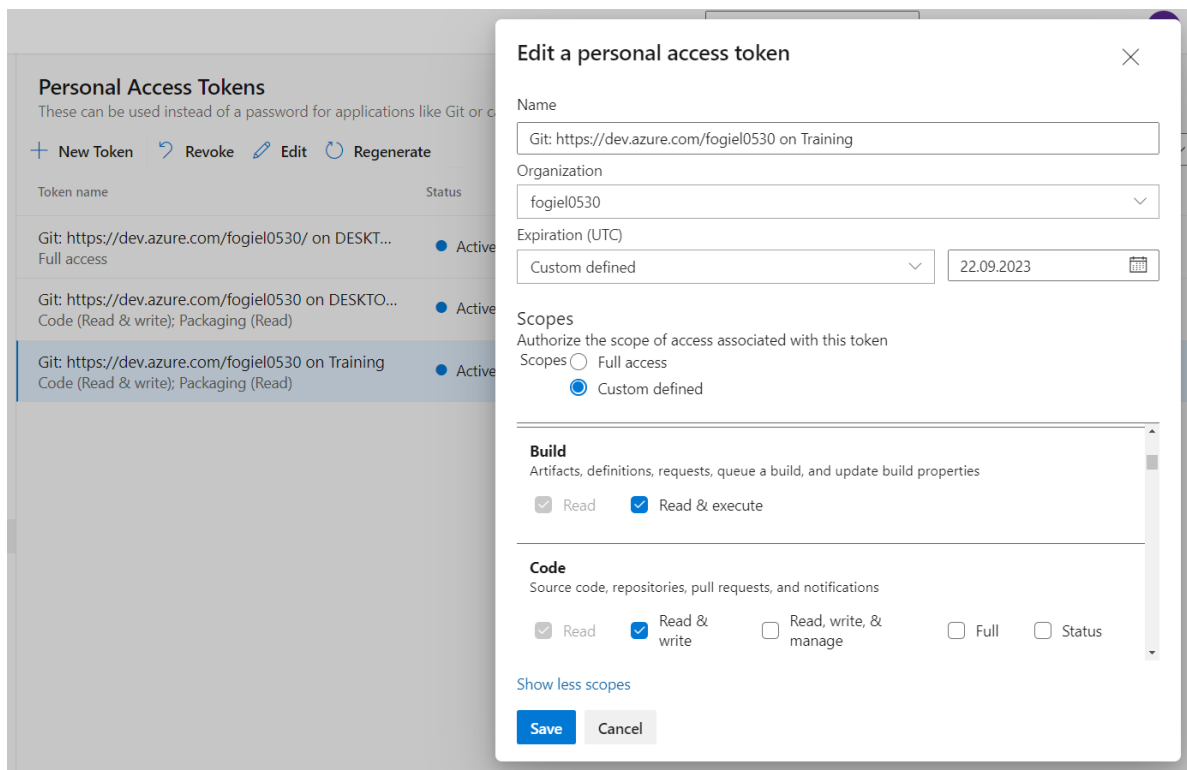
Rys. 3.6 Uprawnienia aplikacji serwerowej

Oprócz wyżej wymienionych uprawnień zostało również nadane uprawnienie *Contributor*, dające pełne prawo do zarządzania zasobami w ramach subskrypcji *Azure dla studentów* (Rys. 3.7).



Rys. 3.7 Nadane uprawnienie Contributor dla subskrypcji

Dla platformy Azure DevOps został wygenerowany klucz PAT (z ang. Personal Access Token) z możliwością czytania i edytowania repozytoriów kodu oraz procesów budowania, co prezentuje Rys. 3.8.



Rys. 3.8 Tworzenie PAT w Azure DevOps

3.6 Inicjalizacja aplikacji

Aplikacja do poprawnego działania wymaga utworzenia wszystkich usług przedstawionych w opisie infrastruktury oraz odpowiedniego ich przygotowania. Pierwszą taką usługą jest baza danych *CosmosDb*, w której należy utworzyć kontenery wyszczególnione w Tab.3.1 :

Tab. 3.1 Opis kontenerów bazy CosmosDb

Kontener	Przeznaczenie
applications	zawiera informacje opisujące możliwe do wdrożenia aplikacje
configurations	zawiera informacje opisujące ustawienia konfiguracyjne aplikacji
deploymentLogs	zawiera logi wykonanych wdrożeń
deployments	zawiera informacje o utworzonych wdrożeniach
projects	zawiera informacje o utworzonych projektach
resourceses	zawiera informacje o możliwych do wdrożenia zasobach

Kontener *applications* należy uzupełnić poprzez dodanie wpisów dla możliwych do wdrożenia aplikacji. Każdy z wpisów musi zawierać informacje wskazane w Tab. 3.2:

Tab. 3.2 Opis właściwości obiektu Application

Właściwość	Przeznaczenie
Name	wewnętrzna nazwa aplikacji
DisplayName	nazwa wyświetlana w aplikacji klienckiej
Type	typ aplikacji 1 – aplikacja kliencka 2- aplikacja serwerowa
BlobName	nazwa pliku z kodem źródłowym aplikacji
IconFile	nazwa ikony aplikacji
SettingsFiles	kolekcja plików znajdujących się w kodzie źródłowym, do których zostaną dopisane informacje o połączonych zasobach
BuildFiles	nazwa pliku YAML w kodzie źródłowym zawierająca definicje budowania
Id	unikalny identyfikator

Kontener *resources* należy uzupełnić poprzez dodanie opisów dla poszczególnych zasobów możliwych do wdrożenia. Każdy z wpisów musi zawierać informacje wyszczególnione w Tab. 3.3:

Tab. 3.3 Opis właściwości obiektu Resource

Właściwość	Przeznaczenie
Name	Wewnętrzna nazwa zasobu
DisplayName	Nazwa wyświetlana w aplikacji klienckiej
IconFile	Nazwa ikony aplikacji
ResourceNaming	Obiekt opisujący sposób tworzenia nazwy zasobu
IsSecretProvider	Wartość true/false wskazująca zasób, który służy do przechowywania kluczy
TemplateFileName	Nazwa pliku JSON z szablonem ARM
Id	Unikalny identyfikator

Ostatnim kontenerem, który musi zostać uzupełniony, jest kontener *configurations*, który widoczny jest na Rys. 3.9. Do kontenera tego należy dodać jeden wpis zawierający w sobie informacje o dostępnych lokalizacjach, środowiskach oraz wymaganych tagach.

The screenshot displays the Azure Synapse Studio interface. The top navigation bar includes links for 'New Container', 'Enable Azure Synapse Link', 'New Notebook', 'Connect to GitHub', and 'New SQL Query'. The main workspace is divided into three panes. The left pane shows a tree view of the 'DATA' section, with 'clouddesigner' expanded to show 'configurations'. The middle pane shows a table of 'configurations' items, with a single row visible. The right pane shows the JSON configuration for the 'configurations' container, which includes 'AvailableLocations', 'AvailableEnvironments', and 'MandatoryTags'.

```

{
  "AvailableLocations": [
    {
      "Name": "westeurope",
      "DisplayName": "West Europe"
    },
    {
      "Name": "northeurope",
      "DisplayName": "North Europe"
    },
    {
      "Name": "swedencentral",
      "DisplayName": "Sweden Central"
    }
  ],
  "AvailableEnvironments": [
    "dev",
    "test",
    "stg",
    "prod"
  ],
  "MandatoryTags": [
    "AppId",
    "CostCenter"
  ]
}

```

Rys. 3.9 Obiekt konfiguracyjny

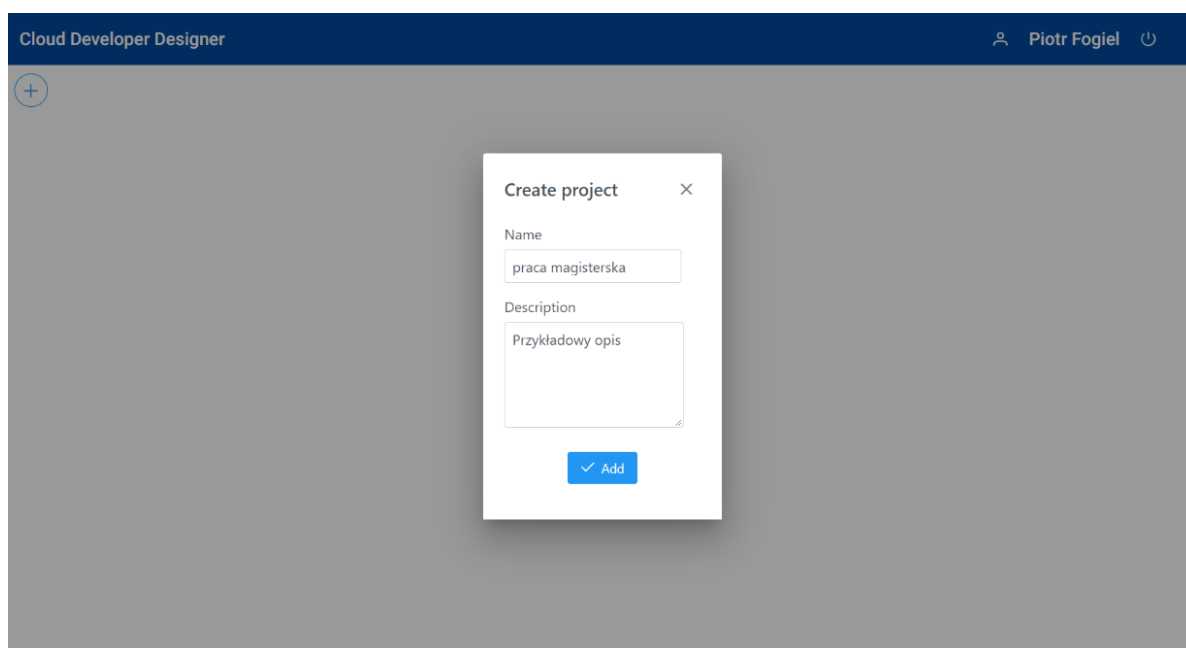
Kolejnym zasobem, który należy skonfigurować po utworzeniu jest Azure Storage Account. Do poprawnego działania aplikacji potrzebne są dwa kontenery *codetemplates* oraz *resourcetemplates*. Kontener *codetemplates* przechowuje w sobie zarchiwizowane szablony kodu, które będą użyte podczas tworzenia repozytorium kodu w usłudze Azure DevOps. Kontener *resourcetemplates* zawiera w sobie szablony ARM, które wykorzystywane są w procesie wdrażania zasobów do chmury.

ROZDZIAŁ 4. ZASADA DZIAŁANIA APLIKACJI

Aplikację zaprojektowano w taki sposób, aby użytkownicy poprzez dostarczenie minimalnej ilości informacji mogli w intuicyjny sposób tworzyć infrastrukturę swoich systemów. Nazwy zasobów są automatycznie podpowiadane, co znacząco przyspiesza proces tworzenia. Graficznie przedstawienie zależności pomiędzy aplikacjami oraz zasobami znacząco ułatwia zrozumienie całej infrastruktury. W tym rozdziale zaprezentowano zasadę działania aplikacji.

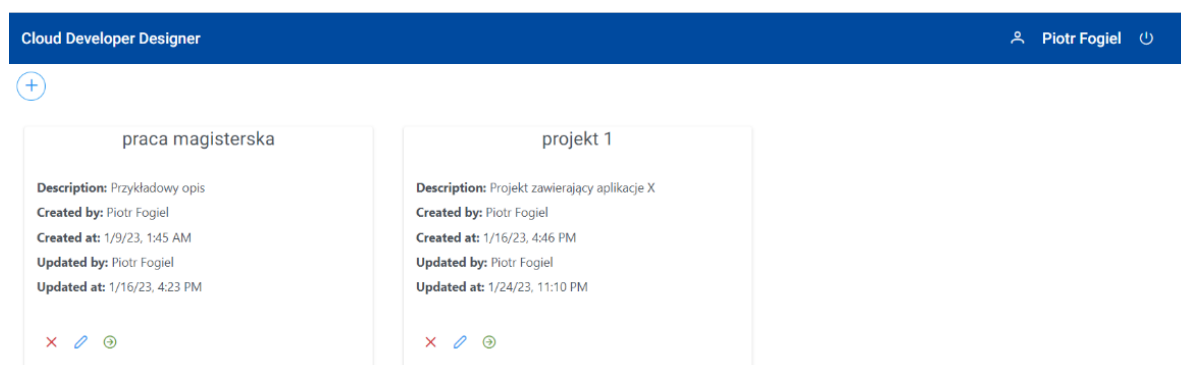
4.1 Tworzenie projektu

Aby rozpocząć pracę z aplikacją należy utworzyć projekt, który jest głównym elementem zawierającym w sobie informacje dotyczące wdrożenia. Użytkownik po kliknięciu ikony plusa w lewym górnym rogu otwiera okno dialogowe (Rys. 4.1), w którym podaje nazwę projektu oraz krótki opis.






Rys. 4.1 Okno dialogowe dodawania projektu

Po poprawnym wprowadzeniu nazwy oraz opisu użytkownik poprzez kliknięcie przycisku „Add” tworzy nowy projekt, a okno dialogowe automatycznie zostaje zamknięte. W głównym panelu aplikacji do istniejących już projektów zostaje dodany kafelek z danymi nowoutworzonego projektu. Na Rys. 4.2 przedstawiono wizualizację listy kafelków.









Rys. 4.2 Lista kafelków zawierających informacje o projektach

Każdy z kafelków składa się z nagłówka zawierającego nazwę projektu oraz z informacji dotyczących twórcy, daty utworzenia, modyfikacji oraz osoby wprowadzającej modyfikację. W stopce kafelka znajdują się trzy ikony o następujących funkcjach:

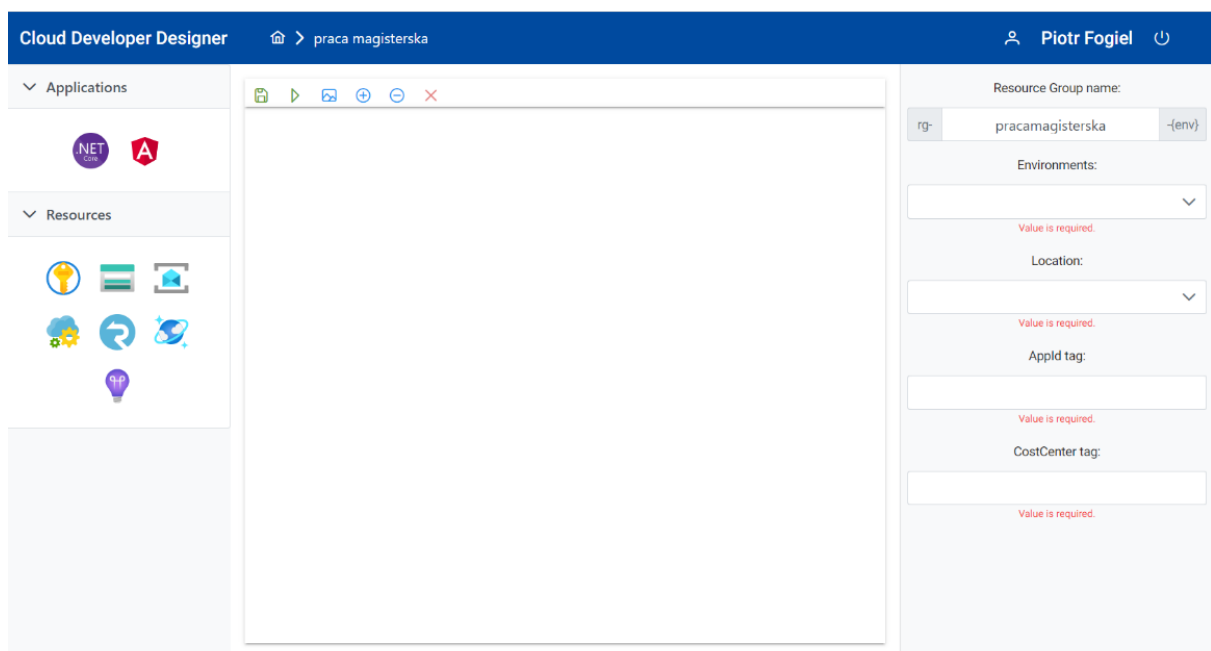
-  Usunięcie projektu
-  Edycja podstawowych informacji o projekcie
-  Przekierowanie do głównego obszaru roboczego

4.2 Tworzenie diagramu wdrożeniowego

Diagram wdrożeniowy tworzony jest w obszarze roboczym, który składa się z trzech części. Po lewej stronie znajdują się obiekty wdrożeniowe w postaci ikon, które użytkownik może przenieść na środkowy panel diagramu. Panel diagramu składa się z górnego paska narzędziowego oraz pola, na którym ma się znaleźć diagram wdrożeniowy. Pasek narzędziowy zawiera w sobie ikony :

-  Zapis projektu diagramu
-  Otworzenie okna wdrożeniowego
-  Wygenerowanie zrzutu ekranowego diagramu w postaci pliku JPEG
-  Powiększenie diagramu
-  Pomniejszenie diagramu
-  Usunięcie pojedynczego elementu diagramu

Na Rys. 4.3 znajduje się całościowy obraz obszaru roboczego.



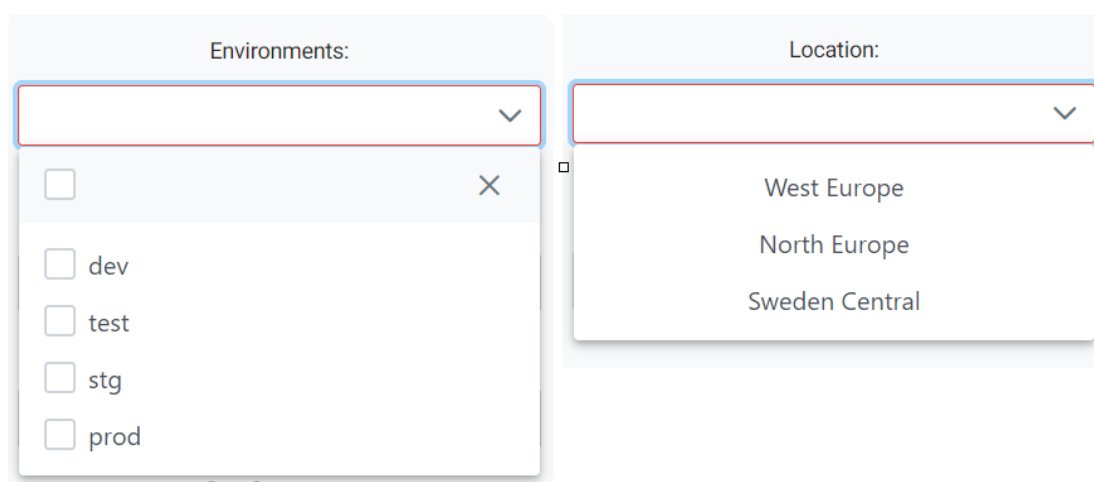
Rys. 4.3 Obszar roboczy

Po prawej stronie obszaru roboczego znajdują się główne informacje dotyczące wdrożenia. Pierwsze pole, które jest automatycznie wypełnianie nazwą projektu służy do utworzenia nazwy *Resource group*. Jak widać na powyższym rysunku, pole tekstowe składa się

z nieedytowalnego prefiksu 'rg', który wymusza na użytkownika posługiwanie się odpowiednią konwencją nazewnictwa. Przyjmuje ona, iż nazwa grupy zasobów ma kształt:

rg- {nazwa}-{środowisko}

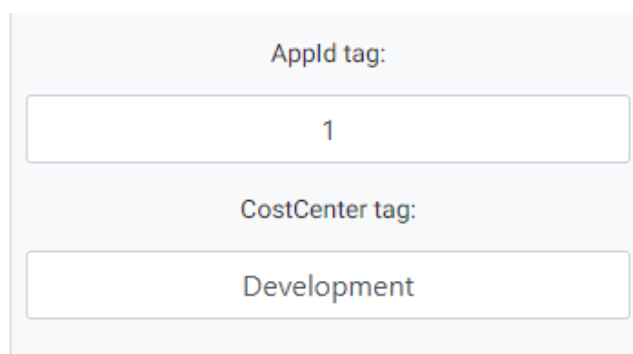
Kolejnymi polami wymaganymi do uzupełniania są dwie rozwijalne listy (Rys. 4.4). Pierwsza z nich - *Environments* - jest listą wielokrotnego wyboru, w której określa się, na ilu środowiskach utworzone zostaną zasoby. Druga – *Location* - to lista jednokrotnego wyboru, w której należy wybrać lokalizację wdrożenia.



The image shows two side-by-side dropdown menus. The left menu is titled 'Environments:' and has a search bar at the top. Below the search bar, there are four checkboxes, each followed by a label: 'dev', 'test', 'stg', and 'prod'. The right menu is titled 'Location:' and has a search bar at the top. Below the search bar, there are three text labels: 'West Europe', 'North Europe', and 'Sweden Central'.

Rys. 4.4 Rozwijalne listy wyboru środowisk i lokalizacji

Tuż pod opisanymi powyżej listami znajdują się automatycznie generowane wolne pola tekstowe, w których użytkownik obligatoryjnie podaje wartości zdefiniowanych w bazie danych Tag-ów (Rys. 4.5).



The image shows two text input fields. The first field is labeled 'Appld tag:' and contains the value '1'. The second field is labeled 'CostCenter tag:' and contains the value 'Development'.

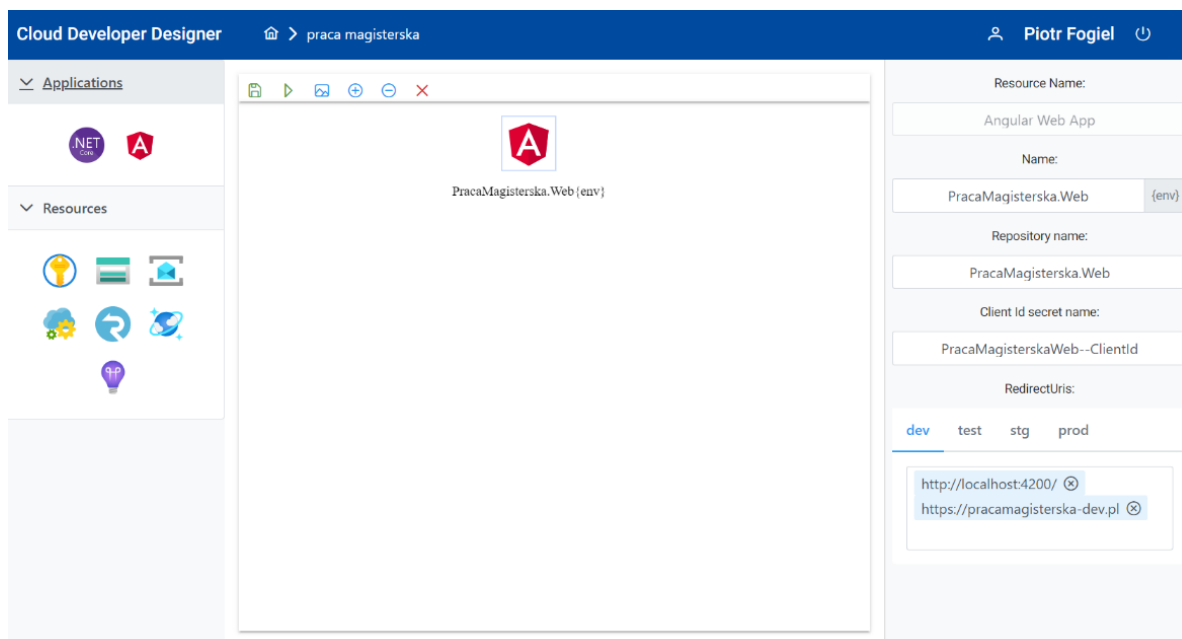
Rys. 4.5 Pola tekstowe do wprowadzania wartości Tag-ów

Wyżej wymienione rozwijane listy oraz pola są automatycznie generowane na podstawie wartości znajdujących się w bazie danych.

Przeciągnięcie ikon umieszczonych w grupie Applications na pole robocze i następne ich zaznaczenie na polu roboczym wywołuje po prawej stronie obszaru roboczego automatyczne zastąpienie bazowych informacji wdrożeniowych informacjami ściśle związanymi z zaznaczonym obiektem. Dla każdego typu obiektów z grupy Applications konieczne jest podanie przez użytkownika:

- *Name* nazwa służąca do rejestracji aplikacji w AAD
- *Repository name* nazwa repozytorium kodu utworzonego w Azure DevOps
- *Client Id secret name* nazwa tajnego klucza usługi KeyVault zawierającego ClientId utworzonego podczas rejestracji aplikacji w AAD

Dla aplikacji klienckich użytkownik może podać również adresy przekierowań dla poszczególnych środowisk *RedirectUris*. Rys 4.6 ukazuje proces definiowania aplikacji klienckiej.

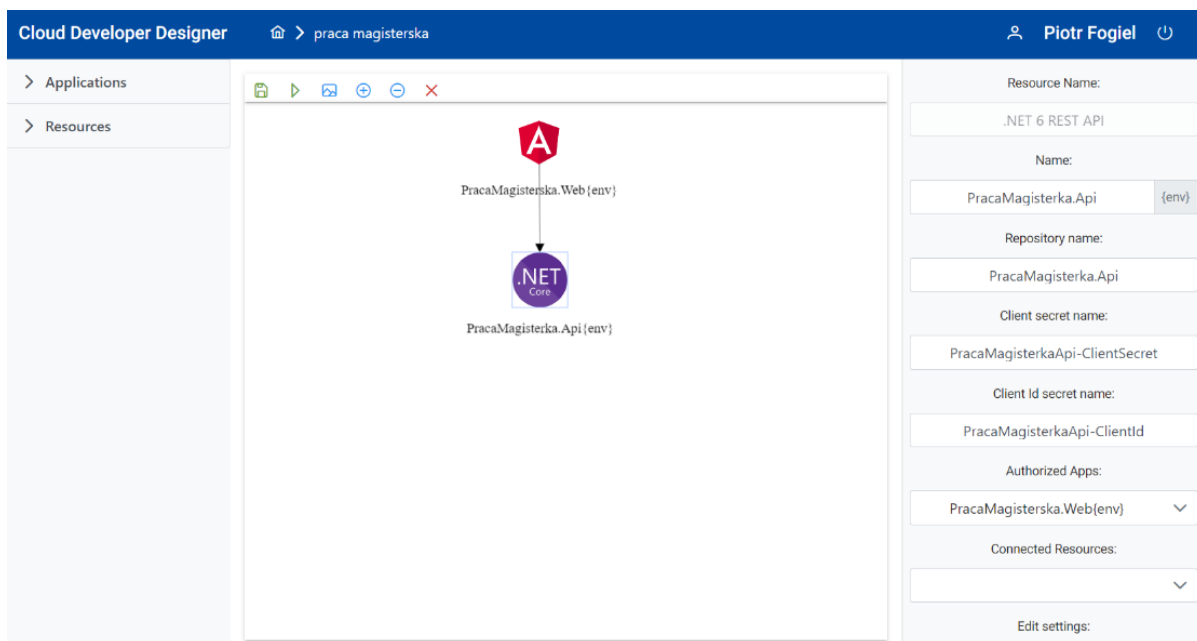


Rys. 4.6 Proces definiowania aplikacji klienckiej

Dla aplikacji typu serwerowego użytkownik dodatkowo zobligowany jest do podania:

- *Client secret name* Nazwa klucza usługi KeyVault zawierającego wygenerowany dla aplikacji *Client Secret* w usłudze AAD;
- *Authorized Apps* Lista rozwijana wielokrotnego wyboru zawierająca nazwy dostępnych na diagramie obiektów z grupy Applications. Wartości wybrane z listy służą do uzupełniania w AAD sekcji *Authorized client applications*;
- *Connected Resources* Lista rozwijana wielokrotnego wyboru zawierająca nazwy dostępnych na diagramie obiektów z grupy Resources. Wartości wybrane z listy służą do uzupełnienia pliku konfiguracyjnego;
- *Edit settings* Przycisk otwierający okno dialogowe, które umożliwia modyfikację pliku konfiguracyjnego aplikacji (w przypadku aplikacji .NET Core – appsettings.json)

Rys. 4.7 przedstawia proces definiowania aplikacji serwerowej.

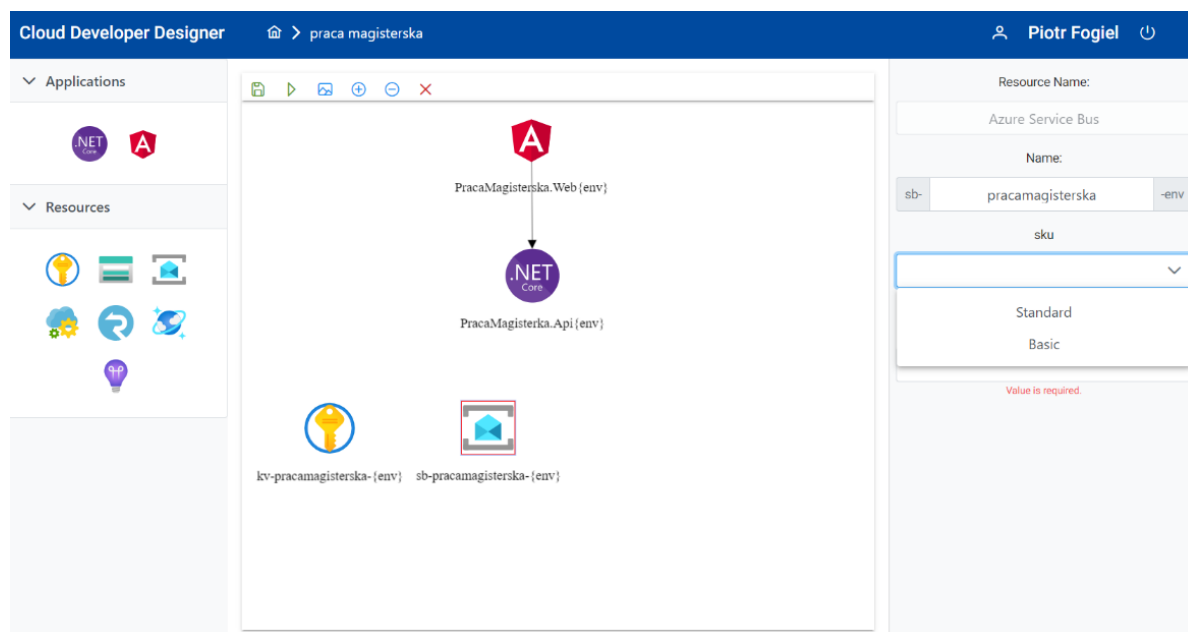


Rys. 4.7 Proces definiowania aplikacji serwerowej

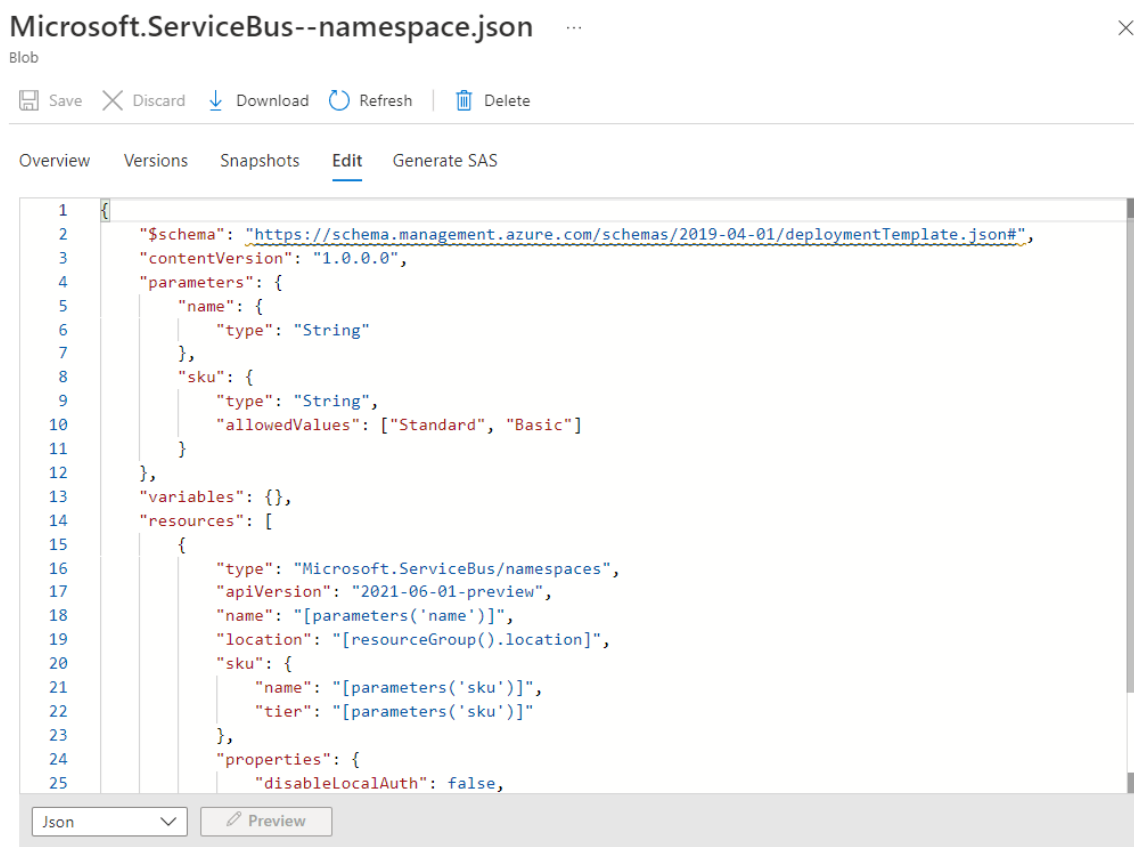
Po przeciągnięciu na pole robocze oraz zaznaczeniu elementu z grupy *Resources* z prawej strony generowana jest lista dostępnych do wypełnienia pól. Pierwsze zablokowane do edycji pole o nazwie Resource zawiera informacje o typie zasobu. Kolejnym polem jest nazwa, która będzie użyta do wdrożenia. Nazwa obiektu wdrożeniowego budowana jest według następującego schematu:

$$\{\text{prefiks}\}-\{\text{nazwa}\}-\{\text{środowisko}\}$$

Wartość prefiksu jest inna dla każdego z typu zasobów i pobierana jest z bazy danych. Dwa wyżej wymienione pola występują dla każdego typu zasobów. Kolejne pola generowane są na podstawie szablonu ARM (Rys. 4.9). Poniższy rysunek przedstawia proces wypełniania zasobu Azure Service Bus, dla którego szablon ARM (Rysunek 38) oprócz parametru *Name*, zawiera również parametr *sku*, który posiada zdefiniowane dozwolone wartości. Wartości te wyświetlone są w postaci listy jednokrotnego wyboru (Rys. 4.8).



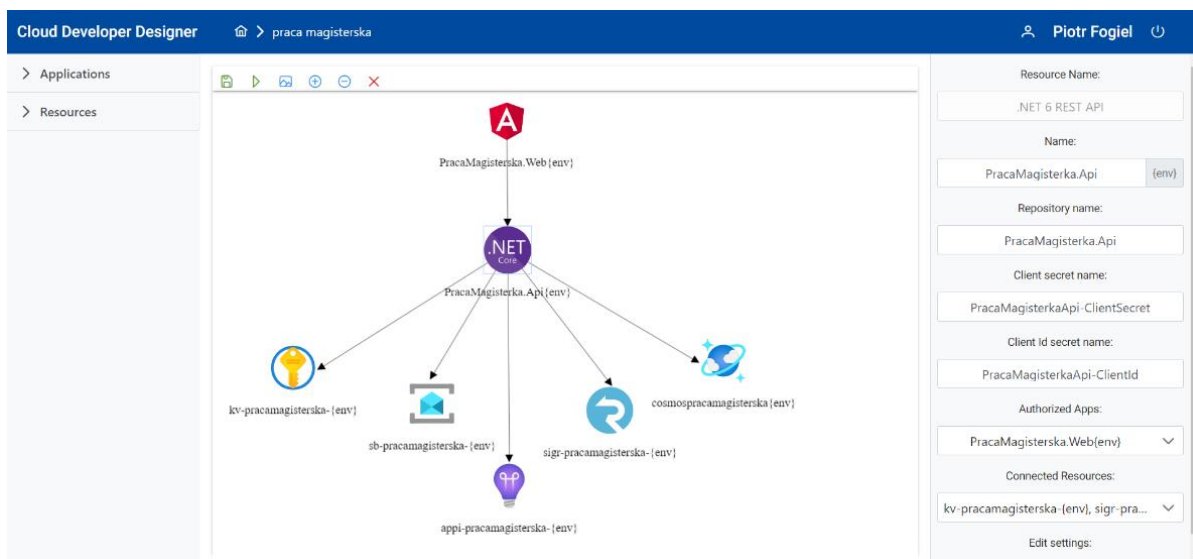
Rys. 4.8 Wypełnianie definicji zasobu Azure Service Bus



Rys. 4.9 Szablon ARM zasobu Azure Service Bus

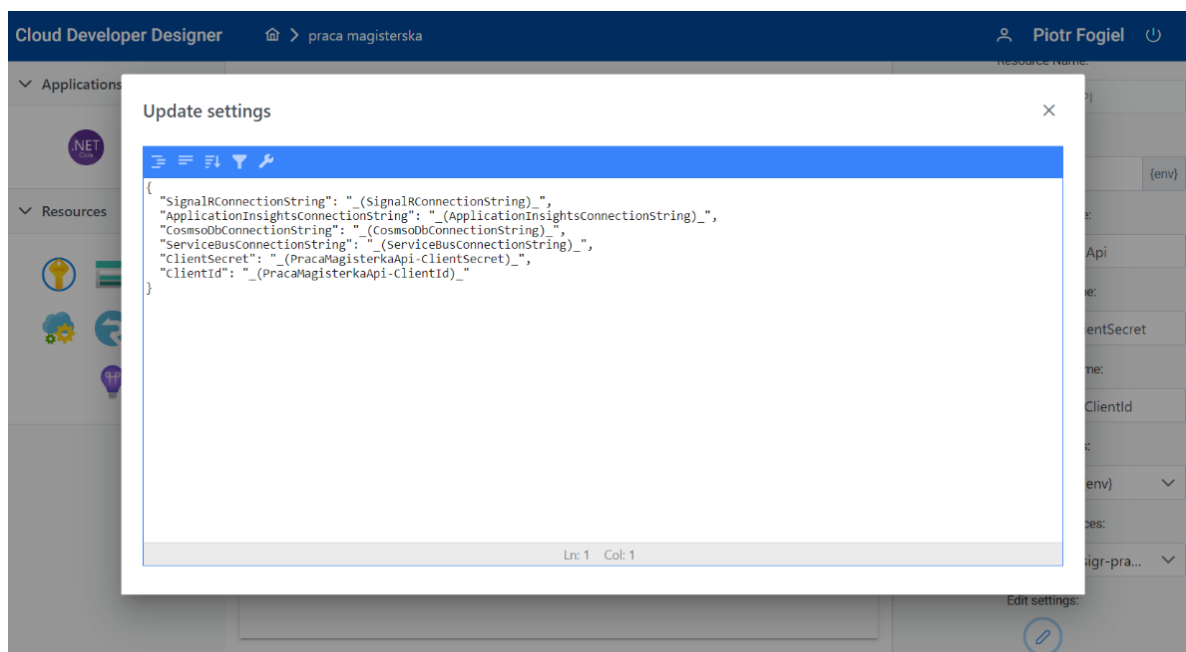
Dla każdego typu zasobu, oprócz zasobu oznaczonego jako Secret Provider, dostępne jest również pole Secret name, które będzie użyte w usłudze KeyVault jako nazwa klucza, w której zapisane będą klucze dostępowe dla utworzonego zasobu. Nazwa ta będzie również użyta do modyfikacji pliku konfiguracyjnego aplikacji serwerowej.

Po utworzeniu wszystkich aplikacji i zasobów oraz połączeniu ich za pomocą wybrania odpowiednich wartości z list „Authorized Apps” i „Connected Resources” aplikacji serwerowej przykładowy diagram wdrożeniowy może wyglądać tak jak na Rys. 4.10.



Rys. 4.10 Przykładowy diagram wdrożeniowy

Przykładowy diagram składa się z aplikacji klienckiej „PracaMagisterska.Web{env}”, która połączona jest z aplikacją serwerową „PracaMagisterska.Api{env}” poprzez użycie funkcji „Authorized Apps”. Diagram zawiera również pięć zasobów chmurowych połączonych z aplikacją za pomocą funkcji „Connected Resources”. Funkcja ta pozwala na modyfikację pliku konfiguracyjnego aplikacji serwerowej poprzez dodanie do niego wartości użytych w polu „Secret Name” zasobu. Wartości te mogą być edytowane poprzez kliknięcie przycisku „Edit settings”. Po kliknięciu otwierane jest okno dialogowe (Rys. 4.11) gdzie wyświetlana jest możliwa do modyfikacji kolekcja klucz-wartość dla wszystkich połączonych zasobów.

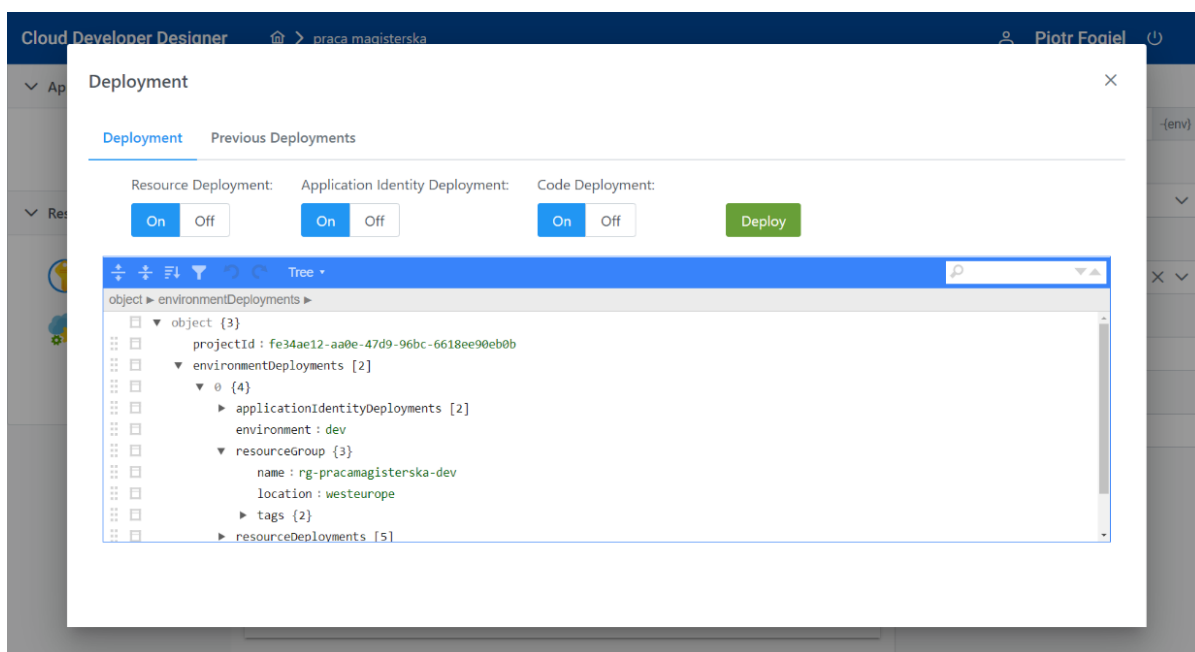


Rys. 4.11 Edytowanie pliku konfiguracyjnego aplikacji serwerowej

Po wypełnieniu wszystkich informacji użytkownik powinien zapisać projekt, a następnie przejść do zrealizowania wdrożenia.

4.3 Okno dialogowe wdrożenia

Aby wykonać wdrożenie należy otworzyć okno dialogowe poprzez naciśnięcie przycisku dostępnego nad diagramem. Okno dialogowe posiada dwie zakładki *Deployment* (Rys. 4.12) oraz *Previous Deployments*. Pierwsza z nich służy do wykonania wdrożenia, a druga do przeglądania poprzednich.



Rys. 4.12 Zakładka Deployment okna wdrożeniowego

W zakładce *Deployment* znajdują się trzy przyciski typu włącz/wyłącz:

Resource Deployment

On – wdrożenie obejmuje utworzenie zasobów

Off – wdrożenie nie tworzy zasobów

Application Identity Deployment

On – wdrożenie obejmują utworzenie aplikacji w usłudze AAD

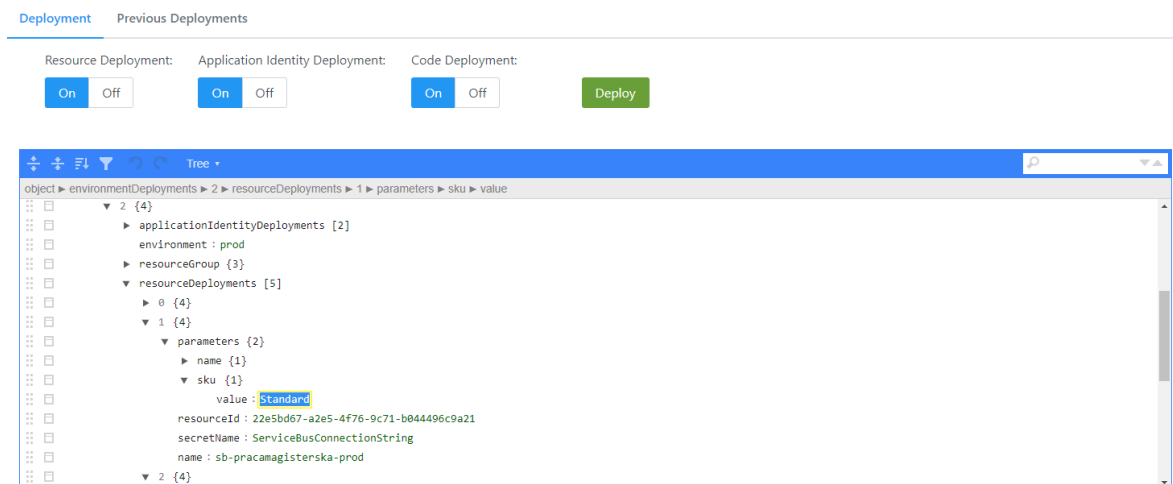
Off – wdrożenie nie tworzy aplikacji w usłudze AAD

Code Deployment

On – wdrożenie obejmujące utworzenie repozytorium kodu w usłudze Azure DevOps

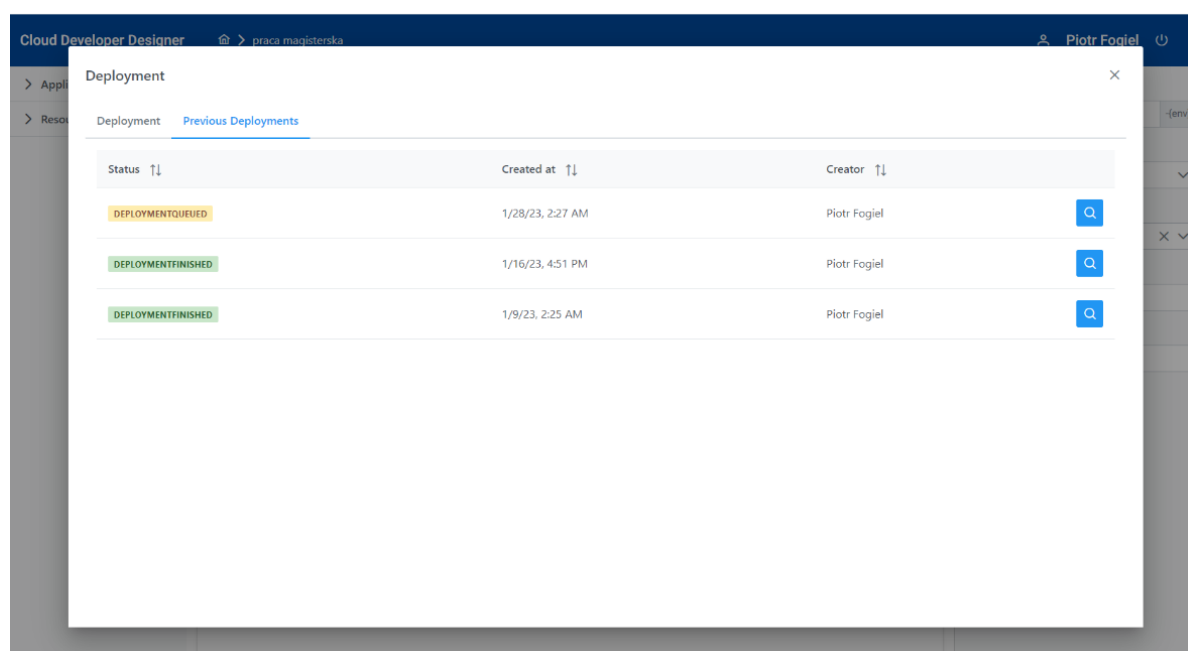
Off – wdrożenie nie tworzy repozytorium kodu

Przycisk *Deploy* powoduje rozpoczęcie procesu wdrażania. Pod sekcją z przyciskami wyświetlany jest obiekt wdrożeniowy, który zostanie wysłany do serwera. Obiekt ten jest możliwy do edycji przez użytkownika w przypadku, gdyby chciał on zmienić poszczególne wartości dla wybranego środowiska. Dla przykładu można przyjąć sytuację, w której użytkownik chciałby zmienić wartość parametru sku usługi Azure Service Bus z wartości Basic na Standard dla środowiska produkcyjnego (Rys. 4.13).






Rys. 4.13 Zmiana parametru zasobu dla poszczególnego środowiska

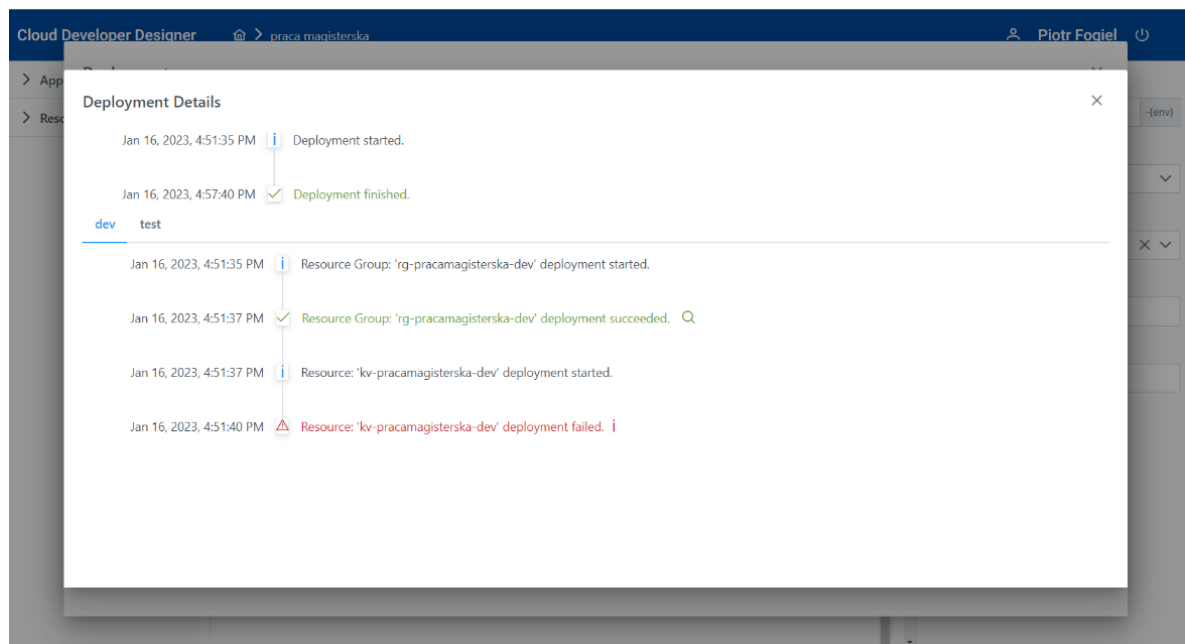
Zakładka *Previous Deployments* (Rys. 4.14) zawiera listę ze wszystkim wykonanymi dla danego projektu wdrożeniami. Lista zawiera trzy kolumny *Status*, *Created at*, *Creator* oraz kolumnę z przyciskiem otwierającym szczegóły wdrożenia.



Rys. 4.14 Zakładka Previous Deployments okna wdrożeniowego

Po wybraniu ikony lupy aplikacja przekierowuje do sekcji *Deployment Details* (Rys. 4.15). Sekcja zawiera dokładne informacje o procesie wdrożenia. Dla każdego ze środowisk widoczna jest osobna zakładka zawierająca szczegółowy przebieg czasowy wdrożenia. W przebiegu czasowym możemy wyróżnić trzy typy odświeżanych automatycznie wiadomości:

-  wiadomość informująca o rozpoczęciu wdrożenia danego zasobu
-  wiadomość informująca o pozytywnym wdrożeniu zasobu wraz z możliwością przejścia pod jego adres URL poprzez kliknięcie ikony lupy
-  wiadomość informująca o błędzie podczas wdrażania zasobu



Rys. 4.15 Przykładowa sekcja Deployment Details

4.4 Proces wdrożenia

Proces wdrożenia jest najważniejszym, a zarazem najtrudniejszym z perspektywy implementacyjnej procesem w całym systemie i zaczyna się po kliknięciu przycisku *Deploy* poprzez użytkownika. Aplikacja kliencka tworzy obiekt *CreateDeploymentCommand*, a następnie wysyła go poprzez zapytanie HTTP Post do Web API, która rozpoznaje i przetwarza model przesłany w zapytaniu w kontrolerze *DeploymentController* w akcji *Deploy* (Rys. 4.16).

```
[HttpPost]
[ProducesResponseType(StatusCodes.Status202Accepted)]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
[ProducesResponseType(StatusCodes.Status401Unauthorized)]
public async Task<ActionResult<Guid>> Deploy([FromBody] CreateDeploymentCommand command, CancellationToken ct)
{
    await _mediator.Send(command, ct);
    return Accepted(command.DeploymentId);
}
```

Rys. 4.16 Akcja Deploy w kontrolerze Deployment

Metoda *Deploy* przekazuje obiekt *CreateDeploymentCommand* poprzez obiekt *_mediator* do klasy *CreateDeploymentCommandHandler* zgodnie z wzorcem CQRS, a następnie po wykonaniu kodu znajdującego się w tej klasie zwraca do aplikacji klienckiej

status 202 – Accepted ⁴⁴. Kod klasy *CreateDeploymentCommandHandler* (Rys. 4.17) tworzy obiekt *Deployment* zgodnie z parametrami przesłanymi w zapytaniu, zapisuje go w bazie danych *CosmosDb*, a następnie wysyła wiadomość do usługi *Azure Service Bus*.

```
public async Task<Unit> Handle(CreateDeploymentCommand command, CancellationToken cancellationToken)
{
    var currentUser = await _currentUserService.GetUserName();

    var deployment = new Deployment(
        command.DeploymentId,
        currentUser,
        _clock.CurrentDate(),
        command.ProjectId);

    await AddCodeDeployments(command, deployment, cancellationToken);
    await AddEnvironmentDeployments(command, deployment, cancellationToken);

    deployment.QueueDeployment();

    await _deploymentRepository.AddAsync(deployment, cancellationToken);
    await _messageBroker.SendMessage(new DeploymentQueued(deployment.Id), cancellationToken);

    return Unit.Value;
}
```

Rys. 4.17 Kod klasy *CreateDeploymentCommandHandler*

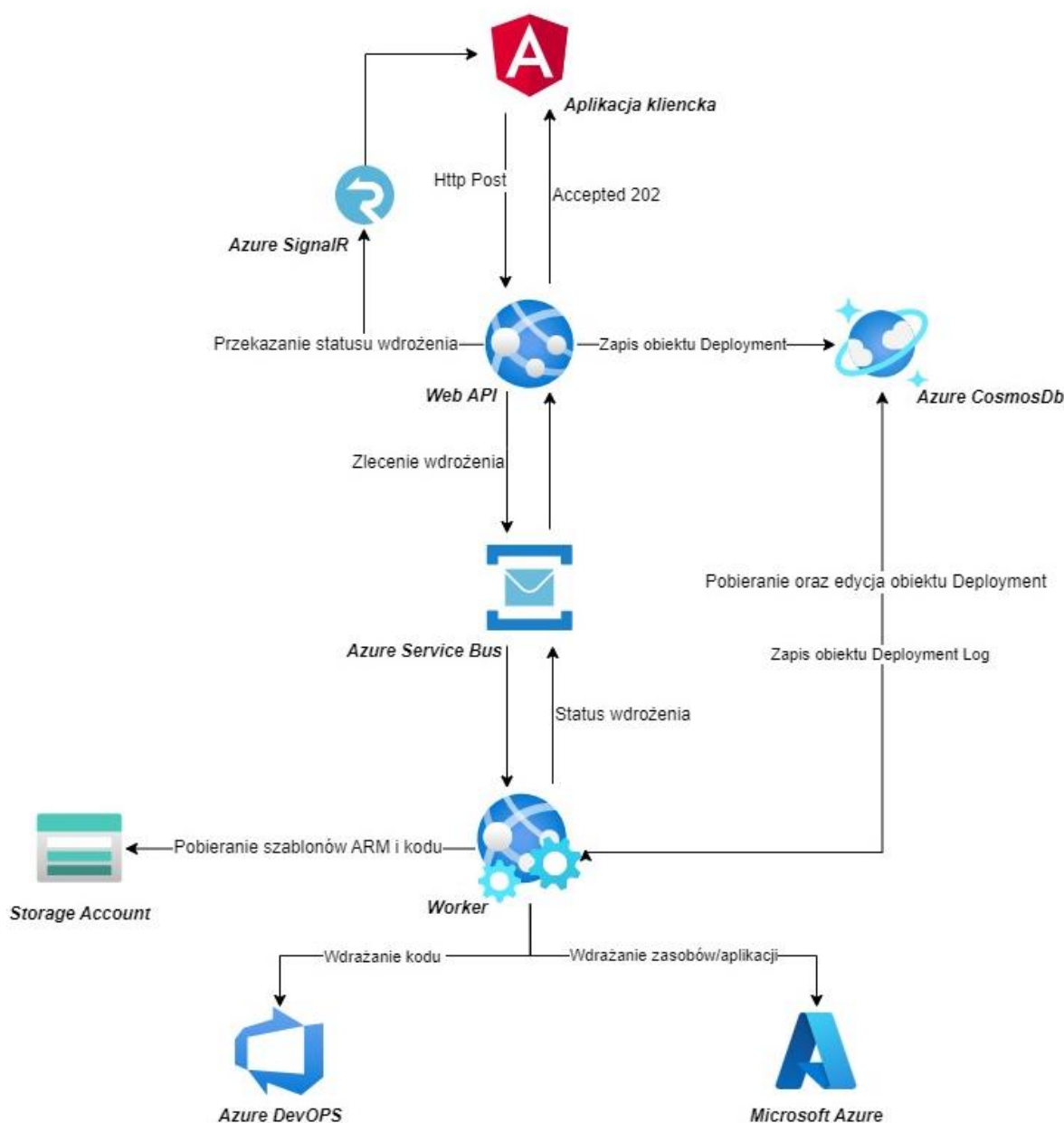
Wiadomość zostaje wysłana do *Workera*, który za pomocą unikalnego identyfikatora przesłanego w wiadomości pobiera obiekt *Deployment* z bazy danych, a następnie wykonuje wdrożenie w chmurze Microsoft Azure oraz w usłudze Azure DevOps. Worker po wykonaniu każdej z czynności wdrożeniowej wysyła wiadomość statusową do Web API poprzez *Azure Service Bus*. Rys. 4.18 przedstawia przykładową wiadomość statusową informującą o poprawnym utworzeniu zasobu.

```
public class ResourceDeploymentSucceeded : DeploymentEvent
{
    public ResourceDeploymentSucceeded(ResourceDeployment resource, string env, Guid deploymentId)
    {
        Id = resource.CloudIdentifier;
        Url = resource.Url;
        Environment = env;
        DeploymentId = deploymentId;
        Name = resource.Name;
        Message = $"Resource: '{Name}' deployment succeeded.";
        Status = DeploymentStatus.OperationSucceeded;
    }
}
```

Rys. 4.18 Przykładowa wiadomość statusowa

⁴⁴ Odpowiedź HTTP, która oznacza, że żądanie klienta zostało przyjęte do przetworzenia, ale jeszcze nie zostało ukończone.

Web API po odczytaniu wiadomości statusowej przesyła tę wiadomość poprzez usługę *Azure SignalR* do aplikacji klienckiej, która w czasie rzeczywistym uaktualnia przebieg czasowy wdrożenia w oknie *Deployment Details*. Schemat procesu wdrożenia przedstawiono na Rys. 4.19.



Rys. 4.19 Schemat procesu wdrażania⁴⁵

4.5 Wynik wdrożenia

Przyjmując, iż użytkownik w oknie dialogowym w sekcji *Deployment* przycisnął przycisk *Deploy* dla niezmienionego diagramu wdrożeniowego z podrozdziału 4.2 oraz wybrał

⁴⁵ Opracowanie własne

dwa środowiska *dev* i *test*, aplikacja utworzyła automatycznie dwie grupy zasobów przedstawione na Rys 4.20.

Resource groups ✨ ...

Katalog domyślny (fogielwiwitedu.onmicrosoft.com)

+ Create ⚙️ Manage view ▾ ↻ Refresh ⬇️ Export to CSV 🔗 Open query | 🏷️ Assign tags

rg-praca Subscription equals all Location equals all ✕ + Add filter

🛡️ 0 Unsecure resources 🌐 0 Recommendations No grouping ▾

☰ List view ▾

<input type="checkbox"/> Name ↑↓	Subscription ↑↓	Location ↑↓	Tags
<input type="checkbox"/> 🌐 rg-pracamagisterska-dev	Azure dla studentów	West Europe	CostCenter: Development Appld: 1 ...
<input type="checkbox"/> 🌐 rg-pracamagisterska-test	Azure dla studentów	West Europe	CostCenter: Development Appld: 1 ...

Rys. 4.20 Utworzone grupy zasobów

Każda z utworzonych grup ma przypisane Tagi z wartościami podanymi podczas tworzenia diagramu wdrożeniowego. Dla każdej z grup zostały utworzone zasoby (Rys. 4.21), do których również przypisane zostały te same Tagi.

rg-pracamagisterska-test ✨ ☆ ...

Resource group

+ Create ⚙️ Manage view ▾ 🗑️ Delete resource group ↻ Refresh ⬇️ Export to CSV 🔗 Open query | 🏷️ Assign tags

✓ Essentials

Resources Recommendations (7)

Filter for any field... Type equals all ✕ Location equals all ✕ + Add filter

Showing 1 to 5 of 5 records. ☐ Show hidden types ⓘ No grouping ▾

<input type="checkbox"/> Name ↑↓	Type ↑↓	Location ↑↓	Tags
<input type="checkbox"/> 💡 appi-pracamagisterska-test	Application Insights	West Europe	CostCenter: Development Appld: 1
<input type="checkbox"/> 🌐 cosmospracamagisterskatest	Azure Cosmos DB account	West Europe	CostCenter: Development Appld: 1
<input type="checkbox"/> 🗝️ kv-pracamagisterska-test	Key vault	West Europe	CostCenter: Development Appld: 1
<input type="checkbox"/> 🚌 sb-pracamagisterska-test	Service Bus Namespace	West Europe	CostCenter: Development Appld: 1
<input type="checkbox"/> 📡 sigr-pracamagisterska-test	SignalR	West Europe	CostCenter: Development Appld: 1

Rys. 4.21 Wynik wdrożenia zasobów dla środowiska test

Dla każdego z środowisk w usłudze Key vault zostały utworzone odpowiednie wpisy (Rys. 4.22) wraz z zapisanymi odpowiednimi wartościami.

Home > Resource groups > rg-pracamagisterska-test > kv-pracamagisterska-test

kv-pracamagisterska-test | Secrets ☆ ...

Key vault

Search

+ Generate/Import Refresh Restore Backup View sample code Manage deleted secrets

Name	Type	Status	Expiration date
ApplicationInsightsConnectionString		✓ Enabled	
CosmosDbConnectionString		✓ Enabled	
PracaMagisterkaApi-ClientId		✓ Enabled	
PracaMagisterkaApi-ClientSecret		✓ Enabled	
PracaMagisterkaWeb--ClientId		✓ Enabled	
ServiceBusConnectionString		✓ Enabled	
SignalRConnectionString		✓ Enabled	

Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Access policies
Events
Objects
Keys

Rys. 4.22 Utworzone wpisy w usłudze Azure Key Vault

W usłudze Azure Active Directory w sekcji *App registration* utworzone zostały dla każdego śrowiska odpowiednie aplikacje (Rys 4.23). Dla aplikacji typu serwerowego dodatkowo zostały wygenerowane klucze aplikacji oraz została uzupełniona sekcja *Authorized client*. Dla aplikacji klienckiej w sekcji *Authentication* została skonfigurowana platforma *Single-page application* wraz z podanymi *Redirect Uris*.

Home > Katalog domyślny

Katalog domyślny | App registrations ☆ ...

Azure Active Directory

+ New registration Endpoints Troubleshooting Refresh Download Preview features Got feedback?

All applications Owned applications Deleted applications Applications from personal account

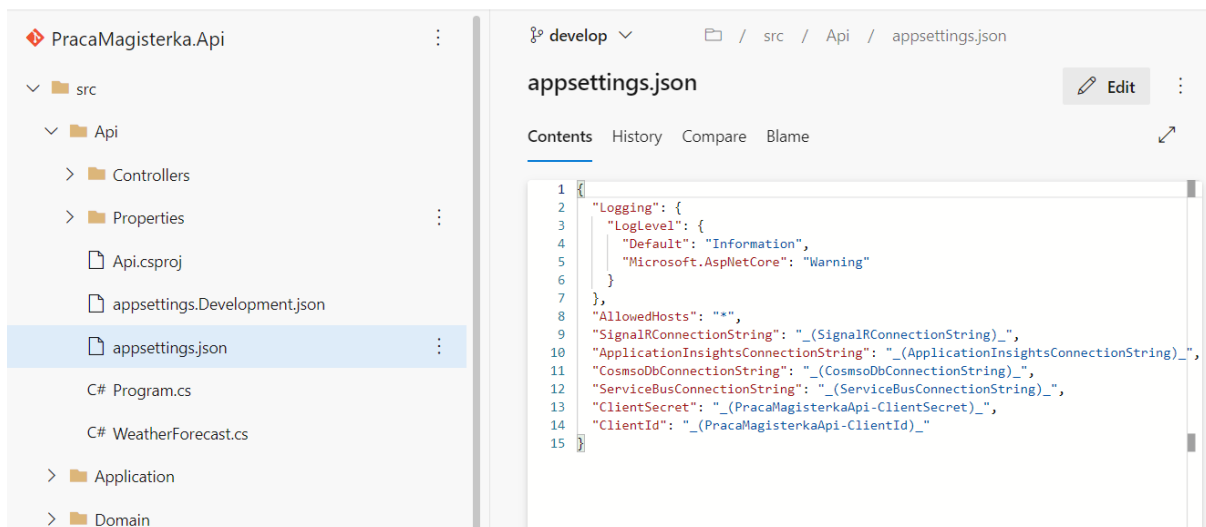
Start typing a display name or application (client) ID to filter these r... Created on : 1/9/2023 to 3/18/2023 Add filters

4 applications found

Display name ↑↓	Application (client) ID	Created on ↑↓	Certificates & secrets
PR PracaMagisterka.Apidev	ff4bfbe9-daad-4309-99ef-967567604fef	1/9/2023	✓ Current
PR PracaMagisterka.Apiatest	8616c4e7-1fdc-49fa-9768-ddf90c19c9dd	1/9/2023	✓ Current
PR PracaMagisterska.Webdev	de15abfa-f192-48d7-9af4-68e134260830	1/9/2023	-
PR PracaMagisterska.Webtest	2a627713-f186-4e8c-8ac2-fea7ee07da4a	1/9/2023	-

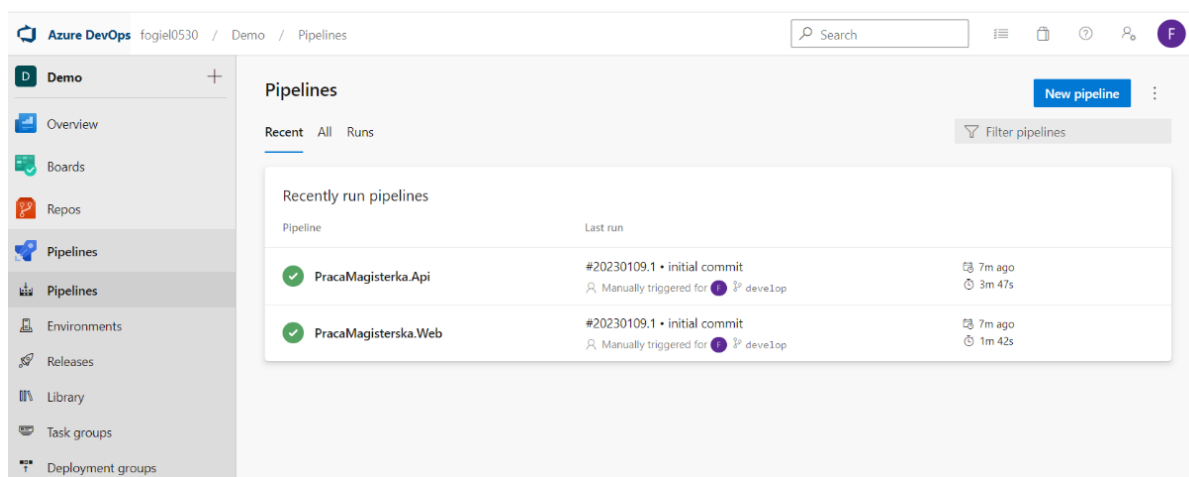
Rys. 4.23 Utworzone aplikacje w usłudze AAD

W usłudze Azure DevOps zostały utworzone dwa repozytoria kodu PracaMagisterska.Api oraz PracaMagisterska.Web. Dla repozytorium kodu PracaMagisterska.Api (Rys. 4.24) plik appsettings.json został uzupełniony poprzez dodanie nazw zdefiniowanych w oknie edycyjnym pliku konfiguracyjnego.



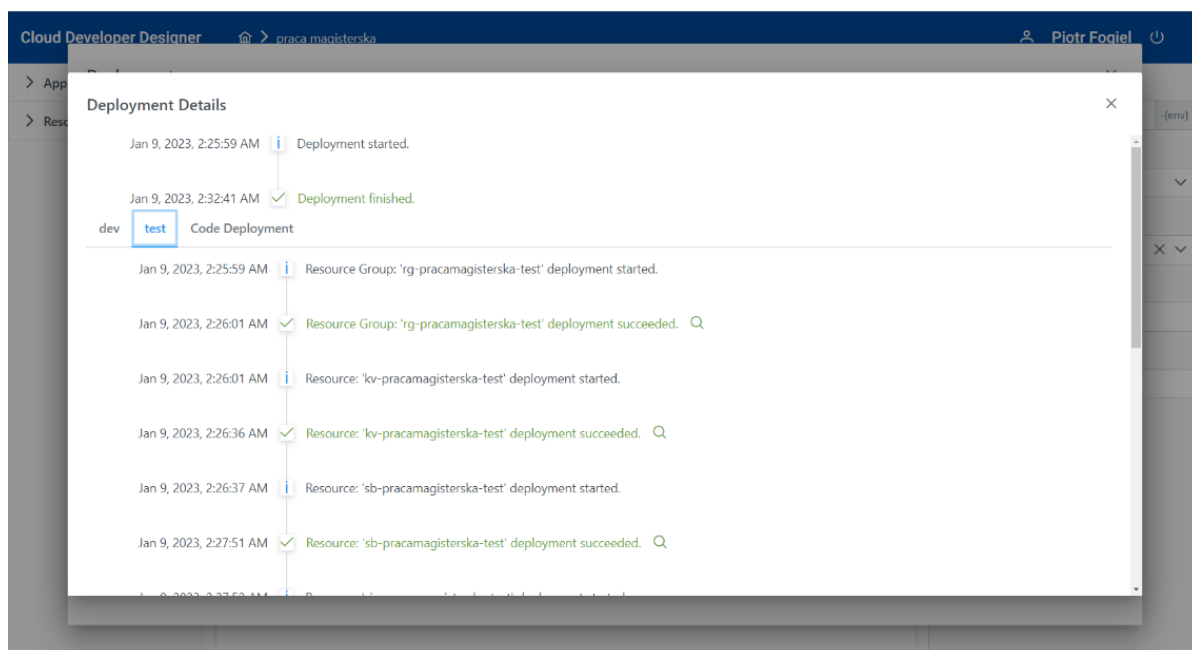
Rys. 4.24 Utworzone repozytorium kodu PracaMagisterska.Api

Dla wyżej wymienionych repozytoriów zostały utworzone oraz uruchomione procesy budowania kodu o tożsamyh nazwach (Rys. 4.25).



Rys. 4.25 Utworzone procesy budowania kodu

Cały proces trwał niespełna 7 minut, co pokazane jest na przebiegu czasowym w oknie Deployment Details (Rys. 4.26). Użytkownik w każdym momencie życia projektu może powrócić do okna by zobaczyć szczegółowe informacje dotyczące wdrożenia.



Rys. 4.26 Okno Deployment Details skończonego wdrożenia

ZAKOŃCZENIE

Jednym z problemów, z którymi borykają się programiści w swojej pracy są powtarzalne czynności, które konsumują stosunkowo dużą ilość czasu. Rozwiązaniem tego problemu jest użycie zaawansowanych narzędzi automatyzujących. Dla stosowania wspomnianych powyżej narzędzi konieczna jest specjalistyczna wiedza z zakresu ich użytkowania, co nieodłącznie wiąże się z poświęceniem czasu na jej przyswojenie. Wobec powyższego dylematu złotym środkiem jest prezentowana w ramach niniejszej pracy aplikacja *Cloud Developer Designer*, która dzięki swojej prostocie nie wymaga od jej użytkowników specjalistycznej wiedzy, jednocześnie pozwala uniknąć konieczności wykonywania powtarzających czynności.

Wprowadzenie do pracy skupia się na wyjaśnieniu znaczenia chmury obliczeniowej, opisuje chmurę Microsoft Azure, platformę Azure DevOps i najważniejsze technologie użyte w projekcie. Wyjaśniono najistotniejsze pojęcia i definicje mające bezpośredni związek z chmurą obliczeniową, a także omówiono technologie, jakie zostały wykorzystane w tworzeniu przedmiotowej aplikacji. W szczególności pochyłono się nad chmurą obliczeniową Microsoft Azure oraz platformą Azure DevOps jako newralgicznymi dla niniejszego projektu.

Omówione zostały także problemy napotymane przez programistów podczas wdrażania systemów do chmury obliczeniowej. Zobrazowano przebieg procesu tworzenia systemów informatycznych mających funkcjonować w ramach chmury Microsoft Azure, a także nakreślono pojawiające się w tym procesie trudności. Skupiono się przy tym na kwestii czasochłonności powtarzających się czynności.

Następnie poruszony został temat założeń i celów przedmiotowej aplikacji, w tym – nawiązując do powyższych rozważań – propozycja rozwiązania dla trudności, z którymi borykają się programiści w pracy. Opisana została również architektura i wzorce projektowe wykorzystane w budowie autorskiej aplikacji webowej. Ukazano także przyjętą dla niniejszego projektu koncepcję infrastruktury i usprawnień systemu. Finalnie, przedstawiono działania mające prowadzić do inicjalizacji tytułowego systemu.

W ostatniej części pracy przedstawiony został dokładny opis działania aplikacji. Wyszczególniono instrukcje, które poprzez stworzenie projektu prowadzą do utworzenia kompletnego diagramu wdrożeniowego. Szczegółowo opisano wszystkie możliwości aplikacji klienckiej wskazując na powiązania z aplikacją serwerową. Opisano również proces wdrożenia

diagramu w sposób tekstowy, jak i graficzny. Ostatnia część rozdziału skupia się również na wyniku działania systemu, który uwidacznia jego użyteczność. Użytkownik z sukcesem jest w stanie utworzyć infrastrukturę aplikacji w kilku prostych krokach, a każda z czynności wdrożeniowych jest wykonana per środowisko.

W ocenie autora założone cele zostały spełnione, a praca może być cennym źródłem informacji dla osób zainteresowanych tematyką automatyzacji w chmurze Microsoft Azure oraz dla programistów chcących usprawnić proces wdrażania swoich projektów w środowisku chmurowym.

BIBLIOGRAFIA

- 1 Canalys, *Global cloud services spend exceeds US\$50 billion in Q4 2021*, w: <https://www.canalys.com/newsroom/global-cloud-services-Q4-2021>, [dostęp: 08.03.2023]
- 2 Google, *What is Angular?*, w: <https://angular.io/guide/what-is-angular>, [dostęp: 13.03.2023]
- 3 Mell P, Grance T, *The NIST Definition of Cloud Computing*, w: <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf>, [dostęp: 19.02.2023]
- 4 Microsoft, *Azure Cosmos DB — Zapraszamy!*, w: <https://learn.microsoft.com/pl-pl/azure/cosmos-db/introduction>, [dostęp: 01.03.2023]
- 5 Microsoft, *Azure DevOps*, w: <https://azure.microsoft.com/pl-pl/products/devops/>, [dostęp: 16.01.2023]
- 6 Microsoft, *Azure REST API reference*, w: <https://learn.microsoft.com/en-us/rest/api/azure/>, [dostęp: 01.03.2023]
- 7 Microsoft, *Co to jest usługa Azure Active Directory*, w: <https://learn.microsoft.com/pl-PL/azure/active-directory/fundamentals/active-directory-what-is>, [dostęp: 11.02.2023]
- 8 Microsoft, *Co to jest Azure Portal?*, w: <https://learn.microsoft.com/pl-pl/azure/azure-portal/azure-portal-overview>, [dostęp: 01.03.2023]
- 9 Microsoft, *Co to jest Azure Service Bus*, w: <https://learn.microsoft.com/pl-pl/azure/service-bus-messaging/service-bus-messaging-overview>, [dostęp: 01.03.2023]
- 10 Microsoft, *Co to jest platforma .NET? Wprowadzenie i omówienie*, w: <https://learn.microsoft.com/pl-pl/dotnet/core/introduction>, [dostęp: 15.03.2023]
- 11 Microsoft, *Co to jest usługa Azure SignalR Service?*, w: <https://learn.microsoft.com/en-us/azure/azure-signalr/signalr-overview>, [dostęp: 15.03.2023]
- 12 Microsoft, *Co to są szablony usługi ARM?*, w: <https://learn.microsoft.com/pl-pl/azure/azure-resource-manager/templates/overview>, [dostęp: 15.03.2023]
- 13 Microsoft, *Definiowanie konwencji nazewnictwa*, w: <https://learn.microsoft.com/pl-pl/azure/cloud-adoption-framework/ready/azure-best-practices/resource-naming>, [dostęp: 16.03.2023]
- 14 Microsoft, *Dokumentacja interfejsu wiersza poleceń platformy Azure*, w: <https://learn.microsoft.com/pl-pl/cli/azure/>, [dostęp: 01.02.2023]

- 15 Microsoft, *Informacje o usłudze Azure Key Vault*, w: <https://learn.microsoft.com/pl-pl/azure/key-vault/general/overview>, [dostęp: 01.03.2023]
- 16 Microsoft, *Omówienie usługi App Service*, w: <https://learn.microsoft.com/pl-pl/azure/app-service/overview>, [dostęp: 01.03.2023]
- 17 Microsoft, *Omówienie usługi Application Insights*, w: <https://learn.microsoft.com/pl-pl/azure/azure-monitor/app/app-insights-overview>, [dostęp: 26.02.2023]
- 18 Microsoft, *Omówienie zestawu Azure SDK dla platformy .NET*, w: <https://learn.microsoft.com/pl-pl/dotnet/azure/sdk/azure-sdk-for-dotnet>, [dostęp: 15.03.2023]
- 19 Microsoft, *Organizowanie zasobów platformy Azure i hierarchii zarządzania przy użyciu tagów*, w: <https://learn.microsoft.com/pl-pl/azure/azure-resource-manager/management/tag-resources?tabs=json>, [dostęp: 21.01.2023]
- 20 Microsoft, *Plan usługi Azure App Service — omówienie*, w: <https://learn.microsoft.com/pl-pl/azure/app-service/overview-hosting-plans>, [dostęp: 26.02.2023],
- 21 Microsoft, *Rozpoczynanie pracy z programem Azure PowerShell*, w: <https://learn.microsoft.com/en-us/powershell/azure/get-started-azureps?view=azps-9.4.0>, [dostęp: 01.02.2023]
- 22 Microsoft, *Windows Azure Platform Now Generally Available in 21 Countries*, w: <https://azure.microsoft.com/en-us/blog/windows-azure-platform-now-generally-available-in-21-countries/>, [dostęp: 09.03.2023]
- 23 Microsoft, *Wzorzec CQRS*, w: <https://learn.microsoft.com/en-us/azure/architecture/patterns/cqrs>, [dostęp: 19.02.2023]
- 24 Microsoft, *Zarządzanie grupami zasobów platformy Azure przy użyciu witryny Azure Portal*, w: <https://learn.microsoft.com/pl-pl/azure/azure-resource-manager/management/manage-resource-groups-portal>, [dostęp: 15.02.2023]
- 25 Palermo J., *The Onion Architecture : part 1*, w: <https://jeffreypalermo.com/2008/07/the-onion-architecture-part-1/>, [dostęp 09.02.2023]

WYKAZ ILUSTRACJI

Rys. 1.1 Tworzenie Resource Group z poziomu Portalu Azure.....	16
Rys. 1.2 Przykładowy szablon ARM	18
Rys. 1.3 Wdrażanie szablonu ARM z poziomu .NET Azure SDK	19
Rys. 2.1 Proces dodawania autoryzowanych aplikacji.....	22
Rys. 2.2 Wygenerowany klucz aplikacji	23
Rys. 2.3 Zapisany klucz aplikacji w usłudze Key Vault	23
Rys. 2.4 Adresy przekierowań dla aplikacji typu SPA	24
Rys. 2.5 Przykładowa konfiguracja sieci wirtualnej dla usługi <i>Storage account</i>	25
Rys. 2.6 Przykładowe tagi dla grupy zasobów	26
Rys. 2.7 Przykładowe klucze dostępowe do usługi Azure Storage Account	27
Rys. 2.8 Przykładowa solucja kodu źródłowego	27
Rys. 2.9 Utworzone repozytorium kodu w usłudze Azure DevOps.....	28
Rys. 2.10 Tworzenie procesu budowania kodu na podstawie pliku YAML.....	28
Rys. 3.1 Onion Architecture - powiązanie między projektami	30
Rys. 3.2 Schemat wzorca CQRS	31
Rys. 3.3 Architektura aplikacji klienckiej	33
Rys. 3.4 Infrastruktura aplikacji	34
Rys. 3.5 Utworzone tożsamości aplikacji.....	34
Rys. 3.6 Uprawnienia aplikacji serwerowej	35
Rys. 3.7 Nadane uprawnienie Contributor dla subskrypcji.....	35
Rys. 3.8 Tworzenie PAT w Azure DevOps.....	36
Rys. 3.9 Obiekt konfiguracyjny.....	38
Rys. 4.1 Okno dialogowe dodawania projektu.....	39
Rys. 4.2 Lista kafelków zawierających informacje o projektach	40
Rys. 4.3 Obszar roboczy	41
Rys. 4.4 Rozwijalne listy wyboru środowisk i lokalizacji	42
Rys. 4.5 Pola tekstowe do wprowadzania wartości Tag-ów	42
Rys. 4.6 Proces definiowania aplikacji klienckiej.....	43
Rys. 4.7 Proces definiowania aplikacji serwerowej	44
Rys. 4.8 Wypełnianie definicji zasobu Azure Service Bus	45
Rys. 4.9 Szablon ARM zasobu Azure Service Bus.....	46

Rys. 4.10 Przykładowy diagram wdrożeniowy	47
Rys. 4.11 Edytowanie pliku konfiguracyjnego aplikacji serwerowej	47
Rys. 4.12 Zakładka Deployment okna wdrożeniowego	48
Rys. 4.13 Zmiana parametru zasobu dla poszczególnego środowiska.....	49
Rys. 4.14 Zakładka Previous Deployments okna wdrożeniowego	50
Rys. 4.15 Przykładowa sekcja Deployment Details	51
Rys. 4.16 Akcja Deploy w kontrolerze Deployment.....	51
Rys. 4.17 Kod klasy CreateDeploymentCommandHandler	52
Rys. 4.18 Przykładowa wiadomość statusowa	52
Rys. 4.19 Schemat procesu wdrażania	53
Rys. 4.20 Utworzone grupy zasobów	54
Rys. 4.21 Wynik wdrożenia zasobów dla środowiska test.....	54
Rys. 4.22 Utworzone wpisy w usłudze Azure Key Vault	55
Rys. 4.23 Utworzone aplikacje w usłudze AAD	55
Rys. 4.24 Utworzone repozytorium kodu PracaMagisterska.Api	56
Rys. 4.25 Utworzone procesy budowania kodu	56
Rys. 4.26 Okno Deployment Details skończonego wdrożenia	57

WYKAZ TABEL

Tab. 1.1 Opis użytych usług chmury Microsoft Azure	13
Tab. 3.1 Opis kontenerów bazy CosmosDb	36
Tab. 3.2 Opis właściwości obiektu Application	37
Tab. 3.3 Opis właściwości obiektu Resource	37

Tytuł pracy:

„Automatyzacja wdrażania środowisk w chmurze Azure z wykorzystaniem autorskiej aplikacji webowej.”

Streszczenie:

Przedmiotowa praca zawiera implementację narzędzia automatyzującego wdrażanie nowych aplikacji do chmury obliczeniowej Microsoft Azure oraz tworzenie repozytorium kodu wraz z procesem jego budowania w usłudze Azure DevOps. Założeniem przyświecającym utworzeniu owego narzędzia jest optymalizacja czasu pracy programisty poprzez wyeliminowanie powtarzalnych czynności.

Title:

„Automation of environment deployment to the Azure cloud using a proprietary application.”

Abstract:

The subject of this work includes the implementation of a tool for automating the deployment of new applications to the Microsoft Azure cloud and creating a code repository along with its building process in the Azure DevOps service. The assumption underlying the creation of this tool is to optimize the programmer's work time by eliminating repetitive tasks.