## Aplikacja CMDB

### Autorzy: Dawid Skrzypacz, Witold Mikołajczak

#### [Dokumentacja Techniczna: System CMDB](#dokumentacja-techniczna-system-cmdb)

1. [Wprowadzenie](#wprowadzenie)
2. [Opis Systemu CMDB](#opis-systemu-cmdb)
3. [Funkcjonalności Systemu](#funkcjonalności-systemu)
4. [Architektura Systemu](#architektura-systemu)
5. [Technologie Wykorzystane](#technologie-wykorzystane)
6. [Instrukcje Uruchomienia Projektu ASP.NET](#instrukcje-uruchomienia-projektu-aspnet)

### Wprowadzenie

System Configuration Management Database (CMDB) stanowi kluczowy element infrastruktury informatycznej, umożliwiając organizacjom zarządzanie i kontrolę nad składnikami oraz konfiguracją swoich zasobów IT. Niniejsza dokumentacja techniczna ma na celu przedstawienie struktury, funkcjonalności oraz technologii wykorzystanych w projekcie implementacji systemu CMDB opartego na platformie ASP.NET i bazie danych SQL Server.

### Opis Systemu CMDB

CMDB jest centralnym repozytorium danych, które przechowuje informacje dotyczące wszystkich zasobów IT w organizacji, takich jak sprzęt komputerowy, oprogramowanie, sieci, usługi oraz ich wzajemne zależności i relacje. System pozwala na zarządzanie pracownikami w firmie oraz przypisywanie do nich dostępnych zasobów, a także umożliwia gromadzenie, zarządzanie i wizualizację danych konfiguracyjnych, co pozwala administratorom na skuteczne monitorowanie i kontrolę infrastruktury IT.

### Funkcjonalności Systemu

* Zarządzanie Zasobami: CMDB umożliwia rejestrowanie, aktualizowanie i usuwanie zasobów IT, takich jak serwery, komputery, urządzenia sieciowe, oprogramowanie, usługi i aplikacje.
* Relacje Między Zasobami: System pozwala definiować i śledzić relacje między różnymi zasobami, co umożliwia kompleksowe zrozumienie infrastruktury IT i identyfikację zależności.
* Wizualizacja Konfiguracji: CMDB oferuje narzędzia do wizualizacji konfiguracji, w tym graficzne prezentacje zasobów i ich relacji, ułatwiając zrozumienie struktury systemu.
* Śledzenie Historii Konfiguracji: System przechowuje historię zmian konfiguracji, co umożliwia śledzenie ewolucji infrastruktury IT oraz przywracanie poprzednich stanów.

### Architektura Systemu

System CMDB oparty jest na architekturze klient-serwer, gdzie warstwa prezentacji jest realizowana za pomocą technologii ASP.NET, a warstwa danych przechowywana jest w bazie danych SQL Server. Komunikacja między warstwami odbywa się poprzez interfejsy programistyczne API.

### Technologie Wykorzystane

* ASP.NET: Platforma do budowy aplikacji internetowych, wykorzystywana do implementacji warstwy prezentacji systemu CMDB.
* SQL Server: System zarządzania relacyjnymi bazami danych, służący do przechowywania danych konfiguracyjnych w CMDB.
* C#: Język programowania używany w środowisku .NET Framework do tworzenia logiki biznesowej i interfejsów aplikacji.
* HTML/CSS: Technologie webowe wykorzystywane do tworzenia interfejsu użytkownika w warstwie prezentacji.
* JavaScript: Język programowania stosowany do implementacji dynamicznych funkcjonalności interfejsu użytkownika.
* REST API: Interfejs programistyczny umożliwiający komunikację między warstwą prezentacji a warstwą danych.

### Instrukcje Uruchomienia Projektu ASP.NET

1. **Pobierz kod źródłowy**: Sklonuj repozytorium projektu z [GitHub](https://github.com/dawidskrzypacz/CMDB).
2. **Otwórz projekt w Visual Studio Code**: Uruchom Visual Studio Code i otwórz folder zawierający projekt CMDB.
3. **Przywróć zależności:** Przed pierwszym uruchomieniem projektu wykonaj polecenie dotnet restore w terminalu Visual Studio Code, aby zainstalować wszystkie niezbędne zależności pakietów NuGet.

dotnet restore

1. **Zbuduj projekt:** Następnie wykonaj polecenie dotnet build, aby zbudować projekt i sprawdzić, czy nie ma błędów kompilacji.

dotnet build

1. **Uruchom aplikację:** Aby uruchomić aplikację, użyj polecenia dotnet run.

dotnet run

Aplikacja zostanie uruchomiona lokalnie i będzie dostępna pod adresem http://localhost:5271 w przeglądarce internetowej.

## Controllers

### DatabaseController

**Opis:** DatabaseController to kontroler odpowiedzialny za sprawdzanie połączenia z bazą danych SQL Azure.

**Odpowiedzialności:**

* **Metoda CheckConnection:** Metoda ta jest wywoływana przy żądaniu GET i służy do sprawdzania, czy połączenie z bazą danych może zostać nawiązane pomyślnie. Wykorzystuje ona SqlConnection do otwarcia połączenia z bazą danych przy użyciu łańcucha połączenia (connectionString). Jeśli połączenie zostanie otwarte pomyślnie, zwracany jest status OK z komunikatem "Connection successful". W przeciwnym razie zwracany jest status 500 z komunikatem "Database connection failed".

### ServersController

**Opis:** ServersController to kontroler zarządzający operacjami CRUD (Create, Read, Update, Delete) na encjach serwerów w aplikacji.

**Odpowiedzialności:**

* **Metoda Index:** Wyświetla listę serwerów z paginacją. Dane serwery są pobierane z kontekstu \_context i paginowane według podanych parametrów page i pageSize.
* **Metoda Details:** Wyświetla szczegóły dla konkretnego serwera na podstawie jego ID.
* **Metoda Create (GET i POST):** Wyświetla formularz tworzenia nowego serwera (GET) oraz przetwarza dane wprowadzone w formularzu i zapisuje nowy serwer do bazy danych (POST).
* **Metoda Edit (GET i POST):** Wyświetla formularz edycji istniejącego serwera (GET) oraz przetwarza dane z formularza i aktualizuje serwer w bazie danych (POST).
* **Metoda DeleteConfirmed:** Usuwa serwer na podstawie podanego ID.
* **Autoryzacja:** Metody tworzenia, edycji i usuwania serwerów są zabezpieczone i wymagają, aby użytkownik miał rolę "Admin" lub "ADMIN".

### UserController

**Opis:** UserController to kontroler zarządzający użytkownikami oraz ich rolami w systemie.

**Odpowiedzialności:**

* **Metoda Index:** Wyświetla listę wszystkich użytkowników.
* **Metoda Create (GET i POST):** Wyświetla formularz tworzenia nowego użytkownika (GET) oraz przetwarza dane z formularza i tworzy nowego użytkownika w systemie (POST).
* **Metoda Edit (GET i POST):** Wyświetla formularz edycji istniejącego użytkownika (GET) oraz przetwarza dane z formularza i aktualizuje dane użytkownika w systemie (POST).
* **Metoda DeleteConfirmed:** Usuwa użytkownika na podstawie jego ID.
* **Zarządzanie rolami:** Zawiera metody do zarządzania rolami użytkowników, takie jak Roles, CreateRole, AssignRole, UserRoles. Te metody umożliwiają tworzenie nowych ról, przypisywanie ról użytkownikom oraz wyświetlanie ról przypisanych do użytkownika.
* **Autoryzacja:** Większość metod w tym kontrolerze wymaga, aby użytkownik miał rolę "Admin" lub "ADMIN".

### HomeController

**Opis:** HomeController to kontroler odpowiedzialny za obsługę strony głównej oraz strony prywatności w aplikacji.

**Odpowiedzialności:**

* **Metoda Index:** Wyświetla stronę główną aplikacji.
* **Metoda Privacy:** Wyświetla stronę polityki prywatności.
* **Metoda Error:** Wyświetla stronę błędu z informacjami o aktualnym identyfikatorze żądania i identyfikatorem śledzenia w kontekście HTTP.

### PhonesController

**Opis:** PhonesController zarządza operacjami CRUD (Create, Read, Update, Delete) na encjach telefonów w aplikacji.

**Odpowiedzialności:**

* **Metoda Index:** Wyświetla listę telefonów z paginacją.

Dane telefony są pobierane z kontekstu \_context i paginowane według podanych parametrów page i pageSize.

* **Metoda Details:** Wyświetla szczegóły dla konkretnego telefonu na podstawie jego ID.
* **Metoda Create (GET i POST):** Wyświetla formularz tworzenia nowego telefonu (GET) oraz przetwarza dane wprowadzone w formularzu i zapisuje nowy telefon do bazy danych (POST).
* **Metoda Edit (GET i POST):** Wyświetla formularz edycji istniejącego telefonu (GET) oraz przetwarza dane z formularza i aktualizuje telefon w bazie danych (POST).
* **Metoda DeleteConfirmed:** Usuwa telefon na podstawie podanego ID.
* **Autoryzacja:** Metody tworzenia, edycji i usuwania telefonów są zabezpieczone i wymagają, aby użytkownik miał rolę "Admin" lub "ADMIN".

### ComputersController

**Opis:** ComputersController zarządza operacjami CRUD na encjach komputerów w aplikacji.

**Odpowiedzialności:**

* **Metoda Index:** Wyświetla listę komputerów z paginacją. Dane komputery są pobierane z kontekstu \_context i paginowane według podanych parametrów page i pageSize.
* **Metoda Details:** Wyświetla szczegóły dla konkretnego komputera na podstawie jego ID.
* **Metoda Create (GET i POST):** Wyświetla formularz tworzenia nowego komputera (GET) oraz przetwarza dane wprowadzone w formularzu i zapisuje nowy komputer do bazy danych (POST).
* **Metoda Edit (GET i POST):** Wyświetla formularz edycji istniejącego komputera (GET) oraz przetwarza dane z formularza i aktualizuje komputer w bazie danych (POST).
* **Metoda DeleteConfirmed:** Usuwa komputer na podstawie podanego ID.
* **Autoryzacja:** Metody tworzenia, edycji i usuwania komputerów są zabezpieczone i wymagają, aby użytkownik miał rolę "Admin" lub "ADMIN".

### EmployeesController

**Opis:** EmployeesController zarządza operacjami CRUD na encjach pracowników w aplikacji.

**Odpowiedzialności:**

* **Metoda Index:** Wyświetla listę pracowników z paginacją. Dane pracowników są pobierane z kontekstu \_context i paginowane według podanych parametrów page i pageSize.
* **Metoda Details:** Wyświetla szczegóły dla konkretnego pracownika na podstawie jego ID.
* **Metoda Create (GET i POST):** Wyświetla formularz tworzenia nowego pracownika (GET) oraz przetwarza dane wprowadzone w formularzu i zapisuje nowego pracownika do bazy danych (POST).
* **Metoda Edit (GET i POST):** Wyświetla formularz edycji istniejącego pracownika (GET) oraz przetwarza dane z formularza i aktualizuje pracownika w bazie danych (POST).
* **Metoda DeleteConfirmed:** Usuwa pracownika na podstawie podanego ID.
* **Autoryzacja:** Metody tworzenia, edycji i usuwania pracowników są zabezpieczone i wymagają, aby użytkownik miał rolę "Admin" lub "ADMIN".

### AccessoriesController

**Opis:** AccessoriesController zarządza operacjami CRUD na encjach akcesoriów w aplikacji.

**Odpowiedzialności:**

* **Metoda Index:** Wyświetla listę akcesoriów z paginacją. Dane akcesoriów są pobierane z kontekstu \_context i paginowane według podanych parametrów page i pageSize.
* **Metoda Details:** Wyświetla szczegóły dla konkretnego akcesorium na podstawie jego ID.
* **Metoda Create (GET i POST):** Wyświetla formularz tworzenia nowego akcesorium (GET) oraz przetwarza dane wprowadzone w formularzu i zapisuje nowe akcesorium do bazy danych (POST).
* **Metoda Edit (GET i POST):** Wyświetla formularz edycji istniejącego akcesorium (GET) oraz przetwarza dane z formularza i aktualizuje akcesorium w bazie danych (POST).
* **Metoda DeleteConfirmed:** Usuwa akcesorium na podstawie podanego ID.
* **Autoryzacja:** Metody tworzenia, edycji i usuwania akcesoriów są zabezpieczone i wymagają, aby użytkownik miał rolę "Admin" lub "ADMIN".