

BAZY DANYCH 2022/2023 3 EF-ZI

PROJEKT BAZY DANYCH

Dziennik wydarzeń obsługi urządzenia przemysłowego

Mateusz Sz wajkosz 168164
3 EF-ZI 2022/2023

Spis treści

1. Temat Projektu	3
2. Opis i cel projektu	3
3. Analiza założeń realizacji (Funkcjonalność projektu)	3
3.1. Informacje dotyczące prezentacji systemu	3
3.2. Analiza założeń realizacji – Logika biznesowa	3
3.3. Analiza założeń realizacji – zakres techniczny	3
3.4. Funkcje bazy danych (systemu)	4
3.5. Funkcje dodatkowe	4
4. Środowisko	4
4.1. Opis środowiska	4
4.2. Specyfikacja środowiska	5
5. Struktura danych	6
5.1. Diagram bazy danych	6
5.2. Tabele	6
5.2.1. EMP.Employees	6
5.2.2. EMP.EmployeesType	6
5.2.3. DVC.Location	7
5.2.4. DVC.DevicesCategory	7
5.2.5. DVC.Devices	7
5.2.6. DVC.DevicesWork	7
5.2.7. EVN.EventsType	7
5.2.8. EVN.EventLogs	8
5.2.9. EVN.ServiceOperations	8
5.2.10. EVN.ServiceOperationsStatus	8
6. Zakładanie bazy danych	8
7. Zarządzanie bazą danych	9
7.1. Automatyczne tworzenie kopii bezpieczeństwa	9
7.2. Role w bazie danych	10
7.3. Użytkownicy w bazie danych	11
8. Obiekty programowalne bazy danych	11
8.1. Wyzwalacze (Triggery)	11
8.1.1. TRG_addServiceOperations	11
8.2. Procedury	12
8.2.1. SP_ServiceOperationForOperator	12
8.2.2. SP_DeviceLogsAllTime	13

8.2.3.	SP_DevicesWorkForOperator	14
9.	Integracja z platformą Low-Code nAxiom	15
10.	Zapytania SQL w nAxiom	16
10.1.	Zapytania generujące listy.....	16
10.1.1.	Lista ActiveServiceOperation.....	16
10.1.2.	Lista CategoryList.....	16
10.1.3.	Lista DevicesList	17
10.1.4.	Lista AllEventList	17
10.1.5.	Lista EventNoService.....	18
10.2.	Zapytania w akcjach SQL poziomu formularzy	18
10.2.1.	Akcja DVCCategory_AddCategory	18
10.2.2.	Akcja DeviceAdd	18
10.2.3.	Akcja DeviceUpdate	18
10.2.4.	Akcja AddService.....	19

1. Temat Projektu

Baza danych – dziennik wydarzeń obsługi urządzenia przemysłowego

2. Opis i cel projektu

Realizowany projekt ma na celu stworzenie systemu bazodanowego oferującego obsługę dziennika zdarzeń urządzeń przemysłowych. W ramach przygotowanego rozwiązania, zaimplementowane zostaną funkcjonalności pozwalające zbierać informacje(logi) o błędach występujących w odniesieniu do urządzeń znajdujących się w przedsiębiorstwie. Informacje te będą zawierały dane dotyczące urządzenia: model, lokalizacja (budynek oraz sektor) w którym się znajduje, operator obsługujący oraz sesji w której urządzenie pracuje. Zrealizowana zostanie obsługa zgłoszeń serwisowych które będą integralną częścią systemu – generowane automatycznie na podstawie zgłaszanych logów. Sposób przygotowania modeli danych – struktur – ma na celu uzyskanie szerokiego spektrum raportowania obszaru urządzeń pracujących w przedsiębiorstwie – awaryjności, przestojów, typów usterek/błędów. Dodatkowym celem jest przystosowanie realizowanego systemu do integracji z zewnętrznym oprogramowaniem które jest wykorzystywane w przedsiębiorstwie takim jak: systemy klasy ERP, MES, MRP + APS (w zakresie dostępności zasobu, gniazda roboczego jakim są urządzenia przemysłowe w cyklu produkcji) oraz platform integrujących, platform Low-Code.

3. Analiza założeń realizacji (Funkcjonalność projektu)

3.1. Informacje dotyczące prezentacji systemu

Na potrzeby prezentacji działającego systemu – zasymulowany zostanie dział produkcyjny firmy MetalProd S.A. (przedsiębiorstwo fikcyjne). Przedsiębiorstwo zajmuje się produkcją oraz obróbką wyrobów metalurgicznych. Na stanie posiadania firmy znajduje się kilka hal produkcyjnych, maszyny do obróbki (różnych gabarytów) oraz dział serwisu (utrzymania) – Warsztat. Urządzenia spięte są systemem MES.

3.2. Analiza założeń realizacji – Logika biznesowa

Rdzenną funkcjonalnością systemu będzie zapis informacji o przychodzących błędach od urządzeń przemysłowych znajdujących się w przedsiębiorstwie. Dane te będą zapisywane w postaci logów. Każdy log dotyczył będzie pojedynczego wydarzenia a zawierał będzie w sobie dane o:

- Sesji urządzania - z którego pochodzi log
- Kodzie błędu – wysyłany przez urządzenie
- Typie błędu – zdefiniowany w systemie typ błędu (dane słownikowe)
- Czasie wystąpienia – dokładny czas zgłoszenia problemu

Dzięki relacyjnemu modelowi systemu będziemy mogli rozszerzyć te informacje o dokładne dane urządzenia, lokalizacji, operatora.

Drugą rdzenną funkcjonalnością będzie obsługa zgłoszeń serwisowych. Zgłoszenia będą odnosić się do logów, w stosunku jedno zgłoszenie, jeden log. Do każdego zgłoszenia przypisać będzie można pracownika(serwisanta). Realizacja będzie odbywać się na zasadzie procesowania statusów (od przyjęcia zgłoszenia do zakończenia lub odrzucenia). Statusy będą zdefiniowane w ramach systemu.

3.3. Analiza założeń realizacji – zakres techniczny

Na podstawie analizy logiki biznesowej do wykonania systemu wykorzystana zostanie relacyjna baza danych. Struktura danych będzie znormalizowana oraz przystosowana do łatwej integracji z zewnętrznymi systemami oraz raportowania.

Jako DBMS wybrany został Microsoft SQL Server w wersji 2019. Głównym kryterium wyboru była integracja z systemami klasy ERP oraz systemami produkcyjnymi które na rynku polskim w większości zbudowane są w oparciu o systemy bazodanowe firmy Microsoft.

3.4. Funkcje bazy danych (systemu)

Funkcje realizowane przez system:

- Zapis logów przychodzących od urzędzeń (sesji urzędzenia)
- Rejestrowanie sesji pracy urzędzeń (wiązanie urzędzenia, lokalizacji, operatora)
- Prowadzenie ewidencji urzędzeń oraz ich kategoryzacja
- Prowadzenie ewidencji lokalizacji
- Prowadzenie ewidencji operatorów oraz serwisantów – pracowników
- Obsługa zgłoszeń serwisowych – ewidencja, procesowanie
- Raportowanie obszaru produkcji w aspekcie urzędzeń przemysłowych (awaryjności, dostępności, serwisowania)

3.5. Funkcje dodatkowe

W ramach realizacji projektu wykonana została integracja bazy danych z platformą Low-Code nAxiom. Wykorzystanie takiego rozwiązania pozwoliło stworzyć realne warunki wykorzystania systemu bazodanowego. Platforma nAxiom ma za zadanie stworzyć środowisko obsługi bazy danych w zakresie dodawania, usuwania oraz aktualizacji informacji w bazie danych, procesowania zgłoszeń serwisowych.

Platforma nAxiom to środowisko Low-Code którego producentem jest firma nAxiom Sp. z o.o. (OPTeam S.A.). W projekcie wykorzystano środowisko lokalne w oparciu o licencję konsultanta(w ramach prac rozwojowych platformy). Rozwiązanie demonstracyjne.

4. Środowisko

Repozytorium projektu na github: <https://github.com/MrMatix78/PRzProjektBD>

4.1. Opis środowiska

Projekt został przygotowany w oparciu o bazę danych Microsoft SQL Server 2019 w wersji Developer – w pracy dla wersji SQL Server 2019 Standard.

Środowisko zostało zainstalowane lokalnie na systemie Windows 10.

Wykreowano dwie bazy danych – produkcyjną oraz testową/developmentalną. Obie bazy posiadają taką samą strukturę.

Na potrzeby testów utworzona została maszyna wirtualna z systemem Windows Server 2019 z zainstalowaną bazą danych SQL Server 2019 Express.

Na potrzeby wykorzystania platformy nAxiom zainstalowana została usługa Windows IIS oraz wykreowana dodatkowa baza danych platformy nAxiom.

4.2. Specyfikacja środowiska

Baza produkcyjna: PROJDB

Baza danych: Microsoft SQL Server 2019 Developer(Standard)

System Operacyjny: Windows 10 Pro

Przydzielona pamięć RAM: 4GB

Przydzielone rdzenie: 2

Baza testowa: PROJDB_TEST

Baza danych: Microsoft SQL Server 2019 Developer(Standard)

System Operacyjny: Windows 10 Pro

Przydzielona pamięć RAM: 4GB

Przydzielone rdzenie: 2

Drugie środowisko (maszyna wirtualna)

Baza danych: Microsoft SQL Server 2019 Express

System Operacyjny: Windows Server 2019

Przydzielona pamięć RAM: 1GB

Przydzielone rdzenie: 1

Platforma nAxiom:

Baza danych: Microsoft SQL Server 2019 Developer(Standard)

System Operacyjny: Windows 10 Pro

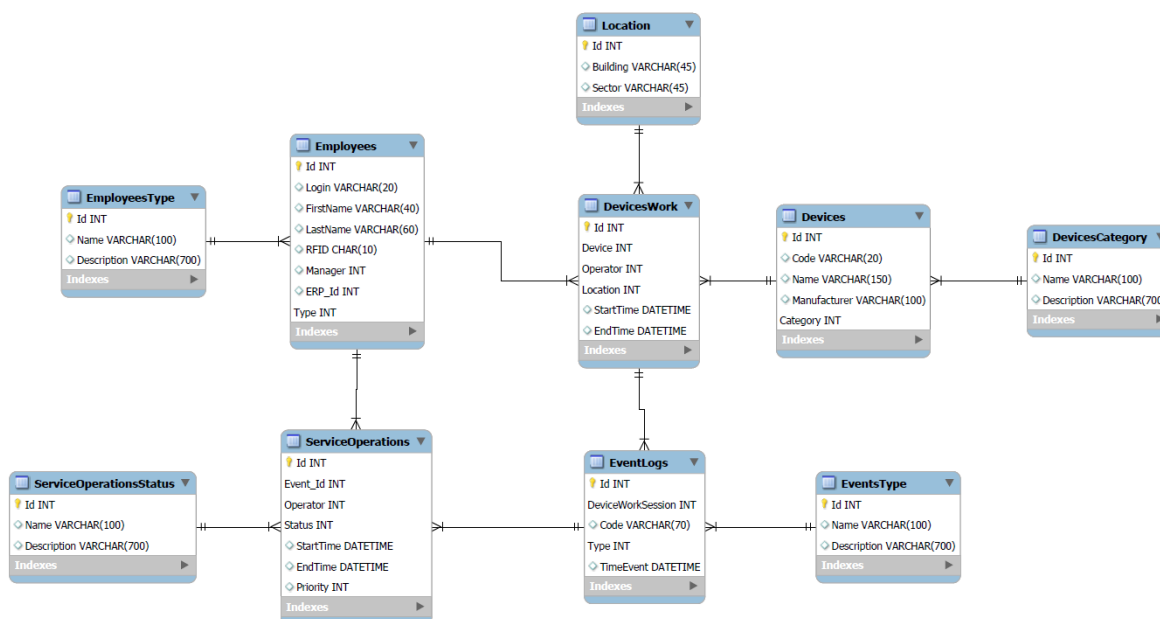
Wersja .NET: 5

Wersja nAxiom: 1.8.5

witryna: <https://naxiomenv1.ms.local/>

5. Struktura danych

5.1. Diagram bazy danych



Rysunek 1 Diagram bazy danych

[Załącznik – Diagram.pdf](#)

5.2. Tabele

5.2.1. EMP.Employees

Tabela przechowująca dane o pracownikach(operatorach) w systemie

Nazwa kolumny	Typ danych	Właściwości	Opis
Id	INT	PK	Klucz główny tabeli
Login	VARCHAR(20)	Not Null ; Unique	Login pracownika w systemie
FirstName	VARCHAR(40)	Not Null	Imię pracownika
LastName	VARCHAR(60)	Not Null	Nazwisko pracownika
RFID	VARCHAR(10)	-	Nr karty dostępowej
Manager	INT	FK	Id przełożonego (jeśli jest)
ERP_Id	INT	Not Null	Id w systemie ERP
Type	INT	FK, Not Null	FK do tabeli z typami pracowników

5.2.2. EMP.EmployeesType

Tabela przechowująca dane o typach pracowników(użytkowników)

Nazwa kolumny	Typ danych	Właściwości	Opis
Id	INT	PK	Klucz główny tabeli
Name	VARCHAR(100)	Not Null	Nazwa typu
Description	VARCHAR(700)	-	Opis typu

5.2.3. DVC.Location

Tabela przechowująca dane o lokalizacjach w przedsiębiorstwie

Nazwa kolumny	Typ danych	Właściwości	Opis
Id	INT	PK	Klucz główny tabeli
Building	VARCHAR(45)	Not Null	Nazwa budynku lokalizacji
Sector	VARCHAR(45)	Not Null	Nazwa sektora lokalizacji w budynku

5.2.4. DVC.DevicesCategory

Tabela przechowująca dane o lokalizacjach w przedsiębiorstwie

Nazwa kolumny	Typ danych	Właściwości	Opis
Id	INT	PK	Klucz główny tabeli
Name	VARCHAR(100)	Not Null	Nazwa kategorii
Description	VARCHAR(700)	-	Opis kategorii

5.2.5. DVC.Devices

Tabela przechowująca dane o urządzeniach w przedsiębiorstwie

Nazwa kolumny	Typ danych	Właściwości	Opis
Id	INT	PK	Klucz główny tabeli
Code	VARCHAR(20)	Not Null ; Unique	Kod urządzenia
Name	VARCHAR(150)	Not Null	Nazwa urządzenia
Manufacturer	VARCHAR(100)	-	Producent urządzenia
Category	INT	FK, Not Null	FK do tabeli kategorii urządzeń

5.2.6. DVC.DevicesWork

Tabela przechowująca dane o sesji pracy urządzenia

Nazwa kolumny	Typ danych	Właściwości	Opis
Id	INT	PK	Klucz główny tabeli
Device	INT	FK, Not Null	FK do tabeli urządzeń
Operator	INT	FK, Not Null	FK do tabeli pracowników
Location	INT	FK, Not Null	FK do tabeli lokalizacji
StartTime	DATETIME	Not Null	Czas rozpoczęcia sesji pracy
EndTime	DATETIME	-	Czas zakończenia sesji pracy

5.2.7. EVN.EventsType

Tabela przechowująca dane o sesji typach(rodzajach) zdarzeń

Nazwa kolumny	Typ danych	Właściwości	Opis
Id	INT	PK	Klucz główny tabeli
Name	VARCHAR(100)	Not Null	Nazwa typu
Description	VARCHAR(700)	-	Opis typu

5.2.8. EVN.EventLogs

Tabela przechowująca dane o zdarzeniach które zostały zalogowane

Nazwa kolumny	Typ danych	Właściwości	Opis
Id	INT	PK	Klucz główny tabeli
DeviceWorkSession	INT	FK, Not Null	FK do tabeli sesji pracy urzędzeń
Code	VARCHAR(70)	Not Null	Kod zgłoszenia
Type	INT	FK, Not Null	FK do tabeli z typami zgłoszeń
TimeEvent	DATETIME	Not Null	Czas zgłoszenia

5.2.9. EVN.ServiceOperations

Tabela przechowująca dane o operacjach serwisowych

Nazwa kolumny	Typ danych	Właściwości	Opis
Id	INT	PK	Klucz główny tabeli
Event_Id	INT	FK, Not Null	FK do tabeli zdarzeń
Operator	INT	FK, Not Null	FK do tabeli pracowników
Status	INT	FK, Not Null	FK do tabeli statusów serwisowych
StartTime	DATETIME	-	Czas rozpoczęcia realizacji serwisowej
EndTime	DATETIME	-	Czas zakończenia realizacji serwisowej
Priority	INT	-	Priorytet zgłoszenia w skali 1-5

5.2.10. EVN.ServiceOperationsStatus

Tabela przechowująca dane o rodzajach statusów serwisowych

Nazwa kolumny	Typ danych	Właściwości	Opis
Id	INT	PK	Klucz główny tabeli
Name	VARCHAR(100)	Not Null	Nazwa typu
Description	VARCHAR(700)	-	Opis typu

6. Zakładanie bazy danych

Baza danych tworzona jest za pomocą skryptu SQL:

```
CREATE DATABASE NAZWA_BAZY
```

Skrypt 1 SQL Tworzenie bazy

Tabele w bazie danych wraz ze schematami tworzone są za pomocą skryptu SQL. Przykładowy wycinek:

```
---Create Schema
IF NOT EXISTS ( SELECT *
                 FROM   sys.schemas
                 WHERE  name = N'EMP' )
    EXEC('CREATE SCHEMA [EMP]');
GO
```

```

----Create table: DVC.Devices
IF NOT EXISTS (SELECT * FROM sysobjects WHERE name='Devices' and
xtype='U')
CREATE TABLE DVC.Devices (
    [Id] INT PRIMARY KEY IDENTITY(1,1),
    [Code] VARCHAR(20) NOT NULL UNIQUE,
    [Name] VARCHAR(150) NOT NULL,
    [Manufacturer] VARCHAR(100),
    [Category] INT NOT NULL,

    FOREIGN KEY ([Category]) REFERENCES DVC.DevicesCategory([Id])
)

```

Skrypt 2 SQL Tworzenie schematu i tabel

Skrypt tworzenia tabel w załączniku: [CreateTablesScript.sql](#)

Dodanie typów statusów zgłoszeń (bazowe). Przykładowy wycinek:

```

INSERT INTO ServiceOperationsStatus VALUES('Nowe
zgłoszenie','Nowe zgłoszenie zarejestrowane w systemie')

```

Skrypt 3 SQL Dodanie statusu

Skrypt dodania danych w tabeli statusów w załączniku: [InsertServiceOperationsStatus.sql](#)

7. Zarządzanie bazą danych

7.1. Automatyczne tworzenie kopii bezpieczeństwa

Na potrzeby zapewnienia bezpieczeństwa danych, utworzony został mechanizm automatycznego tworzenia kopii zapasowej. Zrealizowany został za pomocą Joba SQL Server w usłudze SQL Server Agent. Proces uruchamiany jest codziennie o godzinie 11. Zapis odbywa się na dysku (nadpisywanie pliku).

Nazwa joba: ProjDB Full Backup Days

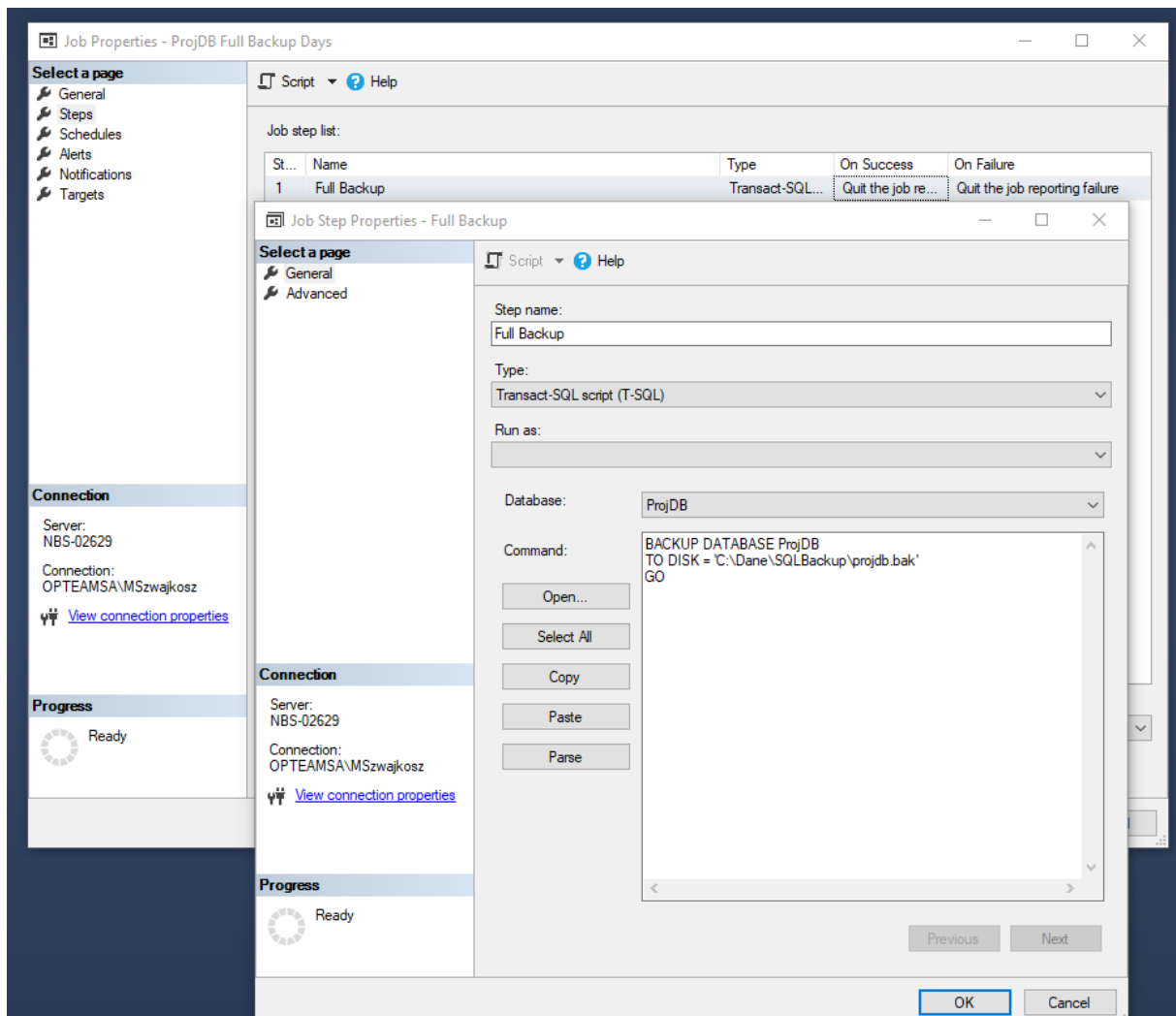
Polecenie SQL:

```

BACKUP DATABASE ProjDB
TO DISK = 'C:\Dane\SQLBackup\projdb.bak'
GO

```

Skrypt 4 SQL Stworzenie backup'u



Rysunek 2 Job backup bazy danych

7.2. Rola w bazie danych

SysDeviceOperator - rola odpowiedzialna za obsługę maszyny. Uprawniona do pracy z sesją urządzania.

SysManager – rola odpowiedzialna za zarządzanie zasobami w bazie. Uprawnienia do edycji urządzeń, kategorii, pracowników

SysServiceOperator – rola odpowiedzialna za realizację zgłoszeń serwisowych. Uprawnienia do pracy z zgłoszeniami serwisowymi.

Przykładowe polecenie tworzące rolę:

```
CREATE ROLE SysManager;
```

Skrypt 5 SQL Stworzenie roli

Przykładowe polecenie nadania uprawnień do roli:

```
GRANT SELECT ON OBJECT::DVC.Devices TO SysManager;
```

Skrypt 6 SQL Nadanie uprawnień

7.3. Użytkownicy w bazie danych

Z uwagi na integrację z platformą nAxiom stworzonych zostało 3 użytkowników z postfixem „NAXIOM” każdy użytkownik posiada przypisaną rolę w bazie danych.

DeviceOperatorNAXIOM – użytkownik odpowiadający operatorowi urządzenia. Członek roli SysDeviceOperator.

ManagerNAXIOM – użytkownik odpowiadający managerowi/kierownikowi. Członek roli SysManager

ServiceOperator – użytkownik odpowiadający serwisantowi. Członek roli SysServiceOperator

Przykładowe polecenie tworzące użytkownika i dodającego do roli:

```
CREATE USER ManagerNAXIOM FOR LOGIN ProjBDMangerNAXIOM;  
ALTER ROLE SysManager ADD MEMBER ManagerNAXIOM;
```

Skrypt 7 SQL Stworzenie użytkownika i przypisanie do roli

8. Obiekty programowalne bazy danych

8.1. Wyzwalacze (Triggery)

8.1.1. TRG_addServiceOperations

Wyzwalacz odpowiada za stworzenie zgłoszenia serwisowego o statusie 1 (nowe zgłoszenie) w momencie wystąpienia zdarzenia na urządzeniu – wprowadzeniu loga do systemu.

Skrypt tworzący: **TRG_addServiceOperation.sql**

```
CREATE TRIGGER EVN.TRG_addServiceOperation  
ON EVN.EventLogs  
AFTER INSERT  
AS  
BEGIN  
    INSERT INTO EVN.ServiceOperations(  
        Event_Id,  
        Operator,  
        Status  
    )  
    SELECT  
        i.Id,  
        1003,  
        1  
    FROM  
        inserted i  
END
```

Skrypt 8 SQL Stworzenie triggera TRG_addServiceOperations

Na potrzeby realizacji użyto operatora o id 1003 który pełni rolę nieprzypisanego serwisanta

8.2. Procedury

8.2.1. SP_ServiceOperationForOperator

Procedura wyświetla raport zrealizowanych operacji serwisowych w odniesieniu do wskazanego operatora w zadanym czasie

Parametry:

@DataStart (DATETIME) data rozpoczęcia serwisu

@DataEnd (DATETIME) data zakończenia serwisu

@Operator (VARCHAR(100)) login operatora

Skrypt tworzący: **SP_ServiceOperationForOperator.sql**

SQL:

```
CREATE PROCEDURE SP_ServiceOperationForOperator
(
    @DataStart AS DATETIME,
    @DataEnd AS DATETIME,
    @Operator AS VARCHAR(100)
)
AS
BEGIN
SET NOCOUNT ON
    SELECT
        D.Name AS 'Urządzenie'
        ,CONCAT(L.Building,' ',L.Sector) AS 'Lokalizacja'
        ,SO.StartTime AS 'Czas rozpoczęcia'
        ,SO.EndTime AS 'Czas zakończenia'
        ,CONCAT((DATEDIFF(MINUTE, SO.StartTime, ISNULL(SO.EndTime,
GETDATE()))), ' min') AS 'Czas w serwisie'
    FROM EVN.ServiceOperations SO
    INNER JOIN EVN.EventLogs EL
        ON SO.Event_Id = EL.Id
    INNER JOIN DVC.DevicesWork DW
        ON EL.DeviceWorkSession = DW.Id
    INNER JOIN DVC.Devices D
        ON DW.Device = D.Id
    INNER JOIN DVC.Location L
        ON DW.Location = L.Id
    INNER JOIN EMP.Employees E
        ON SO.Operator = E.Id
    WHERE @Operator = E.Login AND (DW.StartTime BETWEEN @DataStart
AND @DataEnd)
END
```

Skrypt 9 SQL Stworzenie procedury SP_ServiceOperationForOperator

Przykład wywołania:

EXEC dbo.SP_ServiceOperationForOperator '2022-11-01', '2022-11-30','MARWIT'

	Urządzenie	Lokalizacja	Czas rozpoczęcia	Czas zakończenia	Czas w serwisie
1	TOK Machine FM 435.54	HALA-1 A6	2022-11-04 14:51:50.000	2022-11-07 08:00:00.000	3909 min
2	Elte FGC 5 gen 1	HALA-2 B2	2022-11-05 09:50:00.000	2022-11-05 11:23:43.000	93 min
3	Skav CNC Pro 6TH	HALA-1 A6	2022-11-18 14:50:00.000	NULL	8571 min
4	TRF AnnoCNC 5	HALA-2 B1	2022-11-18 20:16:44.000	NULL	8245 min

Rysunek 3 Wynik wywołania SP_ServiceOperationForOperator

8.2.2. SP_DeviceLogsAllTime

Procedura wyświetla raport wszystkich zdarzeń wskazanego urządzenia

Parametry:

@Code (VARCHAR(20)) Kod urządzenia

Skrypt tworzący: SP_DeviceLogsAllTime.sql

SQL:

```
CREATE PROCEDURE SP_DeviceLogsAllTime
(
@CODE AS VARCHAR(20)
)
AS
BEGIN
SET NOCOUNT ON
SELECT    EL.Id AS [Event Id]
          ,D.Name AS Urządzenie
          , EL.Code AS [Kod błędu]
          , ET.Name AS [Typ błędu]
          , EL.TimeEvent AS [Data zadarzenia]
          , E.Login AS Operator
FROM EVN.EventLogs AS EL
INNER JOIN EVN.EventsType AS ET
ON EL.Type = ET.Id
INNER JOIN DVC.DevicesWork AS DW
ON EL.DeviceWorkSession = DW.Id
INNER JOIN DVC.Devices AS D
ON DW.Device = D.Id
INNER JOIN EMP.Employees AS E
ON DW.Operator = E.Id
WHERE D.Code = @CODE
END
```

Skrypt 10 SQL Stworzenie procedury SP_DeviceLogsAllTime

Przykład wywołania:

EXEC dbo.SP_DeviceLogsAllTime 'TOKFM1'

	Event Id	Urządzenie	Kod błędu	Typ błędu	Data zdarzenia	Operator
1	1	TOK Machine FM 435.54	MESFailure01	Błąd komunikacji z MES	2022-11-04 14:27:53.000	PIOWTR

Rysunek 4 Wynik wywołania SP_DeviceLogsAllTime

8.2.3. SP_DevicesWorkForOperator

Procedura wyświetla raport zawierający listę urządzeń na których pracował wskazanych operator w zadanym okresie czasu

Parametry:

@DataStart (DATETIME) data rozpoczęcia serwisu

@DataEnd (DATETIME) data zakończenia serwisu

@Operator (VARCHAR(100)) login operatora

Skrypt tworzący: SP_DeviceLogsAllTime.sql

SQL:

```
CREATE PROCEDURE SP_DeviceLogsAllTime
(
@CODE AS VARCHAR(20)
)
AS
BEGIN
SET NOCOUNT ON
SELECT EL.Id AS [Event Id]
      ,D.Name AS [Urządzenie]
      , EL.Code AS [Kod błędu]
      , ET.Name AS [Typ błędu]
      , EL.TimeEvent AS [Data zdarzenia]
      , E.Login AS Operator
FROM EVN.EventLogs AS EL
INNER JOIN EVN.EventsType AS ET
ON EL.Type = ET.Id
INNER JOIN DVC.DevicesWork AS DW
ON EL.DeviceWorkSession = DW.Id
INNER JOIN DVC.Devices AS D
ON DW.Device = D.Id
INNER JOIN EMP.Employees AS E
ON DW.Operator = E.Id
WHERE D.Code = @CODE
END
```

Skrypt 11 SQL Stworzenie procedury SP_DevicesWorkForOperator

Przykład wywołania:

EXEC dbo.SP_DevicesWorkForOperator '2022-11-01', '2022-11-30','PIOWTR'

	Urządzenie	Lokalizacja	Czas rozpoczęcia	Czas zakończenia	Czas pracy
1	TOK Machine FM 435.54	HALA-1 A6	2022-11-04 10:30:40.000	2022-11-04 12:30:40.000	120 min
2	TRF AnnoCNC 5	HALA-2 B1	2022-11-05 09:30:00.000	2022-11-05 12:30:00.000	180 min

Rysunek 5 Wynik wywołania SP_DevicesWorkForOperator

9. Integracja z platformą Low-Code nAxiom

Integracja z platformą nAxiom przeprowadza jest za pomocą utworzenia źródła danych na platformie które wskazuje na bazę danych projektu. Źródło danych typu „baza danych” wykorzystuje connection string’a który jest parametryzowany poprzez uzupełnienie danych podczas tworzenia źródła.

Utworzenie źródła pozwala nam wykonywać zapytania SQL na bazie danych oraz przekazywać w zapytaniach parametry instancji dokumentu biznesowego (formularza) za pomocą wzorca `{@PARAMETR}` przykład użycia:

```
SELECT * FROM DVC.Devices WHERE Manufacturer = {@PRODUCENT}
```

Edycja źródła danych

Nazwa:

DB_ProjBD

Aplikacja: *

ProjdbApp

Moduł: *

ReportModule

Typ źródła danych:

Baza danych: Microsoft SQL (SQLServer)

Adres serwera:

127.0.0.1

Port:

1433

Nazwa użytkownika:

nAxiomCon

Hasło:

Nazwa bazy danych:

ProjDB

Nazwa instancji:

MSSQLSERVER

Opis:

Rysunek 6 Źródło danych w nAxiom

10. Zapytania SQL w nAxiom

10.1. Zapytania generujące listy

10.1.1. Lista ActiveServiceOperation

Zapytanie wyświetla listę aktywnych zgłoszeń serwisowych.

```
SELECT
    SO.Event_Id AS 'Event_Id'
    ,SO.Priority AS 'Priorytet'
    ,CONCAT((DATEDIFF(MINUTE,EL.TimeEvent,SO.StartTime)), ' min') AS 'Czas_podjecia_serwisu'
    ,CONCAT((DATEDIFF(MINUTE, SO.StartTime,GETDATE())), ' min') AS 'Czas_w_serwisie'
    ,D.Name AS 'Urządzenie'
    ,EL.TimeEvent AS 'Data_zdarzenia'
    ,SO.StartTime AS 'Data_roz poczeczia_serwisu'
    ,E.Login AS 'Serwisant'
    ,SO.Id AS 'SO_Id'
    ,SO.Status AS 'SOStatus'
    ,SO.Operator as 'Operator'
FROM [ProjDB].EVN.ServiceOperations SO
INNER JOIN [ProjDB].EVN.EventLogs EL
    ON SO.Event_Id = EL.Id
INNER JOIN [ProjDB].DVC.DevicesWork DW
    ON EL.DeviceWorkSession = DW.Id
INNER JOIN [ProjDB].DVC.Devices D
    ON DW.Device = D.Id
INNER JOIN [ProjDB].EMP.Employees E
    ON SO.Operator = E.Id
WHERE SO.Status = 2
```

Rysunek 7 SQL Listy ActiveServiceOperation

A	Event_Id	Priorytet	Czas podjęcia serwisu	Czas w serwisie	Urządzenie	Data zdarzenia	Data rozpoczęcia serwisu	Serwisant
<input checked="" type="checkbox"/>	1002	1	30 min	8495 min	Skav CNC Pro 6TH	2022-11-18 14:20:34	2022-11-18 14:50:00	MARWIT
<input checked="" type="checkbox"/>	1004	3	573 min	8169 min	TRF AnnoCNC 5	2022-11-18 10:43:34	2022-11-18 20:16:44	MARWIT

Rysunek 8 Wynik listy ActiveServiceOperation

10.1.2. Lista CategoryList

Zapytanie wyświetla listę kategorii urządzeń.

```
SELECT * FROM [ProjDB].[DVC].[DevicesCategory]
```

Rysunek 9 SQL Listy CategoryList

Nazwa	Opis
CNC klasa 1	Maszyny CNC klasy 1 - wartość powyżej 100k
CNC klasa 2	Maszyny CNC klasy 2 - wartość powyżej 250k
CNC klasa 3	Maszyny CNC klasy 3 - wartość powyżej 500k





Rysunek 10 Wynik listy CategoryList

10.1.3. Lista DevicesList

Zapytanie wyświetla listę urządzeń w przedsiębiorstwie.

```
SELECT
    DV.Id,
    DV.Code,
    DV.Name,
    DV.Manufacturer,
    DV.Category,
    CT.Name as 'CatName'
FROM [ProjDB].[DVC].[Devices] DV
INNER JOIN [ProjDB].[DVC].[DevicesCategory] CT
    ON DV.Category = CT.Id
```

Rysunek 11 SQL Listy DevicesList

E	Kod	Nazwa	Producent	Kategoria
	TOKFM1	TOK Machine FM 435.54	ZSME	Tokarki tarczowe
	TOKELT4	Elte FGC 5 gen 1	Hitachi	CNC klasa 2
	CNCGRE4	Skav CNC Pro 6TH	Skavaria	CNC klasa 1
	TRFCNC5	TRF AnnoCNC 5	TRF Technology	CNC klasa 3

Rysunek 12 Wynik listy DevicesList

10.1.4. Lista AllEventList

Zapytanie wyświetla listę wszystkich logów(zdarzeń) w systemie.

```
SELECT
    [EL].[Id]
    ,[EL].[DeviceWorkSession]
    ,[EL].[Code]
    ,[EL].[Type]
    ,[EL].[TimeEvent]
    ,[ET].[Name]
FROM [ProjDB].[EVN].[EventLogs] EL
INNER JOIN [ProjDB].[EVN].[EventsType] ET
    ON [EL].[Type] = [ET].[Id]
```

Rysunek 13 SQL Listy AllEventList

Przeciągnij nagłówek kolumny i upuść go tutaj, aby pogrupować według tej kolumny				
Id zdarzenia	Id sesji	Kod błędu	Typ zdarzenia	Data zdarzenia
1	1	MESFailure01	Błąd komunikacji z MES	2022-11-04 14:27:53
2	6	CoolerError44	Awaria chłodzenia	2022-11-05 09:36:00
1002	1003	Awaria tarczy	Uszkodzenie tarczy tnącej	2022-11-18 14:20:34
1004	1008	ERGerror	Błąd komunikacji z MES	2022-11-18 10:43:34

Rysunek 14 Wynik listy AllEventList

10.1.5. Lista EventNoService

Zapytanie wyświetla listę zdarzeń bez operacji serwisowych.

```
SELECT
    EL.Id,
    EL.DeviceWorkSession,
    EL.Type,
    EL.TimeEvent,
    ET.Name,
    SO.Id as 'Soid',
    CONCAT(DATEDIFF(MINUTE, EL.TimeEvent, GETDATE()), ' min') AS 'czas_od_wystapienia'
FROM [ProjDB].[EVN].[EventLogs] EL
INNER JOIN [ProjDB].[EVN].[EventsType] ET
    ON EL.Type = ET.Id
JOIN [ProjDB].[EVN].[ServiceOperations] SO
    ON EL.Id = SO.Event_Id
WHERE SO.Status = 1
```

Rysunek 15 SQL Listy EventNoService

P	Id zdarzenia	Czas zdarzenia	Czas od wystąpienia	Nazwa	Sesja urządzenia
✖	1005	2022-11-18 21:05:43	8114 min	Awaria zasilania	1010

Rysunek 16 Wynik listy EventNoService

10.2. Zapytania w akcjach SQL poziomu formularzy

10.2.1. Akcja DVCCategory_AddCategory

Akcja dodaje(wprowadza) nową kategorię do bazy.

```
INSERT INTO [ProjDB].[DVC].[DevicesCategory](Name, Description)
VALUES({@Name},{@Description})
```

Rysunek 17 SQL akcji DVCCategory_AddCategory

10.2.2. Akcja DeviceAdd

Akcja dodaje(wprowadza) nowe urządzenie do bazy.

```
INSERT INTO [ProjDB].[DVC].[Devices](Code, Name, Manufacturer, Category)
VALUES({@DvcCode},{@Name},{@Manufacturer},{@Category})
```

Rysunek 18 SQL akcji DeviceAdd

10.2.3. Akcja DeviceUpdate

Akcja aktualizacji urządzenia

```
UPDATE [ProjDB].[DVC].[Devices]
SET Code = {@DvcCode},
    Name = {@Name},
    Manufacturer = {@Manufacturer},
    Category = {@Category}
WHERE Id = {@DvcId}
```

Rysunek 19 SQL akcji DeviceUpdate

10.2.4. Akcja AddService

Akcja dodania i aktualizacji operacji serwisowej

```
UPDATE [ProjDB].[EVN].[ServiceOperations]
SET Operator = {@Operator},
    Status = {@Status},
    Priority = {@Priority},
    EndTime = {@EndTime},
    StartTime = {@StartTime}
WHERE Id = {@Soid}
```

Rysunek 20 SQL akcji AddService