API Firmy (w oparciu o HierarchyID)
Przetwarzanie danych hierarchicznych

Tomasz Gajda

1. Założenia projektu

Celem API jest umożliwienie obsługi struktury firmy będącej strukturą **hierarchiczną**. Elementami w hierarchii będą **pracownicy** wraz z ich danymi personalnymi. Głównym założeniem projektu jest wykorzystanie typu **HierarchyID** do przechowywania wspomnianych danych hierarchicznych.

Stos technologiczny:

- C#
- Microsoft SQL Server
- Typ HierarchyID

2. Krótki wstęp teoretyczny

Do stworzenia API wykorzystam typ **HierarchyID** - używamy go, aby przedstawić pozycję w hierarchii. Kolumna typu *hierarchyid* nie reprezentuje automatycznie drzewa. Do aplikacji należy generowanie i przypisywanie wartości *hierarchyid* w taki sposób, aby pożądana relacja między wierszami była odzwierciedlona w wartościach.

3. Zakres funkcjonalności

Poprzez zestaw metod API pozwala użytkownikowi na dodawanie/usuwanie **pracowników** (w przyszłości możliwie również innych elementów tabel) oraz tworzenie wybranych raportów.

Oprócz samego **API**, w projekcie zostaną również uwzględnione testy jednostkowe, skrypty służące do utworzenia przykładowej bazy oraz aplikacja konsolowa, służąca do przedstawienia przykładu działania stworzonego **API**.

1. **Dodawanie pracownika -** AddEmployee()

   Parametry

   - id <<int>>
   - level <<string>>
   - firstName <<string>>
   - lastName <<string>>
   - position <<string>>
   - salary <<int>>

   Typ zwracany

   - <<void>>

2. **Usuwanie pracownika po id -** RemoveEmployeeById()

   Parametry

   - id <<int>>

   Typ zwracany

   - <<void>>

3. **Usuwanie pracownika po imieniu -** RemoveEmployeeByFirstName()

   Parametry

   - firstName <<string>>

   Typ zwracany

   - <<void>>

4. **Usuwanie pracownika po nazwisku -** RemoveEmployeeByLastName()

   Parametry

   - lastName <<string>>

   Typ zwracany

   - <<void>>

5. **Usuwanie pracownika po hierarchii -** RemoveEmployeeByLevel()

   Parametry

   - level <<string>>

   Typ zwracany

   - <<void>>

6. **Usuwanie wszystkich pracowników -** RemoveAllEmployees()

   Parametry

   - <<void>>

   Typ zwracany

   - <<void>>

7. **Zwróć pracownika po id -** GetEmployeeById()

   Parametry

   - id <<int>>

   Typ zwracany

   - <<Employee>>

8. **Zwróć pracownika ze specyficznym HierarchyID -** GetEmployeeByLevel()

   Parametry

   - level <<string>>

   Typ zwracany

   - <<Employee>>

9. **Zwróć pracownika po imieniu -** GetEmployeeByFirstName()

   Parametry

   - firstName <<string>>

   Typ zwracany

   - <<Employee>>

10. **Zwróć pracownika po nazwisku -** GetEmployeeByLastName()

    Parametry

    - lastName <<string>>

    Typ zwracany

    - <<Employee>>

11. **Zwróć pracownika ze specyficznym HierarchyID wraz z jego podwładnymi -** GetEmployeeWithSubordinates()

    Parametry

    - level <<string>>

    Typ zwracany

    - <<List<Employee>>>

12. **Zwróć wszystkich pracowników -** GetAllEmployees()

    Parametry

    - <<void>>

    Typ zwracany

    - <<List<Employee>>>

13. **Zwróć najwyższe wynagrodzenie -** GetMaxSalary()

    Parametry

    - <<void>>

    Typ zwracany

    - salary <<int>>

14. **Zwróć średnie wynagrodzenie -** GetAverageSalary()

    Parametry

    - <<void>>

    Typ zwracany

    - salary <<int>>

Ilość metod i ich specyfikacja ulegnie jeszcze zmianie! Jest to poglądowa próbka.