POLITECHNIKA POZNAŃSKA
WYDZIAŁ ELEKTRYCZNY
INSTYTUT AUTOMATYKI I INŻYNIERII INFORMATYCZNEJ

Konrad Dysput

PRACA DYPLOMOWA INŻYNIERSKA

Wirtualny system plików zarządzający przechowywaniem i synchronizacją danych w chmurze

Promotor: dr Andrzej Sikorski

Poznań, 2017

Spis treści

1. Wstęp 2
1.1. Wirtualny system plików 2
1.2. Cel i zakres pracy 2
2. Fukncjonalności systemu 3
2.1. Porównanie systemów plików NTFS 3
2.2. Porównanie systemów plików SMB 3
2.3. Porównanie systemów plików FAT32 3
2.4. Usługa Blob Azure Storage 3
3. Projekt wirtualnego systemu plików 4
3.1. System katalogów 4
3.2. Menadżer danych 4
3.3. Udostępniane funkcje 4
4. Implementacja systemu 5
4.1. Architektura systemu 5
4.2. Standardy architektury 6
4.3. Przepływ danych 6
4.4. Użyte biblioteki 6
4.5. Komunikacja z chmurą danych Azure 8
5. Testy, wdrożenia oraz scenariusze użycia 9
6. Zakończenie 10
Bibliografia 11

1. Wstęp

Systemy archiwizacji danych w związku z coraz większą ilością przetwarzanych informacji zyskały na ogromnej popularności.

1.1. Wirtualny system plików

1.2. Cel i zakres pracy

2. Fukncjonalności systemu

2.1. Porównanie systemów plików NTFS

2.2. Porównanie systemów plików SMB

2.3. Porównanie systemów plików FAT32

2.4. Usługa Blob Azure Storage

3. Projekt wirtualnego systemu plików

3.1. System katalogów

3.2. Menadżer danych

3.3. Udostępniane funkcje

```
1 public void CodeTemplate(templateValue value, float valueNumber)
2 {
3     var temp = 1;
4     var stringExample = "strig";
5 }
```

Listing 3.1. Code example

4. Implementacja systemu

W celu realizacji poszczególnych założeń projektowych został wykonany złożony program przy użyciu platformy .NET w wersji 4.6. Część serwerowa aplikacji została napisana w języku C# 6.0, a widoki użytkownika została stworzone przy pomocy języków SCSS, JavaScript oraz HTML. W celu zmniejszenia złożoności oraz kosztów utrzymania projektu zdecydowano się na rozbicie architektury na cztery moduły. Każdy z wyodrębnionych modułów ma za zadanie spełniać określoną czynność w architekturze systemu. Rozbicie projektu pozwala na łatwiejsze zarządzanie kodem oraz w przypadku dalszego rozwoju przez większą ilość programistów umożliwia na szybką aklimatyzacje. Zastosowanie warstw projektowych bardzo dobrze sprawdza się przy projektach dużej wielkości w przypadkach, gdy programista dąży do rozłożenia odpowiedzialności komponentów w poszczególnych modułach.

4.1. Architektura systemu

System został podzielony na cztery moduły:

Nazwa	Typ	Opis
Moduł aplikacji użytkownika	Aplikacja internetowa ASP.NET MVC	Aplikacja internetowa odpowiedzialna za sterowanie komunikacji pomiędzy interfejsem użytkownika na stronie internetowej, a logiką aplikacji.
Moduł logiki biznesowej	Biblioteka klas	Aplikacja biblioteczna odpowiedzialna za kontrolowanie przepływu informacji pomiędzy aplikacją internetową, do której użytkownik wysyła żądania, a bazą danych na której wykonywane są operacje pobrania danych niezbędnych do stworzenie i wyświetlenia widoku użytkownikowi.
Moduł bazodanowy	Biblioteka klas	Aplikacja zawierające modele struktury bazodanowych potrzebnych do stworzenia bazy danych (eng. code-first)oraz operacji na nich przy użycia języka zapytań funkcyjnych LINQ.
Moduł testów	Projekt testów jednostkowych	Aplikacja zawierająca scenariusze testowe oraz testy sprawdzające poprawność działania kodu poprzez sprawdzanie oczekiwanego wyjścia z metod.

W celu realizacji aplikacji użytkownika zastosowano wzorzec architektoniczny Model-View-Controller (MVC), który zakłada dodatkowy podział projektu ASP.NET MVC na trzy dodatkowe warstwy.

4.2. Standardy architektury

4.3. Przepływ danych

4.4. Użyte biblioteki

W celu łatwiejszego korzystania z dostępnych kolekcji danych oraz skorzystania ze stworzonych modułó

Biblioteka	Moduły	Opis
Automapper	— Moduł aplikacji użytkownika — Moduł aplikacji biznesowej	biblioteka umożliwiająca mapowanie pomiędzy obiektami różnego typu np. modeli bazodanowych na modele widoków
Unity	— Moduł aplikacji użytkownika	biblioteka umożliwiająca wstrzykiwanie struktur danych. Została użyta w związki z zastosowaniem wzorca architektonicznego odwróconego sterowania (ang. Inversion of Control), aby wstrzykiwać serwisy do przetwarzania logiki biznesowej aplikacji
Entity Framework	— Moduł aplikacji użytkownika — Moduł logiki biznesowej — Moduł dostępu do danych bazodanowych	biblioteka umożliwiająca operacje na kolekcjach danych oraz tworzenie ich.
Microsoft Identity	— Moduł aplikacji użytkownika	Biblioteka zapewniająca aplikacji użytkownika zestaw metody umożliwiających autoryzacje oraz uwierzytelnienie
Newtonsoft Json.NET	— Moduł aplikacji użytkownika	biblioteka umożliwiająca zamianę dowolnego typu obiekt na tekst w formacie JSON
Automapper	— Moduł aplikacji użytkownika — Moduł aplikacji biznesowej	biblioteka umożliwiająca mapowanie pomiędzy obiektami różnego typu np. modeli bazodanowych na modele widoków

Dodatkowo w celu łatwiejszego użytkowania wymienionych bibliotek, zostały stworzone pomocnicze atrybuty, umożliwiające w prosty oraz szybki sposób wykorzystanie funkcjonalności biblioteki. Do tej listy należy zaliczyć:

— **AutomapAttribute** - Atrybut stworzony w celu zamiany zwracanego z kontrolera z modelu bazodanowego na model widoku zaraz po przetworzeniu logiki znajdującej się w akcji.
— **AjaxAttribute** - Atrybut umożliwiający dostęp do danej akcji tylko poprzez użycie asynchronicznego zapytania AJAX (Asynchronous JavaScript and XML). Użycie atrybutu umożliwi nie wykonanie się logiki zawartej w akcji poprzez zapytanie inne niż AJAX.

4.5. Komunikacja z chmurą danych Azure

5. Testy, wdrożenia oraz scenariusze użycia

6. Zakończenie

Bibliografia

[1] Ksi??ka