Maciej Kryca

Maciej.kryca@gmail.com Indeks 7216

Myonenote

Notatnik online

Spis treści

[Dokument inicjujący projekt DIP 4](#_Toc485494409)

[Cel dokumentu 4](#_Toc485494410)

[Tło ustanowienia projektu 4](#_Toc485494411)

[Definicja projektu 4](#_Toc485494412)

[1. Cel projektu 4](#_Toc485494413)

[2. Formuła realizacji projektu 5](#_Toc485494414)

[3. zakres projektu 5](#_Toc485494415)

[4. Produkty projektu 5](#_Toc485494416)

[5. Wyłączenia 5](#_Toc485494417)

[6. Ograniczenia 5](#_Toc485494418)

[7. Założenia 6](#_Toc485494419)

[Podsumowanie kierownicze 6](#_Toc485494420)

[1. biznesowe powody rozpoczęcia projektu 6](#_Toc485494421)

[2. Korzyści 6](#_Toc485494422)

[3. Koszty 6](#_Toc485494423)

[4. Terminy 7](#_Toc485494424)

[5. Zagrożenia 7](#_Toc485494425)

[6. Wnioski 7](#_Toc485494426)

[Organizacja projektu 7](#_Toc485494427)

[Plan komunikacji: 10](#_Toc485494428)

[Plan jakości projektu 10](#_Toc485494429)

[Elementy Sterowania projektem 10](#_Toc485494430)

[1. Raporty 10](#_Toc485494431)

[2. Tolerancje 11](#_Toc485494432)

[3. Kontrola zmian 11](#_Toc485494433)

[Uzasadnienie biznesowe 12](#_Toc485494434)

[1. Opcje 12](#_Toc485494435)

[2. Korzyści 12](#_Toc485494436)

[3. Analiza kosztów i korzyści 12](#_Toc485494437)

[Ogólny plan projektu 12](#_Toc485494438)

[Podsumowanie zagrożeń na podstawie rejestru ryzyka 13](#_Toc485494439)

[Wymagania funkcjonalne i poza funkcjonalne 15](#_Toc485494440)

[Wprowadzenie 15](#_Toc485494441)

[1. Cel dokumentu 15](#_Toc485494442)

[2. Zakres produktu 15](#_Toc485494443)

[3. Definicje akronimy i skróty 15](#_Toc485494444)

[4. Omówienie dokumentu 16](#_Toc485494445)

[Ogólny opis aplikcji 16](#_Toc485494446)

[1. Kontekst funkcjonalny systemu 16](#_Toc485494447)

[2. Charakterystyka użytkowników 17](#_Toc485494448)

[3. Główne funkcje systemu 17](#_Toc485494449)

[4. Ograniczenia 17](#_Toc485494450)

[5. Założenia i zależności 18](#_Toc485494451)

[Wymagania funkcjonalne 18](#_Toc485494452)

[1. Diagram poziomu zerowego 18](#_Toc485494453)

[Funkcje systemu 20](#_Toc485494454)

[1. Diagram rejestracji, logowania i wylogowania klienta 20](#_Toc485494455)

[2. Dodawanie konta użytkownika 21](#_Toc485494456)

[3. Dodawanie notatki, linku, medium 21](#_Toc485494457)

[4. Dodawanie recenzji i ocenianie zasobów repozytorium 23](#_Toc485494458)

[Wymagania poza funkcjonalne 24](#_Toc485494459)

[1. Objętość systemów 24](#_Toc485494460)

[Projekt 25](#_Toc485494461)

[Architektura, technologia i narzędzia do implementacji oprogramowania 25](#_Toc485494462)

[Baza danych – projekt 25](#_Toc485494463)

[1. Wykaz tabel w bazie danych 25](#_Toc485494464)

[2. Wykaz pól 26](#_Toc485494465)

[3. Diagram bazy danych 30](#_Toc485494466)

[Makiety okien programu 30](#_Toc485494467)

[1. Rejestracja użytkownika w aplikacji 30](#_Toc485494468)

[2. Logowanie 31](#_Toc485494469)

[3. Notatki – lista 32](#_Toc485494470)

[4. Notatki zarządzanie 33](#_Toc485494471)

[5. Notatki dodawanie 33](#_Toc485494472)

[6. Oceń kontent 35](#_Toc485494473)

[7. Dodaj recenzje: 36](#_Toc485494474)

[Projekt Logiki 36](#_Toc485494475)

[1. Opis klasy Note (Agregate): 37](#_Toc485494476)

[2. Opis klasy medium (Agregate): 37](#_Toc485494477)

[3. Opis klasy Link (Agregate): 37](#_Toc485494478)

[4. Opis klasy ContentService (Service): 38](#_Toc485494479)

[5. OPIS KLASY UserService (SERVICE): 38](#_Toc485494480)

[Implementacja bazy danych 39](#_Toc485494481)

[Utworzenie projektu: 39](#_Toc485494482)

[Implementacja tabel: 42](#_Toc485494483)

[Implementacja tabeli Links: 42](#_Toc485494484)

[Implementacja tabeli Notes: 43](#_Toc485494485)

[Implementacja tabeli Mediums: 44](#_Toc485494486)

[Implementacja tabeli Projects: 45](#_Toc485494487)

[Implementacja tabeli Rating 46](#_Toc485494488)

[Skrypty wprowadzające dane: 46](#_Toc485494489)

[User profiles: 47](#_Toc485494490)

[Medium types: 48](#_Toc485494491)

[Implementacja logiki biznesowej 49](#_Toc485494492)

[Implementacja warstwy mapującej SQL to C# 49](#_Toc485494493)

[Implementacja warstwy obsługi EntityFramework 51](#_Toc485494494)

[Implementacja warstwy serwisowej 57](#_Toc485494495)

[Implementacja warstwy ViewModel 61](#_Toc485494496)

[Implementacja logiki WWW 63](#_Toc485494497)

[Interfejsy aplikacji 69](#_Toc485494498)

[Implementacja grafczna aplikacji: 70](#_Toc485494499)

# Dokument inicjujący projekt DIP

## Cel dokumentu

Celem „Dokumentu Inicjującego Projekt” („DIP”) jest przygotowanie podstawy zarządzania projektem oraz oceny jego wyniku. DIP po zatwierdzeniu przez Radę Projektu pełni rolę dokumentu referencyjnego. Na podstawie danych w nim zawartych ocenia się stopień realizacji założonych celów projektu. Podczas zamknięcia projektu dokonuje się porównania tego co zostało zaplanowane z tym co zostało osiągnięte. DIP jest dokumentem, który „żyje” przez cały okres realizacji projektu i może podlegać uzupełnieniom oraz zmianom np. w wyniku opracowywania poszczególnych Planów Etapów. Zapisy OPZ są nadrzędne w stosunku do DIP. W związku z powyższym są także wiążące w przypadku zidentyfikowania sprzeczności pomiędzy tymi dokumentami.

## Tło ustanowienia projektu

Firma 4Developers na przestrzeni ostatnich kilku lat, borykała się z problemem obiegiem informacji i wymiana wiedzy w zespołach programistycznych. Co wydłużało proces wytwarzania oprogramowania i podnosiło jego koszty.

4Developers postanowiło przygotować i wdrożyć system do zarządzania notatkami i obiegiem wiedzy w zespołach programistycznych. Do chwili obecnej wiedza posiadana przez zespoły zajmującym się konkretnymi projektami pozostawała w zakresie konkretnego zespołu. Nie raz zadżało się że zespół niepotrzebnie wypracowywał nowe rozwiązanie nie świadomy tego, że inny zespół dużo wcześniej zmierzył się z tym problemem.

Firma postanowiła działać i wprowadzić aplikację, która pozoliograniczyć powyższy problem i przyspieszyć pracę nad kolejnymi projektami.

## Definicja projektu

Celem projektu jest przygotowanie i implementacja systemu do zarządzania notatkami, artykułami i linkami developerów pracujących w firmie.

### Cel projektu

* MyOneNote powinien umożliwiać przechowywanie notatek, mediów i linków w jednym głównym repozytorium wiedzy.
* MyOneNote powinien umożliwiać ocenianie jakości zamieszczonych w nim notatek, linków i mediów.
* MyOneNote powinien umożliwić użytkownikowi dołączenie recenzji do artykułu linku czy medium.
* MyOneNote powinna mieć mozliwość dodawania nowych użytkowników i zarządzania nimi.
* MyOneNote powinien mieć możliwość autoryzacji użytkownikóFw.
* MyOneNote powinien pozwolić na organizowanie użytkowników w zespoły.
* MyOneNote powinienumożliwić dostęp użytkownikowi do właśnych notatek, a także do notatek, linków, mediów innych użytkowników.
* Przetestowanie aplikacji
* Przeszkolenie pracowników w obsłudze aplikacji
* Zakup zasobów informatycznych na platformie Azure

### Formuła realizacji projektu

System zarząddzania notatkami w firmie 4Developers zostanie zrealizowany z wykorzystaniem technologi Microsoft .Net Core. Firma przygotuje także synchronizację z serwerem Active Directory, zorganizuje szkolenie dla pracowników w zakresie obsługi aplikacji. Przygotuje także pełną instrukcję użytkownika i szkolenie w formie filmu dla przyszłyh pracowników.

### zakres projektu

Zadania w zakresie projektu:

* Projekt i implementacja MyOneNote
* Zakup zasobów informatycznych na platformie Azure
* Przetestowanie MyOneNote
* Szkolenie z obsługi MyOneNote
* Przygotowanie instrukcji

### Produkty projektu

Produkty wytworzone przez projekt:

* Zainstalowany i dostosowany System MyOneNote
* Zakupiony serwer aplikacyjny
* Zakupiony serwer bazodanowy
* Instrukcje użytkowników

### Wyłączenia

W ramach realizacji projektu zakłada sie wykluczenia:

* System nie będzie przechowywał materiałów w żadnych formach materialnych
* System nie będzie przechowywał pełnych danych osobowych

### Ograniczenia

Wdrożenie MyOneNote jest uwarunkowane:

* Zakończeniem zakupów zasobów informatycznych na platformie Azure
* Brakiem przerw w pracy podczas wdrożenia aplikacji

### Założenia

Zakłada się, że dyrekcja oraz pracownicy 4Developers będą wspierać wdrożenie systemu MyOneNote i zaalokuje odpowiednie zasoby i budżet.

## Podsumowanie kierownicze

### biznesowe powody rozpoczęcia projektu

Niniejszy projekt jest uruchamiany w celu wdrożenia systemu MyOneNote w firmir 4Developers. Powody wdrożenia projektu:

* Zarządzanie wiedzą
* Stworenie repozytorium informacji w firmie
* Uporządkowanie zasobów wiedzy

### Korzyści

Spodziewane korzyści po wdrożeniu systemu:

* Oszczędności
* Archiwizacja wiedzy
* Przyspieszenie prac developerskich
* Obniżenie kosztów innowacji

Poza powyższymi korzyściami uważa się że posiadaniee MyOneNote spowoduje lepszy wizerunek firmy 4Developers na światowym rynku.

### Koszty

|  |  |  |
| --- | --- | --- |
| **Lp** | **Opis** | **koszty** |
| 1 | Koszty wytworzenia | 50 000,00 zł |
| 2 | Koszty wdrożenia | 5 000,00 zł |
| 3 | Koszty związane z subskrypcjami Azure | 500,00 zł |
| 4 | Testowanie | 1 000,00 zł |
| 5 | Szkolenie pracowników | 2 000,00 zł |
| **Suma** |  | **58 500,00 zł** |

### Terminy

Rozpoczęcie projektu planowane jest na 20.03,2017r. a zakończenie 31.05.2017r.

Powyższa data oparta jest na następującym założeniu:

Wdrożenie systemu powinno nastąpić przed 3 kwartałem 2017r. w którym to rozpoczynają się prace nad kilkoma zbliżonymi do siebie projektami.

### Zagrożenia

Do chwili obecnej zarejestrowano pięć zagrożeń. Najważniejsze zagrożenia dotyczą, możliwości niedotrzymania terminów i przekroczenia kosztów projektu. Poniższa tabela zawiera dwa najpoważniejsze zagrożenia zidentyfikowane do chwili obecnej.

|  |  |  |  |
| --- | --- | --- | --- |
| **Lp.** | **Zagrożenie** | **Priorytet** | **Reakcja na zagrożenia** |
| 1 | Przekroczenie zaplanowanych terminów | duży | Przynieść ryzyko opóżnien przez przydzielenie dodatkowego zespołu deweloperów |
| 2 | Przekroczenie zakładanych kosztów | Średni | Przydzielić rezerwy |

### Wnioski

Koszty poniesione na wdrożenie projektu pozwolą na wzrost wydajności i zwiększenie zysków w dłuższym okresie. W kolejnych latach system MyOneNote będzie można zaoferować w wersji subskrypcji firma zewnętrznym.

## Organizacja projektu

Struktura zespołu zarządzania projektem przedstawia poniższy schemat:

Zespół developerski

Architekt

Główny użytkownik

Z-ca kierownika projektu

Sekretarz projektu

Członek projektu

Przewodniczący rady projektu

Kierownik projektu

Rada projektu

Sponsor

Opis ról projektu:

|  |  |  |
| --- | --- | --- |
| Lp. | Rola | Opis |
| 1 | Sponsor | Zapewnienie odpowiednich środków na realizacje projektu.  Mianowanie Przewodniczącego Rady Projektu.  Zatwierdzenie pozostałych członków RP.  Podejmowanie decyzji, które wykraczają poza uprawnienia RP. |
| 2 | Przewodniczący Rady  Projektu | Podejmowanie decyzji leżących w kompetencjach Rady (po konsultacji pozostałych członków RP).  Monitorowanie i kontrolę postępów projektu  na poziomie strategicznym (ocena końcowa etapu).  Organizowanie posiedzeń RP i  przewodniczenie im. |
| 3 | Kierownik projektu | Zatwierdzanie planów projektu.  Zatwierdzanie harmonogramu działania dla poszczególnych etapów.  Organizowanie, zarządzanie i kontrolowanie zasobów ludzkich.  Przygotowanie i planów awaryjnych dla ryzyka w projekcie.  Kontrolowanie wykonania harmonogramu.  Zatwierdzanie końca etapów projektu.  Zatwierdzanie zmian strategicznych dla projektu |
| 4 | Zastępca  kierownika projektu | Przygotowanie planu projektu.  Planowanie harmonogramu działania dla poszczególnych etapów.  Uzgadnianie i kontrolowanie wykonania Harmonogramu.  Organizowanie, zarządzanie i kontrolowanie zasobów ludzkich.  Przygotowanie planów awaryjnych dla ryzyka w projekcie.  Rozstrzyganie sprzeczności w wymaganiach, jeśli wykonawca stwierdzi takie w toku prac.  Zastępowanie Kierownika Projektu. |
| 5 | Sekretarz projektu | Raportowanie do kierownika projektu o postępie prac.  Tworzenie dokumentacji projektu i jej aktualizacji  Monitorowanie terminów dostarczania dokumentacji i dotrzymywanie ich.  Opracowanie standardów notacji dokumentacji.  Planowanie prac związanych z  opracowywaniem dokumentacji.  Współpraca z innymi uczestnikami projektu. |
| 6 | Członek projektu | Ustalanie zakresu pracy z kierownikiem  projektu.  Raportowanie o postępie pracy zespołu.  Informowanie o zagrożeniach  uniemożliwiających wykonania pracy w określonym terminie i określonym zakresie |
| 7 | Główny użytkownik | Reprezentuje użytkowników końcowych.  Odpowiada za specyfikację potrzeb użytkowników. |
| 8 | Architekt | Odpowiada za architekturę aplikacji  Integrację |
| 9 | Zespół developerów | Przygotowuje aplikację  Instrukcję użytkownika |

Skład osobowy:

* Sponsor: Krzysztof Jerzyna
* Przewodniczący Rady projektu: Jan Nowak
* Kierownik projektu: Adam Jakimik
* Zastępca kierownika projektu: Wojtek Musiał
* Sekretarz projektu: Karolina Kowala
* Członek projektu: Kamila Wąska
* Główny użytkownik: Paweł Bolec
* Architekt: Jan Komasa

## Plan komunikacji:

Plan komunikacji określa poziom i częstotliwość przekazywanych komunikatów, kto jest odpowiedzialny za ich wytworzenie i jaka jest grupa docelowa

|  |  |  |  |
| --- | --- | --- | --- |
| Rodzaj przekazu | Od | Do | Częstotliwość |
| Prezentacja  projektu | Prezentacja  projektu | Prezentacja  projektu | Jednokrotnie |
| Podsumowanie  postępów | Zastępca  kierownika  projektu | Kierownik projektu | Raz w tygodniu |
| Informacja o wdrożeniu | Architekt | Zastępca  kierownika projektu | Jednokrotnie |
| Informacja o postępach | Architekt | Zastępca  kierownika projektu | Raz w tygodniu |
| Informacje o postępach aplikacji | Zespół developerów | Architekt | Codziennie |

## Plan jakości projektu

Plan jakości projektu określa standardy, które muszą zostać spełnione w trakcie wdrażania projektu. Wszystkie produkty będą przedmiotem przeglądów jakości zgodnie ze standardami jakości. Za zapewnienie jakości w projekcie będzie odpowiedzialny kierownik projektu. Kryteria akceptacji produktów zostały zamieszczone w Załączniku nr 1 do niniejszego dokumentu.

## Elementy Sterowania projektem

### Raporty

Poniższe raporty zostały ustanowione w celu zapewnienia, że każdy poziom zarządzania ma wystarczającą i aktualną informację, żeby być w stanie monitorować i kontrolować przebieg Projektu.

|  |  |  |  |
| --- | --- | --- | --- |
| Rodzaj przekazu | Od | Do | Częstotliwość |
| Raport okresowy | Zastępcy  Kierownika  Projektu | Kierownika  Projektu | Tygodniowy |
| Przegląd jakości | Członek Projektu | Zastępcy  Kierownika  Projektu | Po wytworzeniu  produktów |
| Raport o Istotnych  Odchyleniach | Zastępca  Kierownika  Projektu | Kierownika  Projektu | Zależnie od  zdarzeń |
| Raport Końcowy  Etapu | Zastępca  Kierownika  Projektu | Kierownika  Projektu | Na koniec każdego  etapu |
| Raport Końcowy  Etapu | Zastępca  Kierownika  Projektu | Kierownika  Projektu | Na koniec projektu |
| Opis funkcjonalności wysokiego poziomu | Główny użytkownik | Zastępca  Kierownika  Projektu | Na początku każdego etapu |

Członek Projektu dostarczy informacje Zastępcy Kierownika Projektu na temat postępu prac w poszczególnych Grupach Zadań, z częstotliwością określoną w Grupie Zadań oraz o ukończeniu produktu, na koniec każdej Grupy Zadań. Kierownik Projektu jest odpowiedzialny za komunikowanie jakichkolwiek zewnętrznych Zagadnień Projektowych, które mogą mieć wpływ na Projekt. Informacja taka musi być przekazana Zastępcy Kierownika Projektu.

### Tolerancje

Następujące tolerancje zostały przyznane w projekcie:

Czas – 7 dni.

Koszty – 20 % tylko na koszty wdrożenia.

Kierownik Projektu ma prawo do zatwierdzania zwiększenia kosztów do poziomów przyznanych tolerancji. Zastępca Kierownika Projektu ma prawo do udzielenia zgody na przekroczenie planowanego poziomu kosztów do poziomu tolerancji przyznanych dla każdego etapu. Wszelkie prognozy wyjścia poza granice tolerancji będą przekazywane do Kierownika Projektu.

### Kontrola zmian

Zgodnie z metodyką PRINCE2 zmianę można zgłosić każda osoba, która jest uczestnikiem projektu i jest istotnie zainteresowana wynikami projektu. Dlatego też wniosek o wprowadzenia zmian mogą zgłaszać członkowie Rady Projektu, Architekt i członkowie zespołu developerskiego.

Zmiana zgłaszana jest przez osobę inicjującą proces w formie dokumentu

,,Wniosek o Wprowadzenie Zmiany”, który powinien zawierać następujące

informacje:

* Autora oraz datę zgłoszenia.
* Status zgłoszenia.
* Opis proponowanej zmiany, w tym uzasadnienie i kryteria jakościowe.
* Oddziaływanie zmiany.
* Ocena priorytetu zmiany

## Uzasadnienie biznesowe

### Opcje

Poddano analizie jedną opcję. Wykorzystanie istniejącej aplikacji Evernote. Opcja została odrzucona ze względu na brak dostępu do kodu źródłowego, brak możliwości wprowadzenia autorskich funkcjonalności. A także zablokowanie możliwości sprzedaży subskrypcji innym firmom.

### Korzyści

Korzyści spodziewane w efekcie wdrożenia systemu:

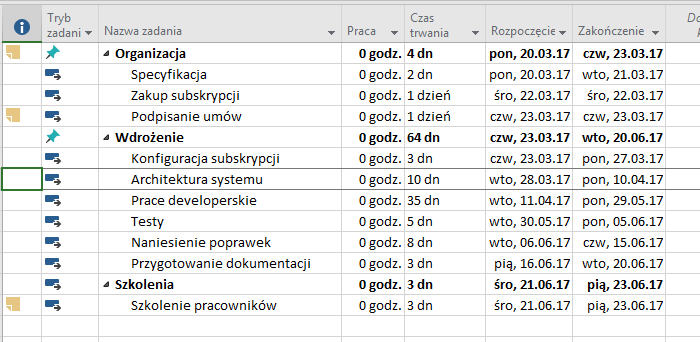
* Redukcja kosztów przy nowych projektach
* Szybka wymiana wiedzy
* Możliwość odsprzedaży subskrypcji
* Centralne repozytorium wiedzy

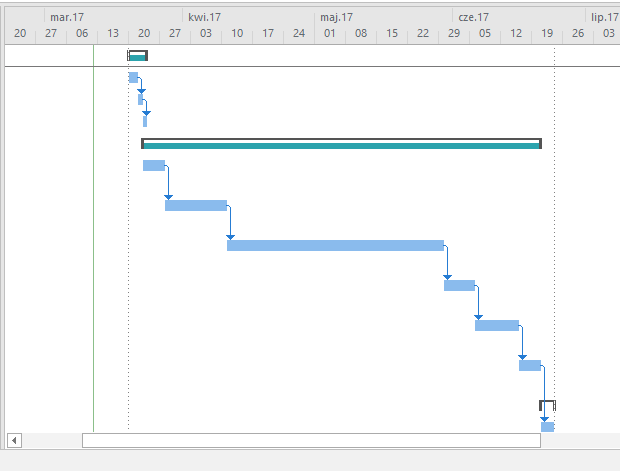
### Analiza kosztów i korzyści

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Okres** | **2017** | | **2018** | | **2019** | | **2020** | | **2021** | |
| **Opis** | **Przychód** | **Koszt** | **Przychód** | **Koszt** | **Przychód** | **Koszt** | **Przychód** | **Koszt** | **Przychód** | **Koszt** |
| Przygotowanie aplikacji |  | 50 000,00 zł |  |  |  |  |  |  |  |  |
| Wdrożenie |  | 5 000,00 zł |  |  |  |  |  |  |  |  |
| Subskrypcje |  | 500,00 zł |  | 500,00 zł |  | 500,00 zł |  | 500,00 zł |  | 500,00 zł |
| Testowanie |  | 1 000,00 zł |  |  |  |  |  |  |  |  |
| Szkolenia |  | 2 000,00 zł |  |  |  |  |  |  |  |  |
| Zaoszczędzony czas pracy | - zł |  | 10 000,00 zł |  | 20 000,00 zł |  | 20 000,00 zł |  | 20 000,00 zł |  |
| Sprzedaż dostępu | - zł |  | - zł |  | 5 000,00 zł | 1 000,00 zł | 10 000,00 zł | 2 000,00 zł | 50 000,00 zł | 5 000,00 zł |
| **Razem** | **- zł** | **58 500,00 zł** | **10 000,00 zł** | **500,00 zł** | **25 000,00 zł** | **1 500,00 zł** | **30 000,00 zł** | **2 500,00 zł** | **70 000,00 zł** | **5 500,00 zł** |

## Ogólny plan projektu

Wdrożenie projektu składa się z trzech etapów:





## Podsumowanie zagrożeń na podstawie rejestru ryzyka

Zagrożenia zarejestrowane i przeanalizowane w Rejestrze Ryzyka. Wyciąg z zagrożeń zidentyfikowanych w etapie Inicjowania Projektu przedstawia poniższa tabela:

|  |  |  |
| --- | --- | --- |
| Zagrożenie | Priorytet | Reakcja na zagrożenia |
| Przekroczenie terminów | Średni | Za alokowanie nowego zespołu developerskiego |
| Przekroczenie kosztów | Duży | Za alokowanie rezerwowych środków |
| Awarie Azure | Niski | Backup danych w AWS |
| Błędy w architekturze | Średni | Zewnętrzne konsultacje eksperckie |
| Błędne wymagania | Niski | Rozbicie implementacji na niezależne branche |

# Wymagania funkcjonalne i poza funkcjonalne

Opracowano zgodnie z wytycznymi

norm IEEE 830 oraz ISO 9126

## Wprowadzenie

### Cel dokumentu

Niniejszy dokument prezentuje wymagania dotyczące oprogramowania, czyli opisuje funkcjonalność budowanego oprogramowania i warunki, jakie ono musi spełniać. Dokument ten został napisany z myślą o odbiorcach projektu, projektantach i programistach.

### Zakres produktu

Aplikacja MyOneNote ma na celu pomóc zespołom pracowników w przechowywaniu notatek, linków i innych mediów, które wykorzystują podczas codziennej pracy i w trakcie samokształcenia. Całość ma na celu usprawnić proces wytwarzania oprogramowania, w firmie 4Developers a w późniejszym czasie ma stać się samodzielną usługą udostępnianą zewnętrznym firmą w ramach subskrypcji.

Do głównych obszarów należą:

* Utworzenie notatki, linka do strony internetowej z opisem, linka z filmem
* Możliwość dodawania recenzji do zamieszczonych mediów
* Ocenienie umieszczonych mediów
* Udostępnieni głównego repozytorium

### Definicje akronimy i skróty

|  |  |
| --- | --- |
| Pojecie | Wyjaśnienie |
| Artefakt | Dowolna grupa informacji lub fizyczny byt wykorzystywany (przetwarzany) przez czynności procesu |
| Użytkownik | Członek zespołu. Odnosi się do kontekstu procesu,  Jest to osoba zarejestrowana w aplikacji , mająca możliwość dodawania i przeglądania zasobów. |
| Repozytorium dokumentów | Odnosi się do kontekstu procesu. Jest to miejsce składowania zasobów tekstowych |
| Administrator | Osoba zarządzająca aplikacją. |
| Diagram czynności | Diagramu stanów, który obrazuje strumień kolejno wykonywanych czynności. Odnosi się do modelowania dynamicznych aspektów systemu. Kładzie nacisk na przepływ sterowania miedzy obiektami. |
| Czas reakcji | To czas, w którym system lub obiekt odpowiada reakcją na nadany sygnał |
| Diagram kontekstowy | przedstawia proces komunikacji obiektów zewnętrznych z systemem |
| Diagram przebiegu | Obrazuje kolejność przesyłania komunikatów w czasie. Przedstawia interakcje. |
| Niezawodność | Jest to własność obiektu mówiąca o tym, czy pracuje on poprawnie (spełnia wszystkie powierzone mu funkcje i czynności) przez wymagany czas i w określonych warunkach eksploatacji (w danym zespole czynników wymuszających). |
| Przypadek użycia | specyfikacja ciągu akcji i ich wariantów, które system (lub inna jednostka) może wykonać poprzez interakcję z aktorami tego systemu |

### Omówienie dokumentu

Rozdział pierwszy niniejszego dokumentu zawiera wstępne informacje dotyczące produktu oraz informacje ułatwiające poruszanie się po dokumencie, takie jak:

• omówienie celu dokumentu

• wyjaśnienie używanych skrótów

W rozdziale drugim omówiono ogólnie zastosowanie aplikacji wraz z krótką charakterystyką użytkowników i funkcjonalności, jaką będzie im udostępniał budowany system.

Rozdział trzeci jest poświęcony szczegółowemu opisowi wymagań funkcjonalnych, które zostały podzielone na grupy według funkcji systemu.

W rozdziale czwartym omówione zostały wymagania poza funkcjonalne, które uzupełniają wymagania z rozdziału trzeciego.

## Ogólny opis aplikcji

### Kontekst funkcjonalny systemu

Aplikacja MyOneNote będzie użytkowania przez pracowników firmy 4Developers z każdego działu firmy. System w początkowej fazie nie będzie zintegrowany z innymi. Kontekst funkcjonowania systemu przedstawia poniższy diagram.



Rysunek 1 DPU -Internetowy system zarzadzania wiedzą

### Charakterystyka użytkowników

Użytkownik: jest osoba dodająca media do aplikacji, mam możliwość zrecenzowania i ocenienia

Administrator: jest to osoba zarządzająca aplikacją.

### Główne funkcje systemu

Aplikacja udostępnia następujące funkcje:

1. Dodawanie notatek
2. Dodawanie Linków do stron
3. Dodawanie mediów (filmów)
4. Przeglądanie notatek innych użytkowników
5. Ocenianie notatek, linków i mediów
6. Dodawanie recenzji do notatek, mediów i linów
7. Obsługa konta użytkownika

Powyżej umieszczono najważniejsze funkcje, które będą omówione bardziej szczegółowo w dalszej części dokumentu.

### Ograniczenia

• Dane osobowe muszą być zabezpieczone zgodnie z ustawą o ochronie danych osobowych z Dz.U.1997 Nr 133 poz. 883

• Regulamin użytkowania

### Założenia i zależności

Prezentowane wymagania dotyczą stanu prawnego na dzień 01.03.2017r.

## Wymagania funkcjonalne

### Diagram poziomu zerowego



|  |  |  |
| --- | --- | --- |
| Nazwa elementu | Typ elementu | Opis |
| Użytkownik | Terminator | Będzie dodawał informację do centralnego repozytorium wiedzy |
| Dane użytkownika | Przepływ | Przepływ danych zawierające informacje o użytkowniku |
| Dane o dostępnych mediach | Przepływ | Przepływ danych zawierających informacje o przechowywanych mediach, które następnie przeglądają użytkownicy |
| Dane użytkownika | Przepływ | Informacje o zalogowanym użytkowniku aplikacji określające jego dostęp do zasobów |
| Dodawanie notatek | Proces | Dodawanie notatek do repozytorium informacji przekazywane do bazy danych mediów |
| Dodawanie mediów | Proces | Dodawanie mediów do repozytorium wiedzy przekazywane do bazy danych mediów |
| Dodawanie Linków | Proces | Dodawanie linków do repozytorium wiedzy przekazywane do bazy danych mediów |
| Dane o linkach | Przepływ | Przepływ danych zawierających informację o zapisywanych linkach |
| Dane o mediach | Przepływ | Przepływ danych zawierających informację o zapisywanych mediach |
| Dane o notatkach | Przepływ | Przepływ danych zawierających informację o zapisywanych notatkach |
| Dane o dostępnych mediach | Przepływ | Przepływ danych o posiadanych wszystkich dostępnych mediów |
| Przeglądanie zasobów | Proces | Przeglądanie posiadanych zasobów |
| Ocenianie zasobów | Proces | Ocenianie dostępnych notatek, mediów i linków |
| Dodawanie recenzji | Proces | Dodawanie recenzji do notatek mediów i linków innych użytkowników |
| Dane o recenzjach | Przepływ | Przepływ danych o dodawanych recenzjach do repozytorium |
| Dane o ocenach | Przepływ | Przepływ danych o ocenianych zasobach przez innych użytkowników |
| Baza danych użytkowników | Magazyn danych | Przechowuje informacje o użytkownikach aplikacji |
| Baza danych mediów | Magazyn danych | Przechowuje informacje o zapisanych mediach. |
| Baza danych recenzji | Magazyn danych | Zapisuje informacje o zapisanych recenzjach i ocenach wystawianych przez użytkowników |

## Funkcje systemu

### Diagram rejestracji, logowania i wylogowania klienta



|  |  |  |
| --- | --- | --- |
| Nazwa elementu | Typ elementu | Opis |
| Użytkownik | Terminator | Pracownik przeglądający notatki innych użytkowników aplikacji |
| 1.0.0 Logowanie użytkownika | Proces | Użytkownik loguje się do aplikacji |
|  | Proces | Przeglądanie zasobów repozytorium |
| 1.2.0 Wylogowanie użytkownika | Proces | Użytkownik zostaje wylogowany z aplikacji |
| Dane użytkownika | Przepływ | Przepływ danych użytkownika |
| Autentykacja użytkownika | Przepływ | Przepływ sprawdzający dane użytkownika |
| Dane logowania | Przepływ | Przepływ potwierdzający dane użytkownika |
| Żądanie wylogowania | Przepływ | Przepływ potwierdzający wylogowanie |
| Baza danych użytkowników | Magazyn danych | Przechowuje dane użytkowników |

### Dodawanie konta użytkownika



|  |  |  |
| --- | --- | --- |
| Nazwa elementu | Typ elementu | Opis |
| Administrator | Terminator | Administrator systemu |
| 2.1.0 Logowanie | Proces | Logowanie administratora systemu do aplikacji |
| 2.2.0 Nadanie roli | Proces | Nadanie roli użytkownikowi aplikacji |
| 2.3.0 Dodanie użytkownika | Proces | Dodanie nowego użytkownika do sytemu |
| 2.4.0 Wylogowanie | Proces | Wylogowanie administratora |
| Dane użytkownika | Przepływ | Dodawanie danych nowych użytkowników do bazy danych |
| Dane roli użytkownika | Przepływ | Nadanie uprawnień użytkownikom |
| Dane z formularza dodawania użytkownika | Przepływ | Pobranie danych użytkownika z formularza WWW |
| Dane z formularza nadawania roli | Przepływ | Pobierani danych nadawanych roli użytkowników |
| Baza danych użytkowników | Magazyn danych | Przechowuje dane użytkowników |

### Dodawanie notatki, linku, medium



|  |  |  |
| --- | --- | --- |
| Nazwa elementu | Typ elementu | Opis |
| Użytkownik | Terminator | Użytkownika aplikacji |
| 3.1.0 Przeglądanie zasobów | Proces | Przeglądanie repozytorium zasobów aplikacji |
| 3.2.0 Formularz dodawania notatki | Proces | Proces, w którym jest wyświetlany formularz dodawania notatki |
| 3.3.0 Formularz dodawania medium | Proces | Proces, w którym jest wyświetlany formularz dodawania medium |
| 3.4.0 Formularz dodawania linka | Proces | Proces, w którym jest wyświetlany formularz dodawania linku |
| 3.5.0 Tagowanie zasobów | Proces | Proces, dodawania tagów do mediów w celu lepszego przyporządkowania |
| Dane z formularza dodawania notatki | Przepływ | Pobranie danych z formularza dodawania notatki |
| Dane formularz dodawania medium | Przepływ | Pobranie danych z formularza dodawania medium |
| Dane z formularza dodawania linka | Przepływ | Pobranie danych z formularza dodawania linka |
| Dodawanie tagów do zasobów | Przepływ | Pobranie tagów do zasobów |
| Dane linka | Przepływ | Zapisanie danych linka |
| Dane medium | Przepływ | Zapisanie danych medium |
| Dane zasobów | Przepływ | Zapisanie danych zasobów |
| Dane notatki | Przepływ | Zapisanie danych notatek |
| Baza danych mediów | Magazyn danych | Baza danych przechowywująca repozytorium zasobów |

### Dodawanie recenzji i ocenianie zasobów repozytorium



|  |  |  |
| --- | --- | --- |
| Nazwa elementu | Typ elementu | Opis |
| Użytkownik | Terminator | Użytkownika aplikacji |
| 4.1.0 Przeglądanie zasobów repozytorium | Proces | Proces przeglądania zasobów dostępnych dla użytkownika |
| 4.2.0 Przyznawanie oceny | Proces | Proces przyznawania kryterium oceny zasobu |
| 4.3.0 Formularz dodawania recenzji | Proces | Proces opisywania recenzji użytkownika dla zasobu |
| Dane z formularza oceny | Przepływ | Pobranie oceny przypisanej do zasobu |
| Dane z formularza recenxji | Przepływ | Pobranie danych z formularza pisania recenzji |
| Rankowanie zasobów | Przepływ | Sortowanie zasobów na podstawie ocen użytkowników |
| Dane recenzji | Przepływ | Dane z formularza recenzji zapisywane do bazy danych |
| Dane oceny | Przepływ | Dane z formularza oceny zapisywane do bazy danych |
| Baza danych recenzji | Magazyn danych | Baza danych recenzji i ocen zasobów |

## Wymagania poza funkcjonalne

### Objętość systemów

* Wszystkie dane powinny być przechowywane w jednej instancji bazy danych
* Wymagania sprzętowe
  + Processor min Intel® Xeon® Processor E5-2400
  + Pamięć RAM: min 16 GB DDR3 ECC UDIMM 1333/1600
  + Dysk twardy min 2TB SSD
* Oprogramowanie
  + System operacyjny Windows Serwer 2016
  + System bazy danych MS SQL Serwer 2016
* Wydajność
  + Aplikacja powinna obsługiwać min 50 użytkowników jednocześnie
  + Aplikacja powinna umożliwić 200 operacji odczytu i zapisu jednocześnie
  + System powinien mieć możliwość rozbudowy zasobów dyskowych
  + System powinien przechowywać permanentnie zapisane dane
* Bezpieczeństwo
  + Dyski muszą być szyfrowane poprzez funkcję udostępnioną przez Windows
  + Dostęp do aplikacji na podstawie utworzonych użytkowników przez administratora

# Projekt

## Architektura, technologia i narzędzia do implementacji oprogramowania

Projektowana aplikacja MyOneNote zastanie zaimplementowana za pomocą programu Visual Studio 2017 z wykorzystaniem technologii ASP.NET Core i ASP.NET MVC 6.

Aplikacja będzie działała w oparciu o relacyjną bazę danych i dokumentową bazę danych. Jako system do zarządzania relacyjną bazą danych został wybrany SQL Server 2014. Całość aplikacji będzie oparta o technologię chmurową Azure. Całość aplikacji będzie dostępna z poziomu przeglądarki WWW.

## Baza danych – projekt

### Wykaz tabel w bazie danych

|  |  |  |  |
| --- | --- | --- | --- |
| Lp. | Nazwa tabeli | Przeznaczenie | System bazy danych |
| 1. | UserProfile | Przechowuje informacje o profilu użytkownika | MS SQL 2014 |
| 3. | Notes | Przechowuje notatki użytkowników | MS SQL 2014 |
| 4. | Links | Przechowuje linki dodawane przez użytkowników | MS SQL 2014 |
| 5. | Mediums | Przechowuje dodane media | MS SQL 2014 |
| 6. | Reviews | Przechowuje recenzje dodawane przez użytkowników | MS SQL 2014 |
| 7. | Ratings | Przechowuje oceny przechowywanych informacji w repozytorium |  |
| 8. | Scores | Słownik z dostępnymi ocenami | MS SQL 2014 |
| 9. | Tags | Przechowuje tagi | MS SQL 2014 |
| 10. | ContentReviews | Tabela łącząca recenzję z notatkami, linkami i mediami | MS SQL 2014 |
| 11. | ContentScores | Tabela łącząca ocenę z notatkami, linkami i mediami | MS SQL 2014 |
| 12. | ContentTags | Tabela łącząca tagi z notatkami, linkami i mediami | MS SQL 2014 |
| 13. | Category | Słownik kategorii | MS SQL 2014 |
| 14. | Projects | Słownik projektów | MS SQL 2014 |
| 15. | MediumTypes | Słownik typów medium | MS SQL 2014 |
| 16. | AspNetRoles | Role użytkowników | MS SQL 2014 |
| 17. | AspNetUserClaims |  | MS SQL 2014 |
| 18. | AspNetUserLogins | Loginy użytkowników | MS SQL 2014 |
| 19. | AspNetUsers | Tabela Użytkowników systemowych | MS SQL 2014 |
| 20. | AspNetUserRoles | Role użytkowników |  |

### Wykaz pól

PK – Primary key

FK – Foreign key

#### UserProfile

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Klucz | Nazwa | Typ | Zakres | Opis |
| PK | Id | Guid | Unikalny | Identyfikator profilu |
| PK | UserId | ??????? | Unikalny | Identyfikator użytkownika |
|  | Organization | Nvarchar(100) | 0-100 znaków | Nazwa organizacji |
|  | Specjalization | Nvarchar(100) | 0-100 znaków | Nazwa specjalizacji |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

#### Notes

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Klucz | Nazwa | Typ | Zakres | Opis |
| PK | Id | Guid | Unikalny | Identyfikator |
|  | Title | Nvarchar(200) | Zakres 0 -200 znaków | Temat notatki |
|  | ShortDescription | Nvarchar(200) | Zakres 0 -200 znaków | Krótki opis notatki |
|  | CreationDate | DateTime | Data | Data utworzenia notatki |
|  | ModificationDate | DateTime | Data | Data ostatniej modyfikacji |
| FK | CreateBy | Guid | Unikalny | Identyfikator użytkownika, który utworzył notatkę. |
| FK | Category | Guid | Unikalny | Identyfikator kategorii |
| FK | Project | Guid | Unikalny | Identyfikator projektu |
|  | Content | String | JSON | Treść notatki z formatowaniem |

#### Links

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Klucz | Nazwa | Typ | Zakres | Opis |
| PK | Id | Guid | Unikalny | Identyfikator |
|  | ShortDescription | Nvarchar(200) | Zakres 0 -200 znaków | Krótki opis linku |
|  | CreationDate | DateTime | Data | Data utworzenia notatki |
|  | ModificationDate | DateTime | Data | Data ostatniej modyfikacji |
| FK | CreateBy | Guid | Unikalny | Identyfikator użytkownika, który utworzył notatkę. |
| FK | Category | Guid | Unikalny | Identyfikator kategorii |
| FK | Project | Guid | Unikalny | Identyfikator projektu |
|  | Content | Nvarchar(max) | Znaki | Treść linku |
|  | WWW | Nvarchar(max) | Znaki | Strona źródłowa |

#### Medium

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Klucz | Nazwa | Typ | Zakres | Opis |
| PK | Id | Guid | Unikalny | Identyfikator |
|  | ShortDescription | Nvarchar(200) | Zakres 0 -200 znaków | Krótki opis linku |
|  | CreationDate | DateTime | Data | Data utworzenia notatki |
|  | ModificationDate | DateTime | Data | Data ostatniej modyfikacji |
| FK | CreateBy | Guid | Unikalny | Identyfikator użytkownika, który utworzył notatkę. |
| FK | Category | Guid | Unikalny | Identyfikator kategorii |
| FK | Project | Guid | Unikalny | Identyfikator projektu |
|  | Content | Nvarchar(max) | Znaki | Treść linku |
|  | WWW | Nvarchar(max) | Znaki | Strona źródłowa |
| FK | MediumType | Guid | Unikalny | Identyfikator typu medium dodawanego przez użytkownika. |

#### Reviews

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Klucz | Nazwa | Typ | Zakres | Opis |
| PK | Id | Guid | Unikalny | Identyfikator |
|  | CreationDate | DateTime | Data | Data utworzenia notatki |
|  | ModificationDate | DateTime | Data | Data ostatniej modyfikacji |
| FK | CreateBy | Guid | Unikalny | Identyfikator użytkownika, który utworzył notatkę. |
|  | Content | Nvarchar(max) | Znaki | Treść recenzji |

#### Ratings

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Klucz | Nazwa | Typ | Zakres | Opis |
| PK | Id | Guid | Unikalny | Identyfikator |
|  | CreationDate | DateTime | Data | Data utworzenia notatki |
|  | ModificationDate | DateTime | Data | Data ostatniej modyfikacji |
| FK | CreateBy | Guid | Unikalny | Identyfikator użytkownika, który utworzył notatkę. |
| FK | Score | Guid | Unikalny | Ocena contentu użytkownika |

#### Scores

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Klucz | Nazwa | Typ | Zakres | Opis |
| PK | Id | Guid | Unikalny | Identyfikator |
|  | Rate | tinyint | Liczba 0-10 | Ocena contentu użytkownika |

#### UserTags

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Klucz | Nazwa | Typ | Zakres | Opis |
| PK | TagId | Guid | Unikalny | Identyfikator |
|  | CreationDate | DateTime | Data | Data utworzenia notatki |
| FK | CreateBy | Guid | Unikalny | Identyfikator użytkownika, który utworzył notatkę. |
|  | Name | Nvarchar(200) | Zakres 0 -200 znaków | Tag dodany przez użytkownika do contentu |

#### Category

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Klucz | Nazwa | Typ | Zakres | Opis |
| PK | CategoryId | Guid | Unikalny | Identyfikator |
|  | CreationDate | DateTime | Data | Data utworzenia notatki |
| FK | CreateBy | Guid | Unikalny | Identyfikator użytkownika, który utworzył notatkę. |
|  | Name | Nvarchar(200) | Zakres 0 -200 znaków | Nazwa kategorii |

#### Projects

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Klucz | Nazwa | Typ | Zakres | Opis |
| PK | ProjectId | Guid | Unikalny | Identyfikator |
|  | CreationDate | DateTime | Data | Data utworzenia notatki |
| FK | CreateBy | Guid | Unikalny | Identyfikator użytkownika, który utworzył notatkę. |
|  | Name | Nvarchar(200) | Zakres 0 -200 znaków | Nazwa projektu |

#### MediumType

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Klucz | Nazwa | Typ | Zakres | Opis |
| PK | MediumTypeId | Guid | Unikalny | Identyfikator |
|  | CreationDate | DateTime | Data | Data utworzenia notatki |
| FK | CreateBy | Guid | Unikalny | Identyfikator użytkownika, który utworzył notatkę. |
|  | Name | Nvarchar(200) | Zakres 0 -200 znaków | Typ medium |

#### ContentReviews

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Klucz | Nazwa | Typ | Zakres | Opis |
| FK | ContentId | Guid | Unikalny | Identyfikator contentu |
| FK | ReviewId | Guid | Unikalny | Identyfikator recenzji |

#### ContentRatings

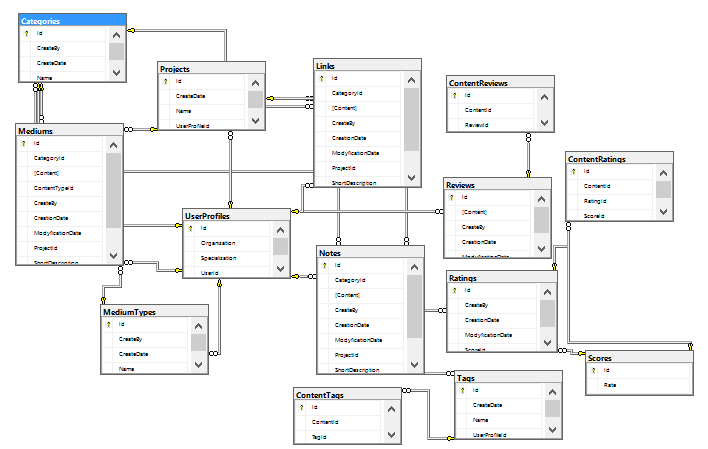
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Klucz | Nazwa | Typ | Zakres | Opis |
| FK | ContentId | Guid | Unikalny | Identyfikator contentu |
| FK | ScoresId | Guid | Unikalny | Identyfikator oceny |

#### ContentTags

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Klucz | Nazwa | Typ | Zakres | Opis |
| FK | ContentId | Guid | Unikalny | Identyfikator contentu |
| FK | TagId | Guid | Unikalny | Identyfikator Tagu |

Tabele zaczynające się od „ASP” są tabelami udostępnianymi przez mechanizm Aap.Net Identity

### Diagram bazy danych



## Makiety okien programu

### Rejestracja użytkownika w aplikacji



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lp. | Typ obiektu | Zdefinjowana treść | Pole edytowalne | Funkcja |
| 1 | Label | Email | Nie | Etykieta do pola email |
| 2 | Label | Hasło | Nie | Etykieta do pola Hasłó |
| 3 | Label | Potwierdź hasło | Nie | Etykieta do pola Potwierdż hasło |
| 4 | TextBox | Brak | Tak | Pobranie emaila |
|  | TextBox | Brak | Tak | Pobranie Hasła |
| 6 | TextBox | Brak | Tak | Pobranie potwierdzenia hasła |
| 7 | Button | Rejestruj | Nie | Wysłanie komunikatu |

### Logowanie



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lp. | Typ obiektu | Zdefinjowana treść | Pole edytowalne | Funkcja |
| 1 | Label | Email | Nie | Etykieta do pola email |
| 2 | Label | Hasło | Nie | Etykieta do pola Hasłó |
| 4 | TextBox | Brak | Tak | Pobranie emaila |
|  | TextBox | Brak | Tak | Pobranie Hasła |
| 7 | Button | Zaloguj | Nie | Wysłanie komunikatu |

### Notatki – lista



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lp. | Typ obiektu | Zdefinjowana treść | Pole edytowalne | Funkcja |
| 1 | DropDown | Notatki | Nie | Opcja nadrzędna menu |
| 2 | ActionLink | Dodaj | Nie | Link do formularza dodawania notatki |
| 3 | ActionLink | Zarządaj | Nie | Link do formularza zarządzania notatki |
| 4 | ActionLink | Dodaj | Nie | Link do listy notatek |
| 5 | DropDown | Linki | Nie | Opcja nadrzędna menu |
| 6 | DropDown | Media | Nie | Opcja nadrzędna menu |
| 7 | DropDown | Recenzje | Nie | Opcja nadrzędna menu |
| 8 | DropDown | Oceny | Nie | Opcja nadrzędna menu |
| 9 | DropDown | Tagi | Nie | Opcja nadrzędna menu |
| 10 | Label | Pobrane z bazy danych | Nie | Wyświetla tytuł notatki |
| 11 | TextArea | Pobrane z bazy danych | Nie | Wyświetla skrócony text notatki |
| 12 | TxtArea | Pobrane z bazy danych | Nie | Wyświetla ocenę |
| 13 | Button | Szczegóły | Nie | Przemosi do strony szczegułów notatki |

### Notatki zarządzanie



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lp. | Typ obiektu | Zdefinjowana treść | Pole edytowalne | Funkcja |
| 1 | Label | Pobrane z bazy danych | Nie | Wyświetla tytuł notatki |
| 2 | Label | Kategoria | Nie | Wyświetla kategorię notatki |
| 3 | Button | Usuń | Nie | Usuwa notatkę |
| 4 | Button | Edytuj | Nie | Pzenosi do formularza edytowania notatki |

### Notatki dodawanie



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lp. | Typ obiektu | Zdefinjowana treść | Pole edytowalne | Funkcja |
| 1 | TextBox | Nie | Tak | Pobranie tytułu notatki |
| 2 | DropDown | Pobrane projekty | Nie | Wybór projeku |
| 3 | DropDown | Pobrane kategorie | Nie | Wybór kategorii |
| 4 | TextArea | Nie | Tak | Pobranie treści notatki |
| 5 | Label | Krótki opis | Nie | Nagówek |
| 6 | Button | Dodaj tag | Nie | Dodaje nagłówek |
| 7 | List | Pobrane z bazy | Nie | Lista dodanych tagów |
| 8 | TextArea | Nie | Tak | Krótki opis |
| 9 | Button | Anuluj | Nie | Anuluje formularz |
| 10 | Button | Zapisz | Nie | Zapisuje formularz do bazy |

### Oceń kontent



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lp. | Typ obiektu | Zdefinjowana treść | Pole edytowalne | Funkcja |
| 1 | Label | Pobrane z bazy | Nie | Wyświetla tytuł kontentu |
| 2 | Label | Pobrane z bazy | Nie | Wyświetla typ kontentu |
| 3 | TextArea | Pobrane z bazy | Nie | Wyświetla krótki opis kontentu |
| 4 | Numeric | Nie | Tak | Pobranie oceny |
| 5 | Button | Oceń | Nie | Zapisuje ocene |
| 6 | ActionLink | Ocenione | Nie | Przenosi do listy ocenionych produktów |
| 7 | ActionLink | Oceń | Nie | Przenosi do formularza ocen |

### Dodaj recenzje:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lp. | Typ obiektu | Zdefinjowana treść | Pole edytowalne | Funkcja |
| 1 | Label | Pobrane z bazy | Nie | Wyświetla tytuł kontentu |
| 2 | TextArea | Pobrane z bazy | Nie | Wyświetla krótki opis kontentu |
| 3 | TextArea | Nie | Tak | Pobranie tytułu recenzji |
| 4 | TextArea | Nie | Tak | Pobranie treści recenzji |
| 5 | Numeric | Nie | Tak | Pobranie oceny |
| 6 | Button | Anuluj | Nie | Anuluje formularz |
| 7 | Button | Zapisz | Nie | Zapisuje recenzje |

## Projekt Logiki

### 1. Opis klasy Note (Agregate):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lp. | Nazwa | Typ | Dostęp | Przeznaczenie |
| 1 | Id | Guid | public | Identyfikator |
| 2 | Title | string | public | Temat notatki |
| 3 | ShortDescription | string | public | Krótki opis notatki |
| 4 | CreationDate | DateTime | public | Data utworzenia notatki |
| 5 | ModificationDate | DateTime | public | Data ostatniej modyfikacji |
| 6 | CreateBy | User | public | Użytkownik |
| 7 | Category | Category | public | Kategoria |
| 8 | Project | Project | public | Projekt |
| 9 | Content | String | public | Treść notatki z formatowaniem |
| 10 | GetRatings | Score | public | Pobranie aktualnej oceny notatki |
| 11 | GetReviews | IList<Review> | public | Pobranie recenzji dodanych przezużytkowników |
| 12 | GetTags | IList<Tag> | public | Pobranie tagów notatatki |

### 2. Opis klasy medium (Agregate):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lp. | Nazwa | Typ | Dostęp | Przeznaczenie |
| 1 | Id | Guid | public | Identyfikator |
| 2 | ShortDescription | string | public | Krótki opis linku |
| 3 | CreationDate | DateTime | public | Data utworzenia notatki |
| 4 | ModificationDate | DateTime | public | Data ostatniej modyfikacji |
| 5 | CreateBy | User | public | Identyfikator użytkownika, który utworzył notatkę. |
| 6 | Category | Category | public | Identyfikator kategorii |
| 7 | Project | Project | public | Identyfikator projektu |
| 8 | GetRatings | Score | public | Pobranie aktualnej oceny notatki |
| 9 | GetReviews | IList<Review> | public | Pobranie recenzji dodanych przezużytkowników |
| 10 | GetTags | IList<Tag> | public | Pobranie tagów notatatki |

### 3. Opis klasy Link (Agregate):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lp. | Nazwa | Typ | Dostęp | Przeznaczenie |
| 1 | Id | Guid | public | Identyfikator |
| 2 | ShortDescription | string | public | Krótki opis linku |
| 3 | CreationDate | DateTime | public | Data utworzenia notatki |
| 4 | ModificationDate | DateTime | public | Data ostatniej modyfikacji |
| 5 | CreateBy | User | public | Identyfikator użytkownika, który utworzył notatkę. |
| 6 | Category | Category | public | Identyfikator kategorii |
| 7 | Project | Project | public | Identyfikator projektu |
|  | 8 | Score | public | Pobranie aktualnej oceny notatki |
| 9 | GetReviews | IList<Review> | public | Pobranie recenzji dodanych przezużytkowników |
| 10 | GetTags | IList<Tag> | public | Pobranie tagów notatatki |

### 4. Opis klasy ContentService (Service):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lp. | Nazwa | Typ | Dostęp | Przeznaczenie |
| 1 | AddRating | Void | public | Metoda umozliwiająca dodanie oceny do agregatu |
| 2 | AddReview | Void | public | Metoda umożliwiająca dodanie recenzji do agregatu |
| 3 | GetUsersRatings | Ilist<Rating> | public | Metoda pobierająca oceny użytkowników |
| 4 | GetUsersRevirews | Ilist<Review> | public | Metoda pobierajaca recenzje użytkowników |
| 5 | GetTop10NoteByProject | Ilist<Note> | public | Metoda pobiera 10 najlepszych notatek według projeektu |
| 6 | GetTop10MediumByProject | Ilist<Medium> | public | Metoda pobiera 10 najlepszych mediów według projeektu |
| 7 | GetTop10LinkByProject | Ilist<Link> | public | Metoda pobiera 10 najlepszych linków według projeektu |

### 5. OPIS KLASY UserService (SERVICE):

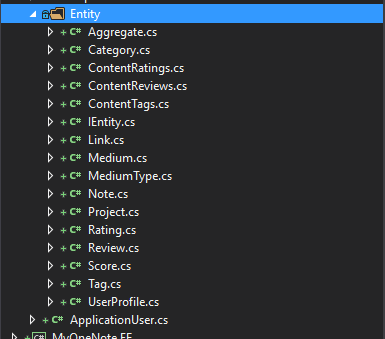
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Lp. | Nazwa | Typ | Dostęp | Przeznaczenie |
| 1 | GetUserByRating | User | public | Metoda pobiera użytkownika, który wystawił ocenę |
| 2 | GetUserByReview | User | public | Metoda pobiera użytkownika, który wystawił recenzję |
| 3 | UpdateUserProfile | Void | Public | Metoda aktualizująca dane użytkownika |

# Implementacja bazy danych

PRzy implementacji warstwy bazodanowej wykożystanow framework EntityFramework 6 przy podejściu Code First. Oznaczenie EF6 oznacza implementację w kodzie c# za pomocą frameworka, oznaczenie SQL to skrypt SQL wygenerowany poprzez framework.

## Utworzenie projektu:

Struktura bazy danych utworzona za pomocą C# i EntityFramework 6.



Skrypt SQL tworzący bazę danych:

USE [master]

GO

CREATE DATABASE [MyOneNote]

CONTAINMENT = NONE

ON PRIMARY

( NAME = N'MyOneNote', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\DATA\MyOneNote.mdf' , SIZE = 8192KB , MAXSIZE = UNLIMITED, FILEGROWTH = 65536KB )

LOG ON

( NAME = N'MyOneNote\_log', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL13.MSSQLSERVER\MSSQL\DATA\MyOneNote\_log.ldf' , SIZE = 8192KB , MAXSIZE = 2048GB , FILEGROWTH = 65536KB )

GO

ALTER DATABASE [MyOneNote] SET COMPATIBILITY\_LEVEL = 130

GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))

begin

EXEC [MyOneNote].[dbo].[sp\_fulltext\_database] @action = 'enable'

end

GO

ALTER DATABASE [MyOneNote] SET ANSI\_NULL\_DEFAULT OFF

GO

ALTER DATABASE [MyOneNote] SET ANSI\_NULLS OFF

GO

ALTER DATABASE [MyOneNote] SET ANSI\_PADDING OFF

GO

ALTER DATABASE [MyOneNote] SET ANSI\_WARNINGS OFF

GO

ALTER DATABASE [MyOneNote] SET ARITHABORT OFF

GO

ALTER DATABASE [MyOneNote] SET AUTO\_CLOSE OFF

GO

ALTER DATABASE [MyOneNote] SET AUTO\_SHRINK OFF

GO

ALTER DATABASE [MyOneNote] SET AUTO\_UPDATE\_STATISTICS ON

GO

ALTER DATABASE [MyOneNote] SET CURSOR\_CLOSE\_ON\_COMMIT OFF

GO

ALTER DATABASE [MyOneNote] SET CURSOR\_DEFAULT GLOBAL

GO

ALTER DATABASE [MyOneNote] SET CONCAT\_NULL\_YIELDS\_NULL OFF

GO

ALTER DATABASE [MyOneNote] SET NUMERIC\_ROUNDABORT OFF

GO

ALTER DATABASE [MyOneNote] SET QUOTED\_IDENTIFIER OFF

GO

ALTER DATABASE [MyOneNote] SET RECURSIVE\_TRIGGERS OFF

GO

ALTER DATABASE [MyOneNote] SET DISABLE\_BROKER

GO

ALTER DATABASE [MyOneNote] SET AUTO\_UPDATE\_STATISTICS\_ASYNC OFF

GO

ALTER DATABASE [MyOneNote] SET DATE\_CORRELATION\_OPTIMIZATION OFF

GO

ALTER DATABASE [MyOneNote] SET TRUSTWORTHY OFF

GO

ALTER DATABASE [MyOneNote] SET ALLOW\_SNAPSHOT\_ISOLATION OFF

GO

ALTER DATABASE [MyOneNote] SET PARAMETERIZATION SIMPLE

GO

ALTER DATABASE [MyOneNote] SET READ\_COMMITTED\_SNAPSHOT OFF

GO

ALTER DATABASE [MyOneNote] SET HONOR\_BROKER\_PRIORITY OFF

GO

ALTER DATABASE [MyOneNote] SET RECOVERY FULL

GO

ALTER DATABASE [MyOneNote] SET MULTI\_USER

GO

ALTER DATABASE [MyOneNote] SET PAGE\_VERIFY CHECKSUM

GO

ALTER DATABASE [MyOneNote] SET DB\_CHAINING OFF

GO

ALTER DATABASE [MyOneNote] SET FILESTREAM( NON\_TRANSACTED\_ACCESS = OFF )

GO

ALTER DATABASE [MyOneNote] SET TARGET\_RECOVERY\_TIME = 60 SECONDS

GO

ALTER DATABASE [MyOneNote] SET DELAYED\_DURABILITY = DISABLED

GO

ALTER DATABASE [MyOneNote] SET QUERY\_STORE = OFF

GO

USE [MyOneNote]

GO

ALTER DATABASE SCOPED CONFIGURATION SET MAXDOP = 0;

GO

ALTER DATABASE SCOPED CONFIGURATION FOR SECONDARY SET MAXDOP = PRIMARY;

GO

ALTER DATABASE SCOPED CONFIGURATION SET LEGACY\_CARDINALITY\_ESTIMATION = OFF;

GO

ALTER DATABASE SCOPED CONFIGURATION FOR SECONDARY SET LEGACY\_CARDINALITY\_ESTIMATION = PRIMARY;

GO

ALTER DATABASE SCOPED CONFIGURATION SET PARAMETER\_SNIFFING = ON;

GO

ALTER DATABASE SCOPED CONFIGURATION FOR SECONDARY SET PARAMETER\_SNIFFING = PRIMARY;

GO

ALTER DATABASE SCOPED CONFIGURATION SET QUERY\_OPTIMIZER\_HOTFIXES = OFF;

GO

ALTER DATABASE SCOPED CONFIGURATION FOR SECONDARY SET QUERY\_OPTIMIZER\_HOTFIXES = PRIMARY;

GO

ALTER DATABASE [MyOneNote] SET READ\_WRITE

GO

## Implementacja tabel:

### Implementacja tabeli Links:

EF6:

public class Link:Aggregate

   {

       public string ShortDescription { get; set; }

       public Category Category { get; set; }

       public Guid CategoryId { get; set; }

       public string Content { get; set; }

       public Project Project { get; set; }

       public Guid ProjectId { get; set; }

       public string Url { get; set; }

   }

SQL:

CREATE TABLE [dbo].[Links](

[Id] [uniqueidentifier] NOT NULL,

[CategoryId] [uniqueidentifier] NOT NULL,

[Content] [nvarchar](max) NOT NULL,

[CreateBy] [uniqueidentifier] NOT NULL,

[CreationDate] [datetime2](7) NOT NULL,

[ModyficationDate] [datetime2](7) NOT NULL,

[ProjectId] [uniqueidentifier] NOT NULL,

[ShortDescription] [nvarchar](250) NULL,

[Url] [nvarchar](200) NULL,

CONSTRAINT [PK\_Links] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

ALTER TABLE [dbo].[Links] WITH CHECK ADD CONSTRAINT [FK\_Links\_Categories\_CategoryId] FOREIGN KEY([CategoryId])

REFERENCES [dbo].[Categories] ([Id])

GO

ALTER TABLE [dbo].[Links] CHECK CONSTRAINT [FK\_Links\_Categories\_CategoryId]

GO

ALTER TABLE [dbo].[Links] WITH CHECK ADD CONSTRAINT [FK\_Links\_Projects\_ProjectId] FOREIGN KEY([ProjectId])

REFERENCES [dbo].[Projects] ([Id])

GO

ALTER TABLE [dbo].[Links] CHECK CONSTRAINT [FK\_Links\_Projects\_ProjectId]

GO

ALTER TABLE [dbo].[Links] WITH CHECK ADD CONSTRAINT [FK\_Links\_UserProfiles\_CreateBy] FOREIGN KEY([CreateBy])

REFERENCES [dbo].[UserProfiles] ([Id])

GO

ALTER TABLE [dbo].[Links] CHECK CONSTRAINT [FK\_Links\_UserProfiles\_CreateBy]

GO

### Implementacja tabeli Notes:

EF6:

public class Note:Aggregate,IEntity

    {

        public string Title { get; set; }

        public string ShortDescription { get; set; }

        public Category Category { get; set; }

        public Guid CategoryId { get; set; }

        public Project Project { get; set; }

        public Guid ProjectId { get; set; }

        public string Content { get; set; }

    }

SQL:

CREATE TABLE [dbo].[Notes](

[Id] [uniqueidentifier] NOT NULL,

[CategoryId] [uniqueidentifier] NOT NULL,

[Content] [nvarchar](max) NOT NULL,

[CreateBy] [uniqueidentifier] NOT NULL,

[CreationDate] [datetime2](7) NOT NULL,

[ModyficationDate] [datetime2](7) NOT NULL,

[ProjectId] [uniqueidentifier] NOT NULL,

[ShortDescription] [nvarchar](250) NULL,

[Title] [nvarchar](250) NULL,

CONSTRAINT [PK\_Notes] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

ALTER TABLE [dbo].[Notes] WITH CHECK ADD CONSTRAINT [FK\_Notes\_Categories\_CategoryId] FOREIGN KEY([CategoryId])

REFERENCES [dbo].[Categories] ([Id])

GO

ALTER TABLE [dbo].[Notes] CHECK CONSTRAINT [FK\_Notes\_Categories\_CategoryId]

GO

ALTER TABLE [dbo].[Notes] WITH CHECK ADD CONSTRAINT [FK\_Notes\_Projects\_ProjectId] FOREIGN KEY([ProjectId])

REFERENCES [dbo].[Projects] ([Id])

GO

ALTER TABLE [dbo].[Notes] CHECK CONSTRAINT [FK\_Notes\_Projects\_ProjectId]

GO

ALTER TABLE [dbo].[Notes] WITH CHECK ADD CONSTRAINT [FK\_Notes\_UserProfiles\_CreateBy] FOREIGN KEY([CreateBy])

REFERENCES [dbo].[UserProfiles] ([Id])

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[Notes] CHECK CONSTRAINT [FK\_Notes\_UserProfiles\_CreateBy]

### Implementacja tabeli Mediums:

EF6:

public class Medium:Aggregate

    {

        public string ShortDescription { get; set; }

        public Category Category { get; set; }

        public Guid CategoryId { get; set; }

        public Project Project { get; set; }

        public Guid ProjectId { get; set; }

        public string Content { get; set; }

        public string Url { get; set; }

        public MediumType ContentType { get; set; }

    }

SQL:

CREATE TABLE [dbo].[Mediums](

[Id] [uniqueidentifier] NOT NULL,

[CategoryId] [uniqueidentifier] NOT NULL,

[Content] [nvarchar](max) NOT NULL,

[ContentTypeId] [uniqueidentifier] NULL,

[CreateBy] [uniqueidentifier] NOT NULL,

[CreationDate] [datetime2](7) NOT NULL,

[ModyficationDate] [datetime2](7) NOT NULL,

[ProjectId] [uniqueidentifier] NOT NULL,

[ShortDescription] [nvarchar](250) NULL,

[Url] [nvarchar](200) NULL,

CONSTRAINT [PK\_Mediums] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

ALTER TABLE [dbo].[Mediums] WITH CHECK ADD CONSTRAINT [FK\_Mediums\_Categories\_CategoryId] FOREIGN KEY([CategoryId])

REFERENCES [dbo].[Categories] ([Id])

GO

ALTER TABLE [dbo].[Mediums] CHECK CONSTRAINT [FK\_Mediums\_Categories\_CategoryId]

GO

ALTER TABLE [dbo].[Mediums] WITH CHECK ADD CONSTRAINT [FK\_Mediums\_MediumTypes\_ContentTypeId] FOREIGN KEY([ContentTypeId])

REFERENCES [dbo].[MediumTypes] ([Id])

GO

ALTER TABLE [dbo].[Mediums] CHECK CONSTRAINT [FK\_Mediums\_MediumTypes\_ContentTypeId]

GO

ALTER TABLE [dbo].[Mediums] WITH CHECK ADD CONSTRAINT [FK\_Mediums\_Projects\_ProjectId] FOREIGN KEY([ProjectId])

REFERENCES [dbo].[Projects] ([Id])

GO

ALTER TABLE [dbo].[Mediums] CHECK CONSTRAINT [FK\_Mediums\_Projects\_ProjectId]

GO

ALTER TABLE [dbo].[Mediums] WITH CHECK ADD CONSTRAINT [FK\_Mediums\_UserProfiles\_CreateBy] FOREIGN KEY([CreateBy])

REFERENCES [dbo].[UserProfiles] ([Id])

GO

ALTER TABLE [dbo].[Mediums] CHECK CONSTRAINT [FK\_Mediums\_UserProfiles\_CreateBy]

### Implementacja tabeli Projects:

EF6:

public class Project:IEntity

    {

        public Guid Id { get; set; }

        public DateTime CreateDate { get; set; }

        public UserProfile UserProfile { get; set; }

        public Guid UserProfileId { get; set; }

        public string Name { get; set; }

        public virtual ICollection<Link> Links { get; set; }

        public virtual ICollection<Medium> Mediums { get; set; }

        public virtual ICollection<Note> Notes { get; set; }

    }

SQL:

CREATE TABLE [dbo].[Projects](

[Id] [uniqueidentifier] NOT NULL,

[CreateDate] [datetime2](7) NOT NULL,

[Name] [nvarchar](200) NOT NULL,

[UserProfileId] [uniqueidentifier] NOT NULL,

CONSTRAINT [PK\_Projects] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Projects] WITH CHECK ADD CONSTRAINT [FK\_Projects\_UserProfiles\_UserProfileId] FOREIGN KEY([UserProfileId])

REFERENCES [dbo].[UserProfiles] ([Id])

GO

ALTER TABLE [dbo].[Projects] CHECK CONSTRAINT [FK\_Projects\_UserProfiles\_UserProfileId]

GO`

### Implementacja tabeli Rating

EF6:

public class Rating:Aggregate

    {

        public Score Score { get; set; }

        public Guid ScoreId { get; set; }

        public virtual ICollection<ContentRatings> ConentRatings { get; set; }

    }

SQL:

CREATE TABLE [dbo].[Ratings](

[Id] [uniqueidentifier] NOT NULL,

[CreateBy] [uniqueidentifier] NOT NULL,

[CreationDate] [datetime2](7) NOT NULL,

[ModyficationDate] [datetime2](7) NOT NULL,

[ScoreId] [uniqueidentifier] NOT NULL,

CONSTRAINT [PK\_Ratings] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Ratings] WITH CHECK ADD CONSTRAINT [FK\_Ratings\_Scores\_ScoreId] FOREIGN KEY([ScoreId])

REFERENCES [dbo].[Scores] ([Id])

GO

ALTER TABLE [dbo].[Ratings] CHECK CONSTRAINT [FK\_Ratings\_Scores\_ScoreId]

GO

ALTER TABLE [dbo].[Ratings] WITH CHECK ADD CONSTRAINT [FK\_Ratings\_UserProfiles\_CreateBy] FOREIGN KEY([CreateBy])

REFERENCES [dbo].[UserProfiles] ([Id])

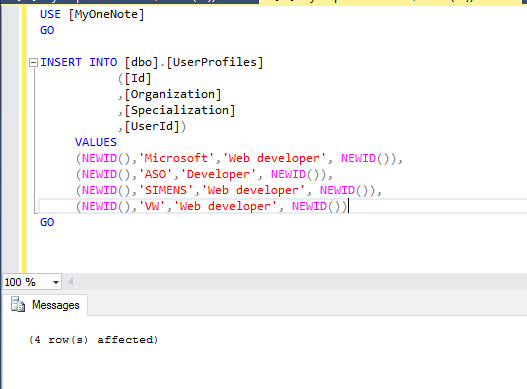
GO

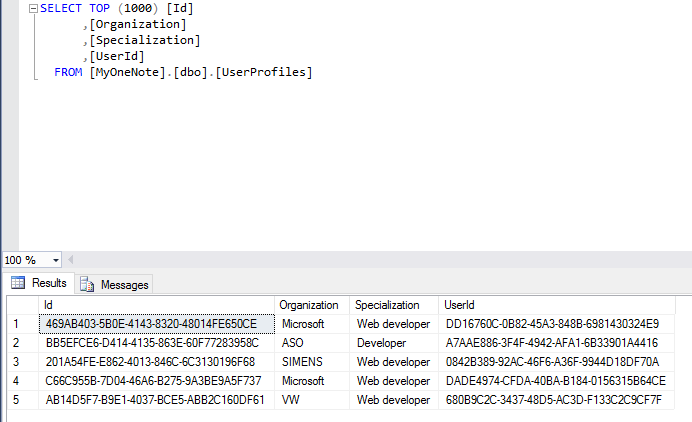
ALTER TABLE [dbo].[Ratings] CHECK CONSTRAINT [FK\_Ratings\_UserProfiles\_CreateBy]

GO

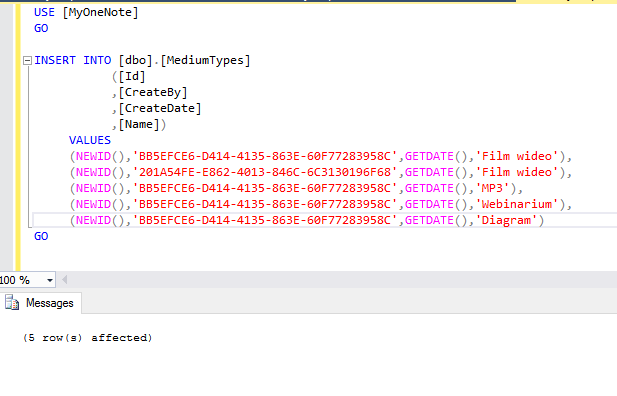
## Skrypty wprowadzające dane:

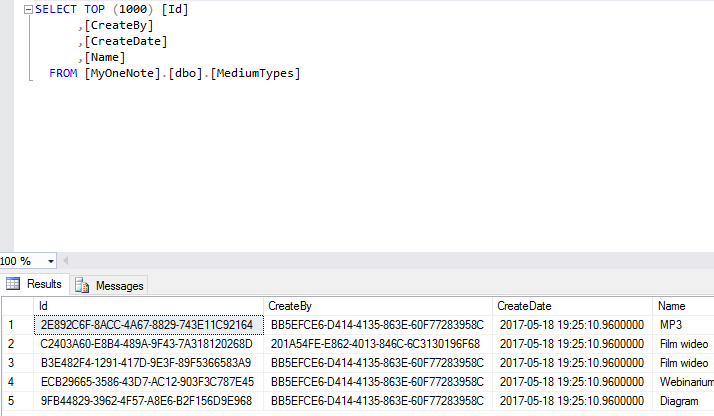
### User profiles:





### Medium types:





# Implementacja logiki biznesowej

Zaimplementowane serwisy aplikacji MyOneNote

|  |  |  |
| --- | --- | --- |
| Nazwa serwisu | Projekt | Implementacja |
| Note | + | + |
| Link | + | + |
| Medium | + | + |
| Category | + | + |
| Project | + | + |
| Rating | + | + |
| Review | + | + |
| User | + | + |
| Tag | + | + |

Zaimplementowane metody aplikacji

|  |  |  |
| --- | --- | --- |
| Nazwa metody | Projekt | Implementacja |
| AddNote | - | + |
| DeleteNote | - | + |
| UpdateNote | - | + |
| GetAllNote | - | + |
| AddMedium | - | + |
| DeleteMedium | - | + |
| UpdateMedium | - | + |
| GetAllMedium | - | + |
| AddLink | - | + |
| DeleteLink | - | + |
| UpdateLink | - | + |
| GetAllLink | - | + |
| RegisterUser | - | + |
| LoginUser | - | + |

## Implementacja warstwy mapującej SQL to C#

MyOneNote.EntityMap

#### NoteMap

public class NoteMap

{

public NoteMap(EntityTypeBuilder<Note> entityTypeBuilder)

{

entityTypeBuilder.HasKey(x => x.Id);

entityTypeBuilder.Property(x => x.Id).ValueGeneratedOnAdd();

entityTypeBuilder.Property(x => x.Content).IsRequired();

entityTypeBuilder.Property(x => x.ShortDescription).HasMaxLength(250);

entityTypeBuilder.Property(x => x.Title).HasMaxLength(250);

entityTypeBuilder.HasOne(x => x.Category).WithMany(x => x.Notes).HasForeignKey(x => x.CategoryId);

entityTypeBuilder.HasOne(x => x.Project).WithMany(x => x.Notes).HasForeignKey(x => x.ProjectId);

entityTypeBuilder.HasOne(x => x.UserProfile).WithMany(x => x.Notes).HasForeignKey(x => x.CreateBy);

}

#### CategoryMap.cs

public class CategoryMap

{

public CategoryMap(EntityTypeBuilder<Category> entityBuilder)

{

entityBuilder.HasKey(x => x.Id);

entityBuilder.Property(x => x.Id).ValueGeneratedOnAdd();

entityBuilder.Property(x => x.Name).HasMaxLength(200);

entityBuilder.HasOne(x => x.UserProfile).WithMany(x => x.Categories).HasForeignKey(x => x.CreateBy);

}

}

#### ProjectMap.cs

public ProjectMap(EntityTypeBuilder<Project> entityTypeBuilder)

{

entityTypeBuilder.HasKey(x => x.Id);

entityTypeBuilder.Property(x => x.Id).ValueGeneratedOnAdd();

entityTypeBuilder.Property(x => x.Name).HasMaxLength(200).IsRequired();

entityTypeBuilder.HasOne(x => x.UserProfile).WithMany(x => x.Projects).HasForeignKey(x => x.UserProfileId);

}

#### ContentReviewsMap.cs

public ContentReviewsMap(EntityTypeBuilder<ContentReviews> entityTypeBuilder)

{

entityTypeBuilder.HasKey(x => x.Id);

entityTypeBuilder.HasKey(x => x.Id);

entityTypeBuilder.Property(x => x.Id).ValueGeneratedOnAdd();

entityTypeBuilder.HasOne(x => x.Review).WithMany(x => x.ContentReviewses).HasForeignKey(x => x.ReviewId);

}

#### MediumMap.cs

public MediumMap(EntityTypeBuilder<Medium> entityTypeBuilder)

{

entityTypeBuilder.HasKey(x => x.Id);

entityTypeBuilder.Property(x => x.Id).ValueGeneratedOnAdd();

entityTypeBuilder.Property(x => x.Content).IsRequired();

entityTypeBuilder.Property(x => x.ShortDescription).HasMaxLength(250);

entityTypeBuilder.Property(x => x.Url).HasMaxLength(200);

entityTypeBuilder.HasOne(x => x.Category).WithMany(x => x.Mediums).HasForeignKey(x => x.CategoryId);

entityTypeBuilder.HasOne(x => x.Project).WithMany(x => x.Mediums).HasForeignKey(x => x.ProjectId);

entityTypeBuilder.HasOne(x => x.UserProfile).WithMany(x => x.Mediums).HasForeignKey(x => x.CreateBy);

}

#### LinkMap.cs

public LinkMap(EntityTypeBuilder<Link> entityTypeBuilder)

{

entityTypeBuilder.HasKey(x => x.Id);

entityTypeBuilder.Property(x => x.Id).ValueGeneratedOnAdd();

entityTypeBuilder.Property(x => x.Content).IsRequired();

entityTypeBuilder.Property(x => x.ShortDescription).HasMaxLength(250);

entityTypeBuilder.Property(x => x.Url).HasMaxLength(200);

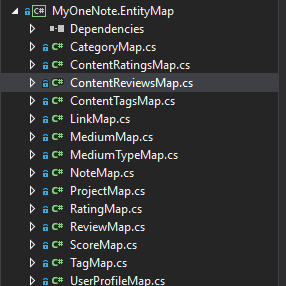
entityTypeBuilder.HasOne(x => x.Category).WithMany(x => x.Links).HasForeignKey(x => x.CategoryId);

entityTypeBuilder.HasOne(x => x.Project).WithMany(x => x.Links).HasForeignKey(x => x.ProjectId);

entityTypeBuilder.HasOne(x => x.UserProfile).WithMany(x => x.Links).HasForeignKey(x => x.CreateBy);

}

*Lista pozostałych implementacji:*



## Implementacja warstwy obsługi EntityFramework

#### MyOneNoteContext.cs

public class MyOneNoteContext : IdentityDbContext<ApplicationUser, IdentityRole,string>

{

public MyOneNoteContext(DbContextOptions<MyOneNoteContext> options) :base(options)

{

}

public DbSet<Category> Categories { get; set; }

public DbSet<Link> Links { get; set; }

public DbSet<Medium> Mediums { get; set; }

public DbSet<MediumType> MediumTypes { get; set; }

public DbSet<Note> Notes { get; set; }

public DbSet<Project> Projects { get; set; }

public DbSet<Rating> Ratings { get; set; }

public DbSet<Review> Reviews { get; set; }

public DbSet<Score> Scores { get; set; }

public DbSet<Tag> Tags { get; set; }

public DbSet<UserProfile> UserProfiles { get; set; }

public DbSet<ContentReviews> ContentReviews { get; set; }

public DbSet<ContentRatings> ContentRatings { get; set; }

public DbSet<ContentTags> ContentTags { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

base.OnModelCreating(modelBuilder);

new CategoryMap(modelBuilder.Entity<Category>());

new LinkMap(modelBuilder.Entity<Link>());

new MediumMap(modelBuilder.Entity<Medium>());

new MediumTypeMap(modelBuilder.Entity<MediumType>());

new NoteMap(modelBuilder.Entity<Note>());

new ProjectMap(modelBuilder.Entity<Project>());

new RatingMap(modelBuilder.Entity<Rating>());

new ReviewMap(modelBuilder.Entity<Review>());

new ScoreMap(modelBuilder.Entity<Score>());

new TagMap(modelBuilder.Entity<Tag>());

new UserProfileMap(modelBuilder.Entity<UserProfile>());

new ContentReviewsMap(modelBuilder.Entity<ContentReviews>());

new ContentRatingsMap(modelBuilder.Entity<ContentRatings>());

new ContentTagsMap(modelBuilder.Entity<ContentTags>());

}

}

#### Repository.cs

public class Repository<TEntity> : IRepository<TEntity> where TEntity : class

{

protected readonly DbContext \_dbContext;

private readonly DbSet<TEntity> \_dbSet;

public Repository(DbContext dbContext)

{

\_dbContext = dbContext ?? throw new ArgumentNullException(nameof(dbContext));

\_dbSet = \_dbContext.Set<TEntity>();

}

public void ChangeTable(string table)

{

if (\_dbContext.Model.FindEntityType(typeof(TEntity)).Relational() is RelationalEntityTypeAnnotations relational)

{

relational.TableName = table;

}

}

public IQueryable<TEntity> Query(Expression<Func<TEntity, bool>> predicate, bool disableTracking = true)

{

if (disableTracking)

{

return \_dbSet.AsNoTracking().Where(predicate);

}

else

{

return \_dbSet.Where(predicate);

}

}

public IQueryable<TEntity> FromSql(string sql, params object[] parameters) => \_dbSet.FromSql(sql, parameters);

public TEntity Find(params object[] keyValues)

{

return \_dbSet.Find(keyValues);

}

public Task<TEntity> FindAsync(params object[] keyValues) => \_dbSet.FindAsync(keyValues);

public Task<TEntity> FindAsync(object[] keyValues, CancellationToken cancellationToken) => \_dbSet.FindAsync(keyValues, cancellationToken);

public TEntity Insert(TEntity entity)

{

return \_dbSet.Add(entity) as TEntity;

}

public void Insert(params TEntity[] entities) => \_dbSet.AddRange(entities);

public void Insert(IEnumerable<TEntity> entities) => \_dbSet.AddRange(entities);

public Task InsertAsync(TEntity entity, CancellationToken cancellationToken = default(CancellationToken))

{

return \_dbSet.AddAsync(entity, cancellationToken);

}

public Task InsertAsync(params TEntity[] entities) => \_dbSet.AddRangeAsync(entities);

public Task InsertAsync(IEnumerable<TEntity> entities, CancellationToken cancellationToken = default(CancellationToken)) => \_dbSet.AddRangeAsync(entities, cancellationToken);

public void Update(TEntity entity)

{

\_dbSet.Update(entity);

}

public void Update(params TEntity[] entities) => \_dbSet.UpdateRange(entities);

public void Update(IEnumerable<TEntity> entities) => \_dbSet.UpdateRange(entities);

public void Delete(TEntity entity) => \_dbSet.Remove(entity);

public void Delete(object id)

{

var typeInfo = typeof(TEntity).GetTypeInfo();

var key = \_dbContext.Model.FindEntityType(typeInfo.Name).FindPrimaryKey().Properties.FirstOrDefault();

var property = typeInfo.GetProperty(key?.Name);

if (property != null)

{

var entity = Activator.CreateInstance<TEntity>();

property.SetValue(entity, id);

\_dbContext.Entry(entity).State = EntityState.Deleted;

}

else

{

var entity = \_dbSet.Find(id);

if (entity != null)

{

Delete(entity);

}

}

}

public void Delete(params TEntity[] entities) => \_dbSet.RemoveRange(entities);

public void Delete(IEnumerable<TEntity> entities) => \_dbSet.RemoveRange(entities);

public IEnumerable<TEntity> GetAll()

{

return \_dbSet;

}

public DbContext GetContext()

{

return \_dbContext;

}

}

#### UnitOfWork.cs

public class UnitOfWork<TContext> : IRepositoryFactory, IUnitOfWork<TContext> where TContext: MyOneNoteContext

{

private readonly TContext \_context;

private bool disposed = false;

private Dictionary<Type, object> repositories;

public UnitOfWork(TContext context)

{

\_context = context ?? throw new ArgumentNullException(nameof(context));

}

public TContext DbContext => \_context;

public void ChangeDatabase(string database)

{

if (\_context.Model.Relational() is RelationalModelAnnotations relational)

{

relational.DatabaseName = database;

}

var connection = \_context.Database.GetDbConnection();

if (connection.State.HasFlag(ConnectionState.Open))

{

connection.ChangeDatabase(database);

}

var items = \_context.Model.GetEntityTypes();

foreach (var item in items)

{

if (item.Relational() is RelationalEntityTypeAnnotations extensions)

{

extensions.Schema = database;

}

}

}

public IRepository<TEntity> GetRepository<TEntity>() where TEntity : class

{

if (repositories == null)

{

repositories = new Dictionary<Type, object>();

}

var type = typeof(TEntity);

if (!repositories.ContainsKey(type))

{

repositories[type] = new Repository<TEntity>(\_context);

}

return (IRepository<TEntity>)repositories[type];

}

public int ExecuteSqlCommand(string sql, params object[] parameters) => \_context.Database.ExecuteSqlCommand(sql, parameters);

public IQueryable<TEntity> FromSql<TEntity>(string sql, params object[] parameters) where TEntity : class => \_context.Set<TEntity>().FromSql(sql, parameters);

public int SaveChanges(bool ensureAutoHistory = false)

{

//**TODO**

return \_context.SaveChanges();

}

public async Task<int> SaveChangesAsync(bool ensureAutoHistory = false)

{

//**TODO**

return await \_context.SaveChangesAsync();

}

public async Task<int> SaveChangesAsync(bool ensureAutoHistory = false, params IUnitOfWork[] unitOfWorks)

{

using (var transaction = \_context.Database.BeginTransaction())

{

try

{

var count = 0;

foreach (var unitOfWork in unitOfWorks)

{

var uow = unitOfWork as UnitOfWork<MyOneNoteContext>;

uow.DbContext.Database.UseTransaction(transaction.GetDbTransaction());

count += await uow.SaveChangesAsync(ensureAutoHistory);

}

count += await SaveChangesAsync(ensureAutoHistory);

transaction.Commit();

return count;

}

catch (Exception ex)

{

transaction.Rollback();

throw ex;

}

}

}

public void Dispose()

{

Dispose(true);

GC.SuppressFinalize(this);

}

protected virtual void Dispose(bool disposing)

{

if (!disposed)

{

if (disposing)

{

if (repositories != null)

{

repositories.Clear();

}

\_context.Dispose();

}

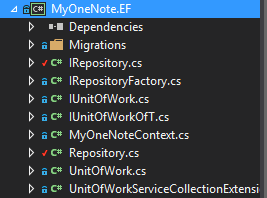
}

disposed = true;

}

}

Pozostałę implementacje:



## Implementacja warstwy serwisowej

#### BaseService.cs

public abstract class BaseService<TEntity> :IBaseService<TEntity> where TEntity: class , IEntity

{

private readonly IUnitOfWork \_unitOfWork;

public readonly IRepository<TEntity> \_baseRepository;

public BaseService(IUnitOfWork unitOfWork)

{

\_unitOfWork = unitOfWork;

\_baseRepository = unitOfWork.GetRepository<TEntity>();

}

public void PerformCommand(Action action)

{

action();

\_unitOfWork.SaveChanges();

}

public TEntity PerformCommand(Func<TEntity> action)

{

TEntity entity = action();

\_unitOfWork.SaveChanges();

return entity;

}

public TEntity Add(TEntity entity)

{

PerformCommand(() =>

{

\_baseRepository.Insert(entity);

});

return entity;

}

public void Delete(TEntity entity)

{

PerformCommand(() =>

{

\_baseRepository.Delete(entity);

});

}

public TEntity Update(TEntity entity)

{

PerformCommand(() =>

{

\_baseRepository.Update(entity);

});

return entity;

}

public virtual IEnumerable<TEntity> GetAll()

{

return \_baseRepository.GetAll();

}

#### NoteService.cs

public class NoteService:BaseService<Note>, INoteService

{

public NoteService(IUnitOfWork unitOfWork):base(unitOfWork)

{

}

public override IEnumerable<Note> GetAll()

{

var context = \_baseRepository.GetContext();

var notes = context.Set<Note>().Include(note =>note.Category)

.Include(note=>note.Project);

return notes.ToList();

}

}

#### UserService.cs

public class UserService:BaseService<UserProfile>, IUserService

{

private readonly UserManager<ApplicationUser> \_userManager;

private readonly SignInManager<ApplicationUser> \_loginManager;

private readonly RoleManager<IdentityRole> \_roleManager;

public UserService(

IUnitOfWork unitOfWork,

UserManager<ApplicationUser> userManager,

SignInManager<ApplicationUser> loginMenager,

RoleManager<IdentityRole> roleManager):base(unitOfWork)

{

\_userManager = userManager;

\_loginManager = loginMenager;

\_roleManager = roleManager;

}

public async Task<ApplicationUser> Register(UserRegisterVM model)

{

ApplicationUser user = new ApplicationUser();

user.UserName = model.UserName;

user.Email = model.Email;

user.FullName = model.FullName;

user.BirthDate = model.BirthDate;

IdentityResult result = await \_userManager.CreateAsync

(user, model.Password);

if (!result.Succeeded)

throw new RegisterUserException(result.Errors.ToString());

if (!\_roleManager.RoleExistsAsync("NormalUser").Result)

{

IdentityRole role = new IdentityRole();

role.Name = "NormalUser";

IdentityResult roleResult = \_roleManager.

CreateAsync(role).Result;

if (!roleResult.Succeeded)

{

throw new RegisterUserException(roleResult.Errors.ToString());

}

}

\_userManager.AddToRoleAsync(user,

"NormalUser").Wait();

PerformCommand(() =>

{

\_baseRepository.Insert(new UserProfile()

{

UserId = user.Id,

});

});

return user;

}

public UserProfile GetUserProfile(ApplicationUser user)

{

string userId = user.Id.ToString();

return \_baseRepository.Find(new UserProfile(userId));

}

public async Task<bool> Login(LoginVM login)

{

var result = await \_loginManager.PasswordSignInAsync(login.UserName, login.Password, login.RememberMe, lockoutOnFailure: false);

if (result.Succeeded)

return true;

return false;

}

public UserProfile GetserProfileByClaims(string userId)

{

Expression<Func<UserProfile, bool>> filter()

{

return x => x.UserId.Equals(userId);

}

var temp = \_baseRepository.Query(filter()).FirstOrDefault();

return temp;

}

}

#### ProjectService.cs

public class ProjectService:BaseService<Project>,IProjectService

{

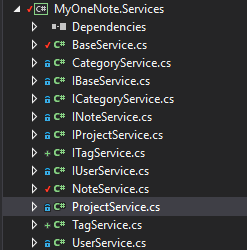
public ProjectService(IUnitOfWork unitOfWork) : base(unitOfWork)

{

}

}

#### Pozostałe implementacje:



## Implementacja warstwy ViewModel

#### ListNoteVM.cs

public class ListNoteVM

{

public string Title { get; set; }

public Guid Id { get; set; }

public string Description { get; set; }

public string Category { get; set; }

public Guid CategoryId { get; set; }

public decimal Rating

{

get

{

Random random = new Random();

return random.Next(0, 100);

}

}

}

#### LoginVM.cs

public class LoginVM

{

[Required]

[DataType(DataType.Password)]

public string Password { get; set; }

[Required]

public bool RememberMe { get; set; }

[Required]

public string UserName { get; set; }

}

#### AddNoteVM.cs

public class AddNoteVM

{

public string Title { get; set; }

public Guid ProjectId { get; set; }

public List<SelectListItem> Projects { get; set; }

public Guid CategoryId { get; set; }

public List<SelectListItem> Categories { get; set; }

public string Content { get; set; }

public string Description { get; set; }

public Guid[] TagsIds { get; set; }

public List<SelectListItem> Tags { get; set; }

}

#### CategoryVM.cs

public class CategoryVM

{

public Guid Id { get; set; }

public DateTime CreateDate { get; set; }

public string UserProfileName { get; set; }

public Guid CreateBy { get; set; }

public string Name { get; set; }

public AddCategoryVM AddCategoryVm { get; set; }

}

#### ProjectVM.cs

public class ProjectVM

{

public Guid Id { get; set; }

[Display(Name = "Data utworzenia")]

public DateTime CreateDate { get; set; }

[Display(Name = "Nazwa")]

public string Name { get; set; }

}

#### ProjectsVM.cs

public class ProjectsVM

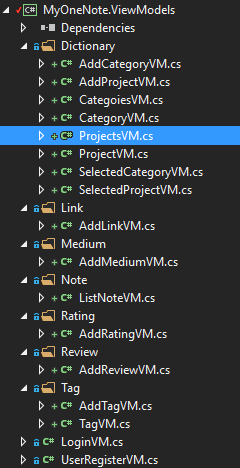
{

public IEnumerable<ProjectVM> ProjectMm { get; set; }

public AddProjectVM AddProjectVM { get; set; }

}

#### Pozostała implementacja:



## Implementacja logiki WWW

#### NoteController.cs

public class NoteController:UserContextController

{

private readonly INoteService \_noteService;

private readonly ICategoryService \_categoryService;

private readonly IProjectService \_projectService;

private readonly ITagService \_tagService;

public NoteController(UserManager<ApplicationUser> userManager, IUserService userService

, INoteService noteService

, IProjectService projectService

, ICategoryService categoryService

, ITagService tagService) : base(userManager, userService)

{

\_noteService = noteService ?? throw new ArgumentException(nameof(noteService));

\_categoryService = categoryService ?? throw new ArgumentException(nameof(categoryService));

\_projectService = projectService ?? throw new ArgumentException(nameof(projectService));

\_tagService = tagService ?? throw new ArgumentException(nameof(tagService));

}

public IActionResult AddNote()

{

AddNoteVM vm = new AddNoteVM();

vm.Categories = Mapper.Map<IEnumerable<Category>, List<SelectListItem>>(\_categoryService.GetAll());

vm.Projects = Mapper.Map<IEnumerable<Project>, List<SelectListItem>>(\_projectService.GetAll());

vm.Tags = Mapper.Map<IEnumerable<Tag>, List<SelectListItem>>(\_tagService.GetAll());

return View(vm);

}

public IActionResult Note()

{

IEnumerable<ListNoteVM> list = null;

try

{

var noteList = \_noteService.GetAll();

list = Mapper.Map<IEnumerable<Note>, IEnumerable<ListNoteVM>>(noteList);

}

catch (Exception e)

{

throw e;

}

return View(list);

}

public IActionResult Details(string id)

{

return View();

}

public RedirectToActionResult InsertNote(AddNoteVM note)

{

if (ModelState.IsValid)

{

try

{

Note newNote = Mapper.Map<AddNoteVM, Note>(note);

newNote.CreateBy = GetProfile(this.User).Id;

newNote.CreationDate =DateTime.Now;

\_noteService.Add(newNote);

}

catch (Exception e)

{

Console.WriteLine(e);

throw;

}

}

return RedirectToAction("Note");

}

public IActionResult Menage()

{

IEnumerable<ListNoteVM> list = null;

try

{

var noteList = \_noteService.GetAll();

list = Mapper.Map<IEnumerable<Note>, IEnumerable<ListNoteVM>>(noteList);

}

catch (Exception e)

{

throw e;

}

return View(list);

}

public RedirectToActionResult Delete(string id)

{

try

{

\_noteService.Delete(new Note() { Id = Guid.Parse(id) });

}

catch (Exception ex)

{

throw ex;

}

return RedirectToAction("Menage");

}

}

#### DictionaryController.cs

public class DictionaryController : UserContextController

{

private readonly ICategoryService \_categoryService;

private readonly IProjectService \_projectService;

public DictionaryController(

UserManager<ApplicationUser> userManager,

IUserService userService,

IProjectService projectService,

ICategoryService categoryService):base(userManager,userService)

{

\_categoryService = categoryService;

\_projectService = projectService;

}

[HttpPost]

public RedirectToActionResult AddCategory(AddCategoryVM categoryVm)

{

if (ModelState.IsValid)

{

Category category = null;

try

{

category = Mapper.Map<Category>(categoryVm);

category.CreateBy = GetProfile(this.User).Id;

category.CreateDate = DateTime.Now;

\_categoryService.Add(category);

}

catch (Exception ex)

{

throw ex;

}

}

return RedirectToAction("Category");

}

public RedirectToActionResult DeleteCategory(string id)

{

try

{

\_categoryService.Delete(new Category(){Id = Guid.Parse(id)});

}

catch (Exception ex)

{

throw ex;

}

return RedirectToAction("Category");

}

public RedirectToActionResult DeleteProject(string id)

{

try

{

\_projectService.Delete(new Project() { Id = Guid.Parse(id) });

}

catch (Exception ex)

{

throw ex;

}

return RedirectToAction("Project");

}

[HttpGet]

public IActionResult Category()

{

CategoiesVM vm = new CategoiesVM();

vm.CategoryVm = Mapper.Map<IEnumerable<Category>, IEnumerable<CategoryVM>>(\_categoryService.GetAll());

return View(vm);

}

[HttpPost]

public RedirectToActionResult AddProject(AddProjectVM model)

{

Project project = null;

if (ModelState.IsValid)

{

try

{

project = Mapper.Map<AddProjectVM, Project>(model);

project.UserProfileId = GetProfile(this.User).Id;

project.CreateDate = DateTime.Now;

\_projectService.Add(project);

}

catch (Exception ex)

{

throw ex;

}

}

return RedirectToAction("Project");

}

[HttpGet]

public IActionResult Project()

{

ProjectsVM vm = new ProjectsVM();

vm.ProjectMm = Mapper.Map<IEnumerable<Project>, IEnumerable<ProjectVM>>(\_projectService.GetAll());

return View(vm);

}

}

#### Implementacja kontenera IoC

public class ApiModule: Autofac.Module

{

protected override void Load(ContainerBuilder builder)

{

var dataAccess = Assembly.GetEntryAssembly();

//builder.RegisterGeneric(typeof(UnitOfWork<>))

// .As<IUnitOfWork>()

// .InstancePerLifetimeScope();

builder.RegisterType<AuthMessageSender>()

.As<IEmailSender>()

.InstancePerLifetimeScope();

builder.RegisterType<AuthMessageSender>()

.As<ISmsSender>()

.InstancePerLifetimeScope();

builder.RegisterType<NoteService>()

.As<INoteService>()

.InstancePerLifetimeScope();

builder.RegisterType<CategoryService>()

.As<ICategoryService>()

.InstancePerLifetimeScope();

builder.RegisterType<ProjectService>()

.As<IProjectService>()

.InstancePerLifetimeScope();

builder.RegisterType<UserService>()

.As<IUserService>()

.InstancePerLifetimeScope();

builder.RegisterType<TagService>()

.As<ITagService>()

.InstancePerLifetimeScope();

//builder.RegisterAssemblyTypes(dataAccess)

// .Where(t => t.Name.EndsWith("Service"))

// .AsImplementedInterfaces();

}

}

#### Implementacja Automapper

public class MapperProfile:Profile

{

public MapperProfile()

{

CreateMap<Category, CategoryVM>()

.ForMember(x=>x.UserProfileName,opt=>opt.MapFrom(dest=>dest.UserProfile.Id))

.ForMember(x=>x.Name,opt=>opt.MapFrom(dest=>dest.Name))

.ReverseMap();

CreateMap<Category, AddCategoryVM>()

.ForMember(x=>x.Name,opt=>opt.MapFrom(dest=>dest.Name))

.ReverseMap();

CreateMap<Project, ProjectVM>().ReverseMap();

CreateMap<Project, AddProjectVM>().ReverseMap();

CreateMap<Category, SelectedCategoryVM>()

.ForMember(x => x.Name, opt => opt.MapFrom(dest => dest.Name))

.ForMember(x => x.Id, opt => opt.MapFrom(dest => dest.Id))

.ReverseMap();

CreateMap<Project, SelectedProjectVM>()

.ForMember(x => x.Name, opt => opt.MapFrom(dest => dest.Name))

.ForMember(x => x.Id, opt => opt.MapFrom(dest => dest.Id))

.ReverseMap();

CreateMap<Category, SelectListItem>()

.ForMember(x => x.Text, opt => opt.MapFrom(dest => dest.Name))

.ForMember(x => x.Value, opt => opt.MapFrom(dest => dest.Id))

.ReverseMap();

CreateMap<Project, SelectListItem>()

.ForMember(x => x.Text, opt => opt.MapFrom(dest => dest.Name))

.ForMember(x => x.Value, opt => opt.MapFrom(dest => dest.Id))

.ReverseMap();

CreateMap<Tag, SelectListItem>()

.ForMember(x => x.Text, opt => opt.MapFrom(dest => dest.Name))

.ForMember(x => x.Value, opt => opt.MapFrom(dest => dest.Id))

.ReverseMap();

CreateMap<Note, ListNoteVM>()

.ForMember(x => x.Id, opt => opt.MapFrom(dest => dest.Id))

.ForMember(x => x.Description, opt => opt.MapFrom(dest => dest.ShortDescription))

.ForMember(x=>x.Category,opt=>opt.MapFrom(dest=>dest.Category.Name))

.ForMember(x=>x.CategoryId,opt=>opt.MapFrom(dest=>dest.Category.Id))

.ReverseMap();

CreateMap<Note, AddNoteVM>()

.ForMember(x => x.CategoryId, opt => opt.MapFrom(dest => dest.CategoryId))

.ForMember(x => x.Content, opt => opt.MapFrom(dest => dest.Content))

.ForMember(x => x.Description, opt => opt.MapFrom(dest => dest.ShortDescription))

.ForMember(x => x.ProjectId, opt => opt.MapFrom(dest => dest.ProjectId))

.ForMember(x => x.Title, opt => opt.MapFrom(dest => dest.Title));

CreateMap<AddNoteVM, Note>()

.ForMember(x => x.CategoryId, opt => opt.MapFrom(dest => dest.CategoryId))

.ForMember(x => x.Content, opt => opt.MapFrom(dest => dest.Content))

.ForMember(x => x.ShortDescription, opt => opt.MapFrom(dest => dest.Description))

.ForMember(x => x.ProjectId, opt => opt.MapFrom(dest => dest.ProjectId))

.ForMember(x => x.Title, opt => opt.MapFrom(dest => dest.Title));

}

}

## Interfejsy aplikacji

#### IBaseService.cs

public interface IBaseService<TEntity> where TEntity:class ,IEntity

{

IEnumerable<TEntity> GetAll();

TEntity Update(TEntity entity);

void Delete(TEntity entity);

TEntity Add(TEntity entity);

TEntity PerformCommand(Func<TEntity> action);

void PerformCommand(Action action);

}

#### IUserService.cs

public interface IUserService

{

Task<ApplicationUser> Register(UserRegisterVM model);

UserProfile GetUserProfile(ApplicationUser user);

Task<bool> Login(LoginVM login);

UserProfile GetserProfileByClaims(string userId);

}

#### IRepository.cs

public interface IRepository<TEntity> where TEntity : class

{

void ChangeTable(string table);

IQueryable<TEntity> Query(Expression<Func<TEntity, bool>> predicate, bool disableTracking = true);

IQueryable<TEntity> FromSql(string sql, params object[] parameters);

TEntity Find(params object[] keyValues);

Task<TEntity> FindAsync(params object[] keyValues);

Task<TEntity> FindAsync(object[] keyValues, CancellationToken cancellationToken);

TEntity Insert(TEntity entity);

void Insert(params TEntity[] entities);

void Insert(IEnumerable<TEntity> entities);

Task InsertAsync(TEntity entity, CancellationToken cancellationToken = default(CancellationToken));

Task InsertAsync(params TEntity[] entities);

Task InsertAsync(IEnumerable<TEntity> entities, CancellationToken cancellationToken = default(CancellationToken));

void Update(TEntity entity);

void Update(params TEntity[] entities);

void Update(IEnumerable<TEntity> entities);

void Delete(object id);

void Delete(TEntity entity);

void Delete(params TEntity[] entities);

void Delete(IEnumerable<TEntity> entities);

IEnumerable<TEntity> GetAll();

DbContext GetContext();

}

#### IUnitOfWork.cs

public interface IUnitOfWork : IDisposable

{

void ChangeDatabase(string database);

IRepository<TEntity> GetRepository<TEntity>() where TEntity : class;

int SaveChanges(bool ensureAutoHistory = false);

Task<int> SaveChangesAsync(bool ensureAutoHistory = false);

int ExecuteSqlCommand(string sql, params object[] parameters);

IQueryable<TEntity> FromSql<TEntity>(string sql, params object[] parameters) where TEntity : class;

}

#### IEmailSender.cs

public interface IEmailSender

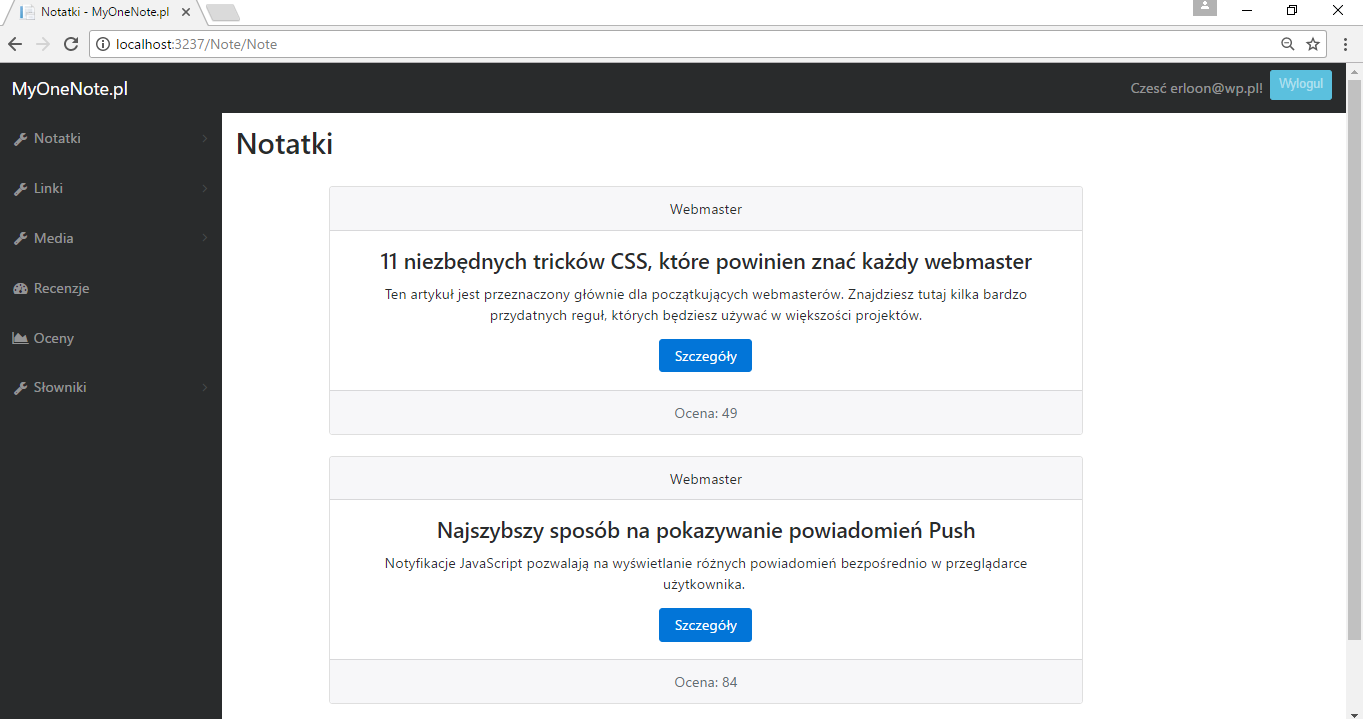
{

Task SendEmailAsync(string email, string subject, string message);

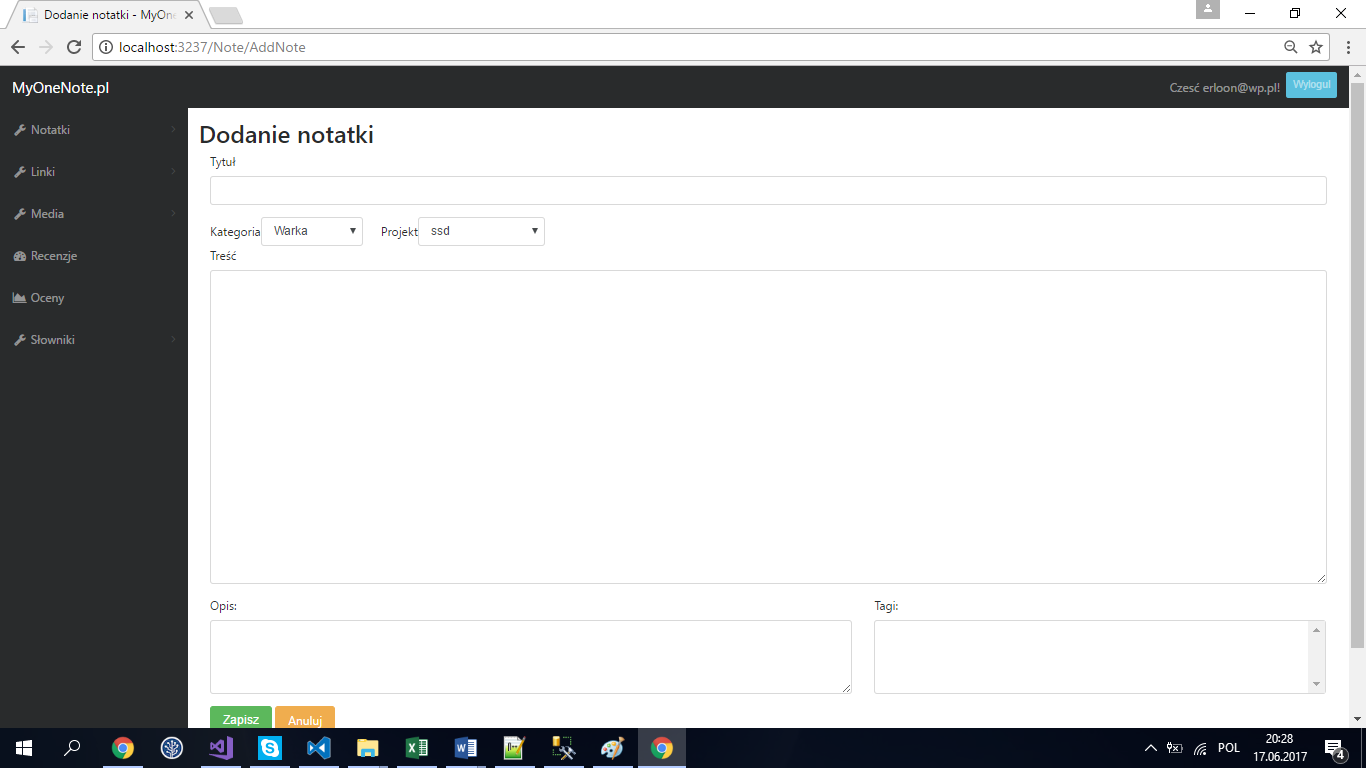
}

## Implementacja grafczna aplikacji:

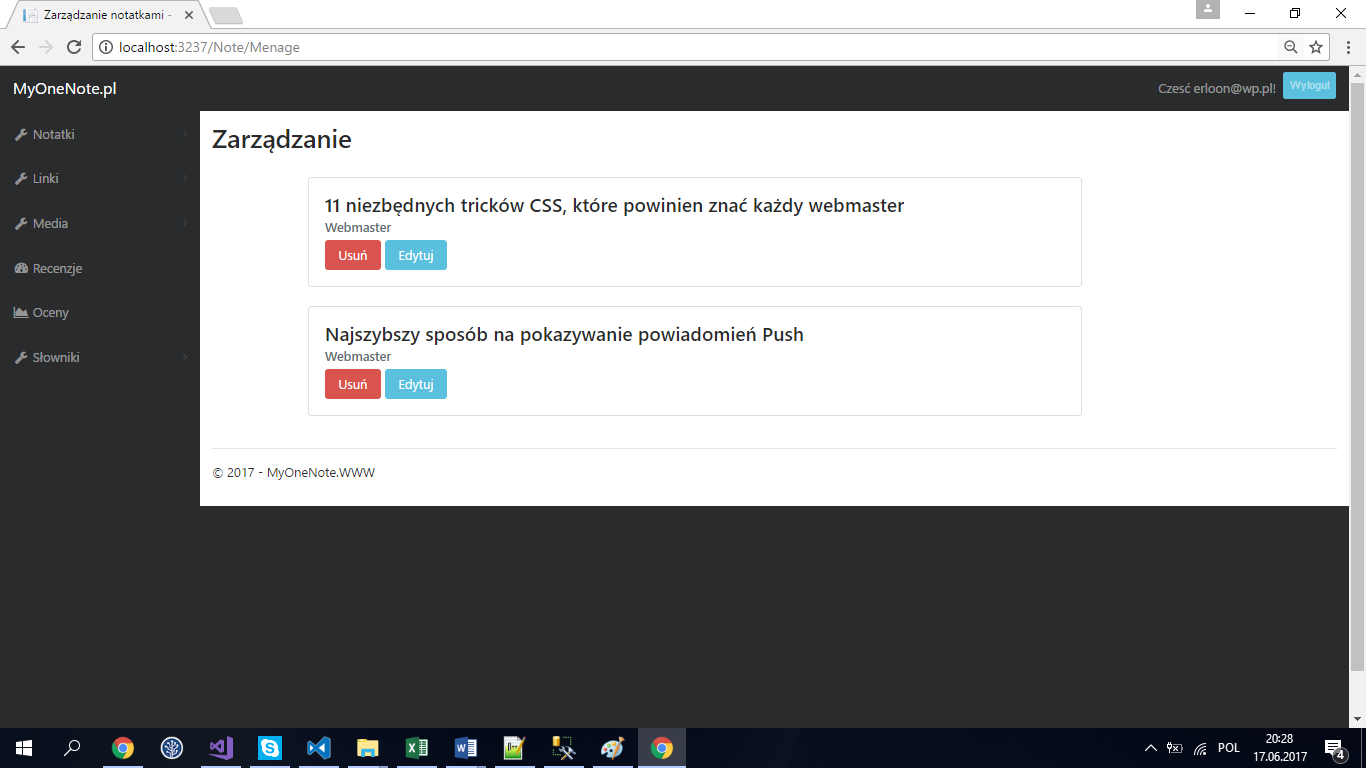
#### Okno głowne modułu Note:



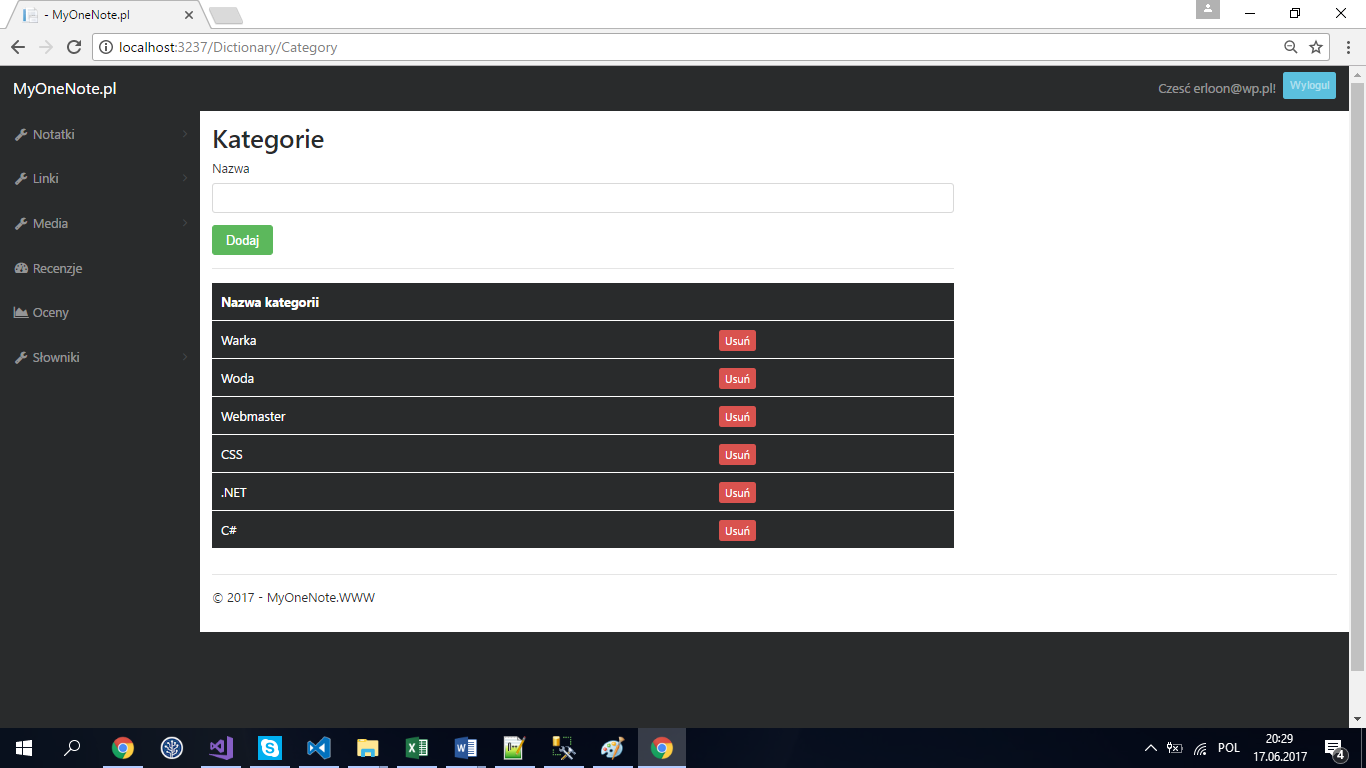
#### Implementacja okna Dodaj notatkę



#### Implementacja okna Zarządzania notatkami:



Implementacja okna Kategorie



Implementacja okna Projekty

