Portal społecznościowy PastExplorer

Opis systemu i elementy specyfikacji

## Opis systemu

Aplikacja internetowa utworzona w ramach tego projektu pozwalać będzie za pomocą zdjęć umieszczanych przez użytkowników obserwować jak różnego rodzaju “rzeczy” (np. ludzie, budynki, krajobrazy) zmieniały się wraz z upływem czasu. Każdy użytkownik będzie mógł zakładać albumy będące kolekcjami zdjęć zgodnych tematycznie robionych w różnych odstępach czasu, np. codziennie, cotygodniowo, bądź corocznie.

Aplikacja będzie mocno nakierowana “społecznościowo”, tj. zawierać elementy pozwalające na tworzenie wokół serwisu społeczności użytkowników aktywnie uczestniczących w jego rozwoju, np. ocenianie i komentowanie albumów.

Przykłady zastosowań:

* codziennie robione zdjęcia rozwoju naszego dziecka
* cotygodniowo robione zdjęcia stanu budowy pewnego obiektu, np. stadionu piłkarskiego
* corocznie robione zdjęcia rynku w naszym mieście



## Charakterystyka użytkowników

Z portalu korzystać będą dwie grupy użytkowników:

* Zwykli użytkownicy –posiadać będą możliwość oglądania istniejących albumów zdjęć oraz ich ocenianie i komentowanie, a także tworzenie własnych; ponadto użytkownicy będą mogli śledzić dowolny album, tj. będą automatycznie powiadamiani o pojawianiu się nowych zdjęć
* Administratorzy – posiadać będą te same możliwości co zwykli użytkownicy, jednak dodatkowo będą odpowiedzialni za moderacją i nadzór nad działaniem serwisu; oprócz typowych zadań administracyjnych (zarządzanie kontami użytkowników, modyfikacja komentarzy, itp.) użytkownicy tego typu będą również mogli definiować kategorie, w ramach których tworzone będą mogły być albumy

## Wymagania funkcjonalne

Poniższa lista prezentuje funkcjonalności jakie posiadać będzie portal PastExplorer z podziałem na kategorie.

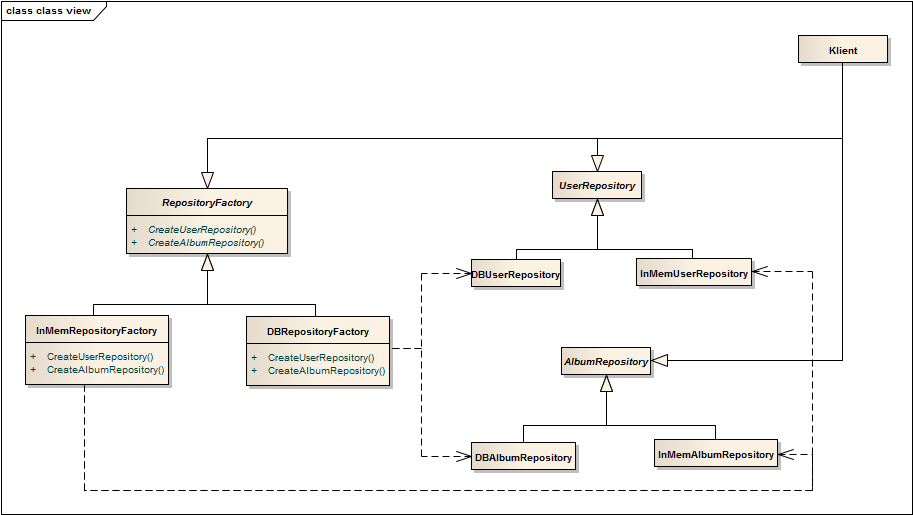
1. Profile użytkowników
   * rejestracja konta, aktywacja przez e-mail
   * logowanie/wylogowywanie
   * widok szczegółowy profilu, przeglądanie profilów innych użytkowników
   * ustawienia konta/profilu
     + dane dodatkowe (wiek, opis profilu)
     + ustawienia powiadomień
     + zmiana oraz odzyskiwanie hasła
     + usunięcie konta (wraz ze wszystkimi albumami, zdjęciami, komentarzami)
2. Albumy zdjęć
   * tworzenie nowego albumu
     + dane podstawowe (nazwa, opis, kategoria)
     + ustawienia powiadomień e-mail (odstęp czasowy pomiędzy kolejnymi powiadomieniami)
     + ustawienia poziomu prywatności
       - publiczny - dostępny dla wszystkich (również dla niezarejestrowanych)
       - prywatny - dostępny dla właściciela i wybranych osób:
         * o podanych nazwach użytkowników
         * znających hasło dostępu do albumu
     + ustawienia praw do komentowania albumu
       - automatycznie zezwalaj na komentarze
       - każdy komentarz wymaga autoryzacji
       - całkowite zablokowanie komentarzy
   * przeglądanie istniejących albumów
     + wyświetlanie informacji o albumie (nazwa, opis, ocena, komentarze, itp.)
     + przeglądarka zdjęć w albumie (atrakcyjna wizualizacja w formie „kliszy fotograficznej”)
   * wyszukiwanie albumów
   * komentowanie albumów
   * zarządzanie albumami
     + dodawanie zdjęć
     + usuwanie zdjęć
     + usunięcie całego albumu
     + zmiana ustawień albumu
3. Przeglądarka albumów
   * pasek czasu - zdjęcia wyświetlane w formie „kliszy”, przesuwane za pomocą suwaka, efektu zbliżony do filmu (obserwacje zmian obiektu wraz z upływem czasu)
   * przyciski szybkiej nawigacji (przewijanie w czasie dzień/tydzień/miesiąc/rok w przód lub w tył)
4. Elementy społecznościowe
   * ocenianie albumów (oceny dwustopniowe: podoba mi się / nie podoba mi się)
   * rankingi albumów (np. najlepiej ocenianie, najpopularniejsze, itp.)
   * śledzenie albumów (otrzymywanie poprzez e-mail powiadomień o nowych zdjęciach pojawiających się w śledzonym albumie)
5. Administracja
   * edycja/usuwanie komentarzy, zdjęć oraz kont użytkowników (profili)
   * edycja listy dostępnych kategorii albumów

## Wymagania niefunkcjonalne

1. Zachowanie spójności pomiędzy albumami w bazie danych i zdjęciami należącymi do tych albumów, przechowywanymi oddzielnie w systemie plików
2. Łatwość użytkownia – portal powinien być intuicyjny w obsłudze i w działaniu przypominać powinien istniejące już, popularne portale społecznościowe, co ułatwi korzystanie z niego nowym użytkownikom

## Wzorce projektowe

### Fabryka abstrakcyjna (Abstract Factory)



**Cel wzorca:** izolacja kodu aplikacji od rodziny klas odpowiedzialnych za zapewnienie dostępu do bazy danych, wymuszenie użycia klas dostępowych należących do jednej rodziny w celu zapewnienia spójności w użyciu magazynów danych.

Klasy abstrakcyjne \*Repository definiują interfejs dostępu do tabel zawierających rekordy odpowiedniego modelu danych, np. użytkownika (UserRepository) czy albumu (AlbumRepository). Klasa abstrakcyjna RepositoryFactory definiuje interfejs fabryk pozwalających na tworzenie obiektów \*Repository realizujących dostęp do danych w specyficzny, konkretny sposób. W projekcie zastosowane zostaną dwa sposoby dostępu do danych: komunikacja z motorem bazodanowym (DB\*Repository) oraz przechowywanie danych w pamięci ulotnej (InMem\*Repository). Drugi mechanizm dostępu uzasadniony jest koniecznością przeprowadzenia testów logiki biznesowej w odizolowaniu od warstwy bazodanowej, która może zakłócić te testy, przez co powinna zostać przetestowana osobno. Ponadto zastosowanie tego wzorca pozwoli na wygodniejszą pracę na zmieniającej się strukturze modeli obiektów bez konieczności synchronizacji schematu i zawartości bazy danych.

**Implementacja:**

*Definicja rodziny klas dostępu do bazy danych:*

public abstract class UserRepository {

public User GetById(int id);

// … inne operacje na bazie uzytkownikow …

}

public class DBUserRepository : UserRepository {

// … implementacja operacji na uzytkownikach w bazie danych …

}

public class InMemUserRepository : UserRepository {

// … implementacja operacji na uzytkownikach w pamieci operacyjnej …

}

*Definicja fabryki abstrakcyjnej i jej konkretnych implementacji (baza danych lub pamięć operacyjna):*

public interface RepositoryFactory {

UserRepository CreateUserRepository();

// …

}

public class DBRepositoryFactory : RepositoryFactory {

public UserRepository CreateUserRepository {

return new DBUserRepository();

}

// … metody fabryczne dla innych klas z rodziny DB\*Repository …

}

public class InMemRepositoryFactory : RepositoryFactory {

public UserRepository CreateUserRepository {

return new InMemUserRepository();

}

// … metody fabryczne dla innych klas z rodziny InMem\*Repository …

}

*Użycie fabryki do tworzenia obiektów dostępowych bazy danych:*

RepositoryFactory factory = RepositoryFactory.GetInstance();

UserRepository users = factory.CreateUserRepository();

// … użycie obiektu ‘users’ do operowania na bazie użytkowników

### Singleton

* RepositoryFactory (a raczej konkretna implementacja) z poprzedniego punktu
* SessionProvider dla DB\*Repository