Technologie Obiektowe – projekt

System zarządzania rezerwacjami sal

# Opis projektu

Projekt ma na celu stworzenie systemu zarządzania rezerwacjami sal, którego głównymi możliwościami są:

* Zarządzanie użytkownikami
* Zarządzanie salami
* Zarządzanie i walidacja rezerwacji

## Główne funkcjonalności

### Moduł użytkowników

* Dodanie użytkownika
* Pobranie użytkownika
* Pobranie listy użytkowników
* Usunięcie użytkownika

### Moduł sal

* Dodanie sali
* Pobranie sali
* Pobranie listy sal sortując je według zadanego kryterium
* Usunięcie sali

### Moduł rezerwacji

* Dodanie rezerwacji z wyróżnieniem organizatora i zaproszonych oraz walidacją dat i godzin rozpoczęcia i zakończenia
* Pobranie rezerwacji
* Pobranie listy rezerwacji
* Usunięcie rezerwacji

# Wzorce projektowe

W projekcie zostały zastosowane koncepty projektowe takie jak SOLID, DRY, YAGNI i KISS, a ponadto wzorce projektowe:

* Observer – organizator oraz zaproszeni są powiadamiani gdy zostanie stworzona lub usunięta rezerwacja dla tych użytkowników
* Strategy – sortowanie sal jest wykonywane w zależności od przekazanego parametru w zapytaniu http
* Adapter – Obsługa sal w Usecase, ze względu na użycie wzorca projektowego Strategy, wymaga zastosowania dodatkowego adaptera implementującego interfejs wspólny dla wszystkich Usecase

# Wzorce architektoniczne

Projekt zostałzbudowany z wykorzystaniem architektury „Clean Architecture” – jest to rozbudowana wersja architektury heksagonalnej.

## Domain

Składa się z modeli, reprezentujących podstawowe encje, nie zawierających żadnej logiki biznesowej. Są to klasy User, Room oraz Reservation. Wszystkie implementują interfejs Identifiable.

Ponadto, definiują interfejsy używane przez wzorce projektowe Observer i Strategy.

## Infrastructure

Reprezentuje infrastrukturę używaną przez program – są to downstreamowe wyjścia programu. W przypadku tego projektu, jest to Repository (innymi przykładami może być QueuePublisher, FileStorage, HttpClient, itd.).

Repository składa się z interfejsu Repository<Identifiable>, klasy abstrakcyjnej BaseRepository<Identifiable> oraz konkretnych implementacji UserRepository, RoomRepository oraz ReservationRepository.

## Usecase

Reprezentuje logikę biznesową. Wykorzystuje infrastrukturę do wykonywania swoich funkcjonalności.

Usecase składa się z interfejsu Usecase<Identifiable>, klasy abstrakcyjnej BaseUsecase<Identifiable> oraz konkretnych implementacji UserUsecase, RoomUsecase oraz ReservationUsecase.

## Interface

Reprezentuje sposoby interakcji z programem – są to upstreamowe wejścia programu. W przypadku tego projektu, jest to Http (innymi przykładami może być QueueListener, CLI, GUI, itd.).

Http składa się z interfejsu HttpInterface<Identifiable>, klasy abstrakcyjnej BaseHttpInterface<Identifiable> oraz konkretnych implementacji UserHttpInterface, RoomHttpInterface oraz ReservationHttpInterface.