Traveler

Architecture Notebook

There is guidance within this template that appears in a style named InfoBlue. This style has a hidden font attribute that allows you to toggle whether it is visible or hidden in this template. Use the Microsoft® Word® menu **Tools > Options > View > Hidden Text** check box to toggle this setting. There is also an option for printing: **Tools > Options > Print**.

# Purpose

This document describes the decisions, constraints, justifications, significant elements, and any other overarching aspects of the system that shape the design and implementation.

[Always address Sections 2 through 6 of this template. Other sections are recommended, depending on the amount of novel architecture, the amount of expected maintenance, the skills of the development team, and the importance of other architectural concerns.]

# Architectural goals and constraints

[Insert a reference or link to the requirements that must be implemented to realize the architecture.

Formulate a set of goals that the architecture needs to meet in its structure and behavior. Identify critical issues that must be addressed by the architecture, such as: Are there hardware dependencies that should be isolated from the rest of the system? Does the system need to function efficiently under unusual conditions?] Wymagania

* **Bezpieczeństwo**: aplikacja powinna wspomagać ochronę przed nieautoryzowanym dostępem oraz przeciwdziałać utracie spójności danych.
* **Wydajność i skalowalność:**
  + **Czasy odpowiedzi:** Odpowiedź użytkownika na działanie systemu będzie miała średni czas odpowiedzi maksymalnie 5 sekund podczas obciążenia podanego poniżej. Wyjątkiem od tej reguły są raporty(czas ich uzyskania może zająć maksymalnie 30 sekund).
  + **Pojemność:** Produkt jednocześnie obsłuży 100 użytkowników w okresie od 8:00 do 16:00. W pozostałych okresach maksymalne obciążenie wyniesie 50 użytkowników.
  + **Uruchomienie:** Produkt uruchomi się w czasie 2 minuty
* **Pielęgnowalność**: aplikacja powinna ułatwiać wprowadzanie zmian.

# Decisions and justifications

[List the decisions that have been made regarding architectural approaches and the constraints being placed on the way that the developers build the system. These will serve as guidelines for defining architecturally significant parts of the system. Justify each decision or constraint so that developers understand the importance of building the system according to the context created by those decisions and constraints. This may include a list of DOs and DON’Ts to guide the developers in building the system.] Taktyki

|  |  |
| --- | --- |
| Goal | How achieved (Tactics) |
| Bezpieczeństwo | Autentykacja Autoryzacja Zapewnienie poufności danych (hasła) Walidacja danych otrzymanych od użytkownika Walidacja danych zapisywanych do bazy |
| Wydajność i skalowalność | Wprowadzenie cache bazy danych Wprowadzenie cache przetworzonych danych  Równoważenie obciążenia |
| Pielęgnowalność | Wprowadzanie pośredników - interfejsów |

# Architectural Mechanisms

[List the architectural mechanisms and describe the current state of each one. Initially, each mechanism may be only name and a brief description. They will evolve until the mechanism is a collaboration or pattern that can be directly applied to some aspect of the design.] Jak taktyki zrobić

## Autentykacja

Logowanie przy użyciu nazwy użytkownika i hasła. Zastosowanie mechanizmu Spring Security.

## Autoryzacja

Dostęp oparty o role i uprawnienia RBAC. Zastosowanie mechanizmu Spring Security.

## Zapewnienie poufności danych (hasła)

Zastosowanie funkcji skrótu bcrypt z pakietu Spring Security.

## Walidacja danych otrzymanych od użytkownika

Wykorzystanie standardu JSR-303 Bean Validation, zaimplementowanego przez Hibernate Validator.

## Walidacja danych zapisywanych do bazy

Wykorzystanie mechanizmu restrykcji zapisywanych danych bazy Postgresql. Pola w tabeli zostaną oznaczone poprzez Constraints takie jak Check, Not-Null, Unique. Check wymuszana na danych ewaluację warunku do wartości true. Not-Null wymusza obecność danych a Uniqe unikatowość.

## Wprowadzenie cache bazy danych

Wykorzystanie mechanizmu keszowania frameworku ORM Hibernate.

## Wprowadzenie cache przetworzonych danych

Wykorzystanie mechanizmu biblioteki Ehcache. Keszowane będą dane pobrane i przetworzone przez warstwę serwisową aplikacji. Zapisane w pamięci podręcznej informacje będą wykorzystywane przez kontrolery.

## Równoważenie obciążenia

Wykorzystanie dwóch instancji aplikacji i mechanizmu load balancera do rozdzielania żądań użytkowników.

## Wprowadzanie pośredników - interfejsów

Wykorzystanie paradygmatu odwrócenia kontroli i mechanizmu wstrzykiwania zależności do inicjalizacji obiektów, zaimplementowanego przez Spring. Przykładowe wykorzystanie:

**Definicja interfejsu**

public interface TouristEventService **{**

/\* ... \*/

**}**

**Definicja klasy serwisu**

@Service

public class TouristEventServiceImpl **implements** TouristEventService **{**

/\* ... \*/

**}**

**Wstrzyknięcie serwisu w kontroler**

@Controller

public class TouristEventController **{**

@Inject

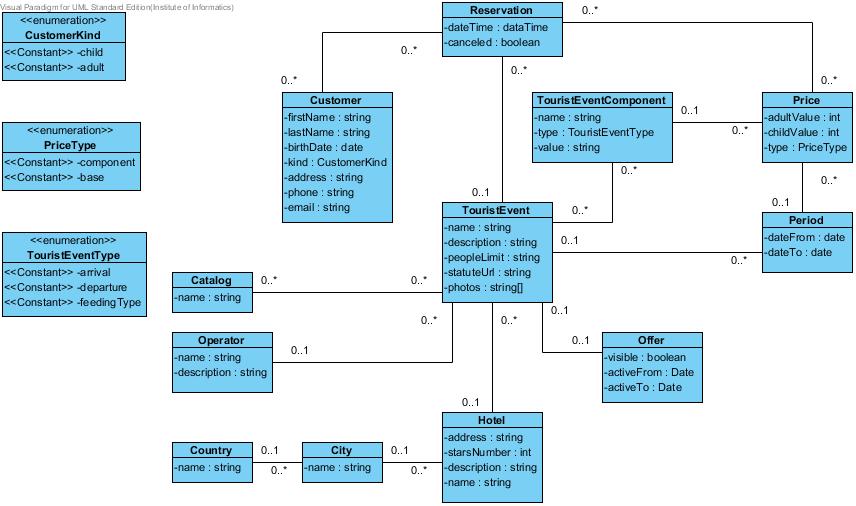
private TouristEventService touristEventService**;**

/\* ... \*/

**}**

# Key abstractions

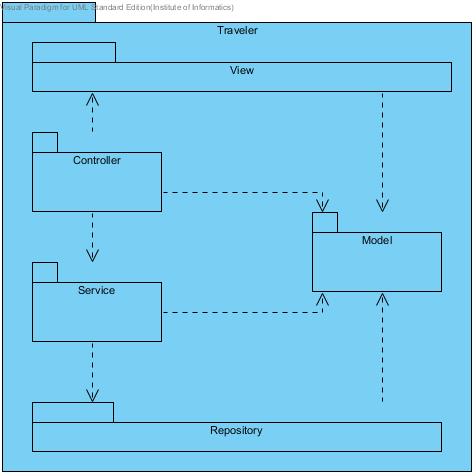
[List and briefly describe the key abstractions of the system. This should be a relatively short list of the critical concepts that define the system. The key abstractions will usually translate to the initial analysis classes and important patterns.] Perspektywa informacyjna



# Architectural views

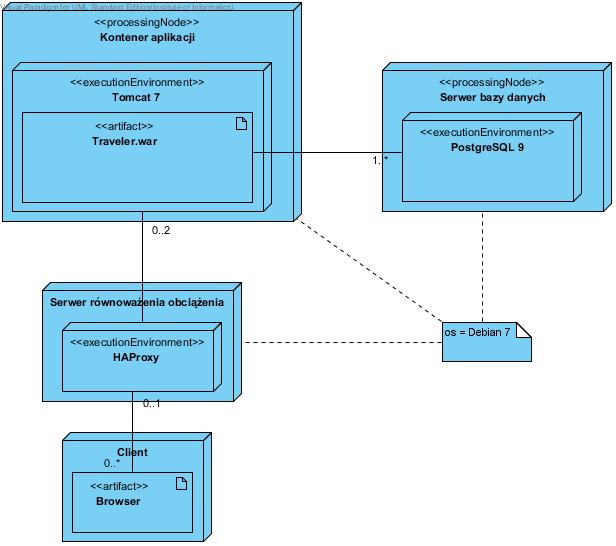
[Describe the architectural views that you will use to describe the software architecture. This illustrates the different perspectives that you will make available to review and to document architectural decisions.] + Krótkie opisy

## Perspektywa wytwarzania



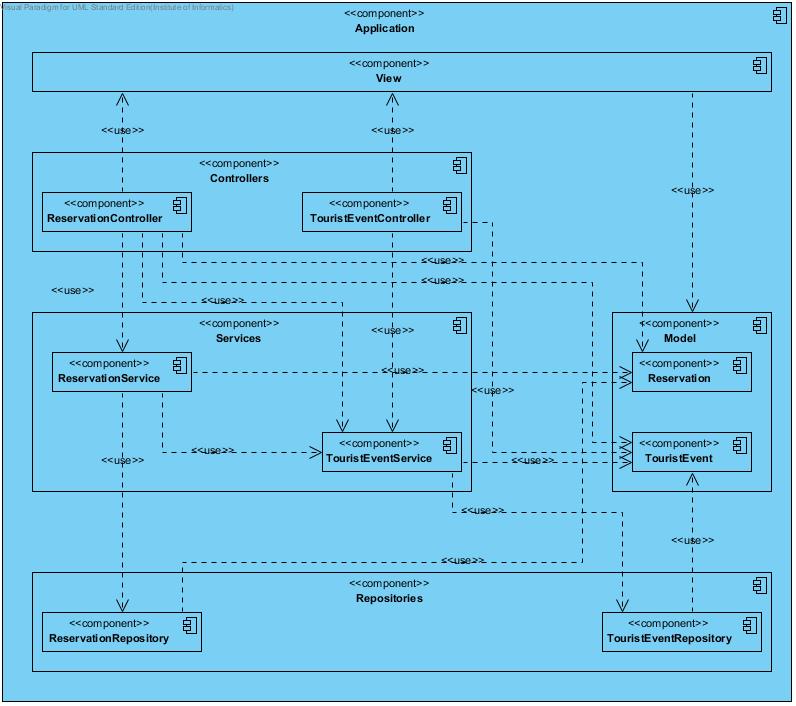
Aplikacja znajdować sie będzie w jednym głównym pakiecie Traveler. Pakiet ten będzie zawierał podpakiety, każdy reprezentujący osobną warstę aplikacji. Na samym sczycie będzie warstwa widoku, która wykorzystywać będzie obiekty modelu. Warstwa kontrolerów będzie odpowiedzialna za wywołanie odpowiednich akcji z warstwy serwisowej oraz zwrócenie odpowiedniego widoku. Swoje działania będzie opierać na wynikach zwróconych z warstwy serwisowej lub właściowościach obiektów modelu. Warstwa serwisowa wykonje logikę biznesową z użyciem modeli biznesowych. Do utrwalania danych wykorzystuje wzorzec repozytorium reprezentowany przez warstwę o takiej samej nazwie.

## Perspektywa rozmieszczenia



Aplikacje zainstalwoane będzie w kontenerach aplikacji Tomcat. Przewidujemy konieczność uruchomienia dwóch instancji aplikacji w celu rozłożenia obciążenia. Aplikacja będzie korzystać z rozproszonej bazy danych. Rozłożeniem obciążenia pomiędzy aplikacjami zajmie się load balancer.

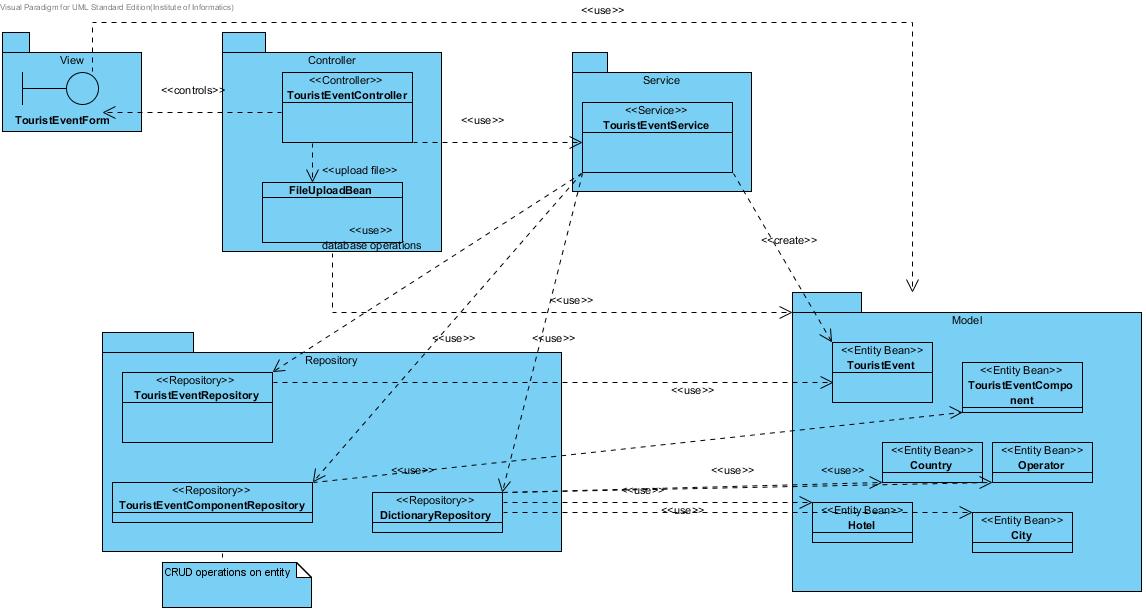
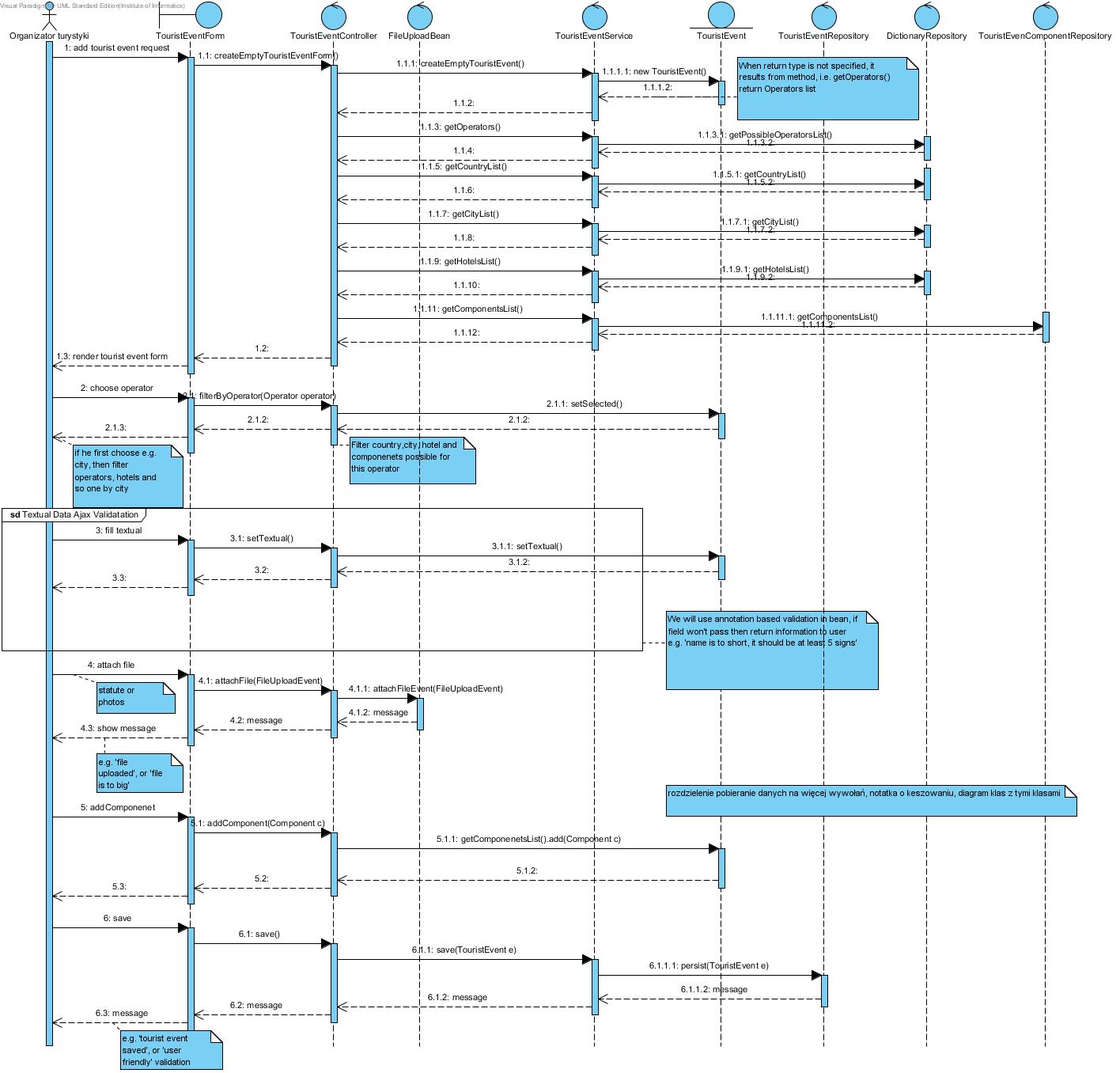
## Perspektywa funkcjonalna



Opis warstw perspektywy wytwarzania jest również prawdziwy dla perspektywy funkcjonalnej. W niej również przedstwione są przykładowe konkretne komponenty systemu.

# Use-case realizations (for selected use-cases)

**PU Dodanie danych imprezy turystycznej**



**Przypadek użycia: Dodanie terminów imprezy turystycznej**

