eSothebys

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

* Aplikacja użytkownika ma pracować z rozdzielczością ekranu stacji roboczej przy założeniu minimalnej obsługiwanej rozdzielczości 1024x768px
* Funkcje w systemie powinny być jednoznacznie nazwane, a ich obsługa nie powinna wymagać dodatkowych szkoleń
* System powinien posiadać przejrzysty i intuicyjny interfejs użytkownika
* Do obsługi systemu wymagana jest przeglądarka internetowa Internet Explorer 7.0, Opera 9.x, Firefox 2.0, Safari 5.0 lub nowsze
* Do obsługi systemu wymagany jest system operacyjny Windows Vista/7/8, Mac OSX, Linux Gentoo 12 lub nowsze
* System musi być w stanie obsłużyć do 10000 użytkowników jednocześnie
* Średni czas odpowiedzi systemu przy transakcjach odczytu danych nie może przekraczać 5 sekund
* Czas odpowiedzi systemu przy transakcjach dodawania danych do pojedynczego elementu nie może przekraczać 10 sekund
* Czas odpowiedzi systemu przy transakcjach edycji danych dotyczących nie więcej niż 300 obiektów nie może przekraczać 30 sekund
* System musi być dostosowany do przepisów prawa polskiego
* System powinien być dostępny 24 godziny na dobę
* System powinien być dostępne 99.9% czasu w roku
* System powinien być łatwy w utrzymywaniu

[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?]

# 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.]

|  |  |
| --- | --- |
| Goal | How achieved (Tactics) |
| Krótki czas odpowiedzi serwera | * Load balancer, który wybiera serwer, który będzie w stanie najszybciej obsłużyć zapytanie użytkownika * Cachowanie części danych |
| Autentykacja i autoryzacja | Zastosowanie spring security, które udostępnia te akcje |
| Logowanie wydarzeń na serwerze | Wykorzystanie biblioteki log4j, slf4j |
| Persystencja danych | Zastosowanie JPA + Hibernate i połączenia JDBC |
| Internacjonalizacja | I18n |

# 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.]

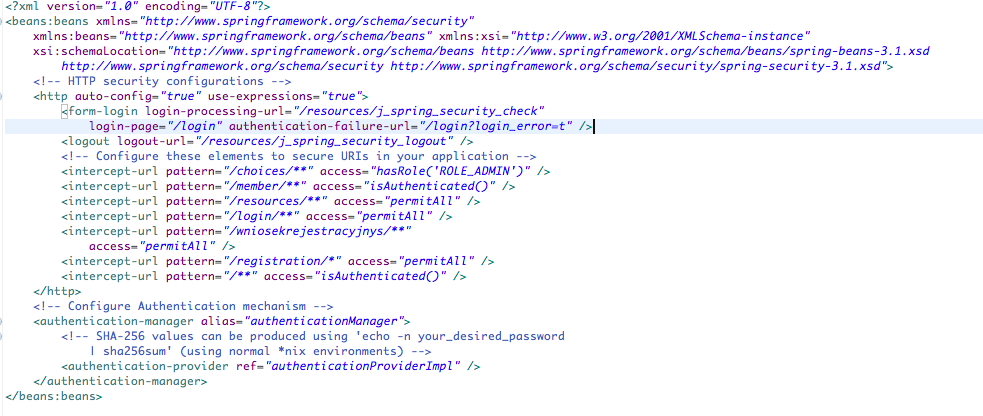
## Spring Security

Framework udostępnia mechanizmy zarówno autentykacji jak i autoryzacji. Wykorzystany został sposób autoryzacji za pomocą ról użytkowników, tzw. „Role-based access control” (RBAC).

Dostęp do zasobów jest wyryfikowany zarówno na poziomie URL zasobu, jak i w samym zasobie (widok JSPx) poprzez zastowanie odpowiednich JSP tagów dostarczonych przez Spring Security.

Za autoryzację odpowiedzialna jest klasa pl.wroc.pwr.wiz.io.psi.service.security.AuthenticationProviderImpl, która jest implenetacją interfejsu org.springframework.security.authentication.AuthenticationProvider z frameworka Spring Security. Klasa ta pobiera role z bazy danych, jakie posiada użytkownik próbujący zalogować się do systemu. Dodatkowo, Spring Security rejestruje zdarzenia wykorzystując loggery.

Poniżej zrzut ekranu przedstawia przykładowe konfiguracje zabezpieczeń.



## ORM

Persystencja danych jest obsługiwana przez 3 mechanizmy: Java EE EntityManager, adnotacje JPA oraz framework Hibernate. EntityManager wykorzystuje Hibernate do interakcji z bazą danych. Same klasy są oznaczane przez adnotacje JPA:

- @RooJpaActiveRecord

- @OneToMany, @ManyToOne, @ManyToMany konfigurujące liczności

- @NotNull

- @Temporal – wskazujące format daty

- @Column, @Table – definiujące nazwy kolumn, tabel

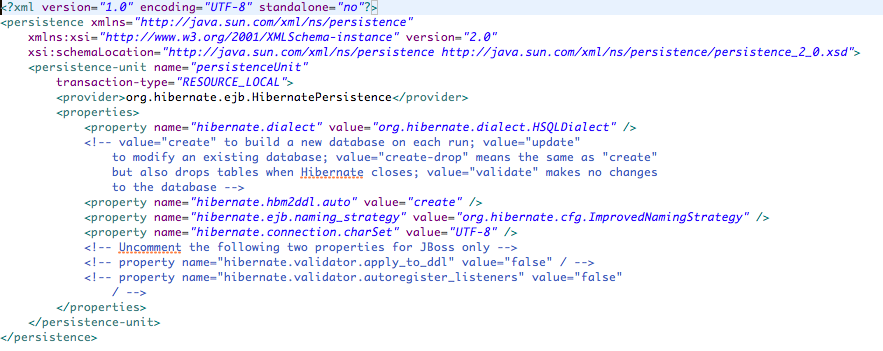
- @Value – definiująca wartość domyślną

Wykorzystanie dwóch pierwszych mechanizmów uniezależnia aplikację od konkretnego narzędzia ORM, a sama aplikacja jest napisana obiektowym paradygmatem. Aplikacja w pełni napisana obiektowo jest niezależna od samej bazy danych (szczególnie jej dialektu SQL).

Główna konfiguracja Entity Managera znajduje się w applicationContext.xml:



Konfiguracja mechanizmu odpowiedzialnego za ORM:



Na pierwszym obrazku pierwszy bean przedstawia konfigurację połączenia (pobierane z pliku database.properties), kolejno managera tranzakcji JPA, a na końcu fabrykę Entity Manager, która wykorzysuje wcześniej zdefiniowane połączenie oraz mechanizm ORM, który zdefiniowany jest w pliku persistence.xml (kolejny obrazek). Możemy zmienić narzędzie ORM Hibernate na inny bez wpływu na inne konfiguracje czy kod.

## JavaMailSender

Umożliwia wysyłanie maili z potwierdzeniem aktywacji wykorzystując smtp.

Kofiguraja JMS jest w pliku: email.properties. W systemie wykorzystany jest adres email założony w

gmail.com

## I18n

System wspiera internacjonalizację – system wspiera 3 wersje językowe. Widoki JSP odwołują się do kluczy, które są pobierane z plików tekstowych application*(\_flaga\_jezyka)*.properties, message*(\_flaga\_jezyka)*.properties.

## Hashowanie

Hasła są szyfrowane za pomocą algorytmu SHA-256. Dzięku temu w bazie danych nie przechowujemy haseł w postaci typu plain-text.

## Logowanie zdarzeń

Zdarzenia w aplikacji są rejestrowane w dzienniku zdarzeń wykorzystująć frameworki SLF4J oraz LOG4J.

Mechanizmy te pozwalają na łatwą konfigurację typów zdarzeń, także ich miejsce zapisu (konsola, plik tekstowy czy baza danych).

## AspectJ

Klasy modelowe opatrzone są adnotacjami generującymi kod, który powtarza się dla większości elementów, może on zostać wygenerowny. Tzw. Boilerplate. Wygenerowany kod tworzony jest w plikacj z rozszerzeniem \*.aj, który w momencie budowania klas źródłowych wstrzykiwany jest do klasy przez „Maven AspectJ Plugin”, a odpowiednio skonfigurowane środowisko programistyczne umożliwia jego wykorzystanie.

Dzięki wykorzystaniu aspektów klasa modelowa początkowo zawiera tylko prywatne pola. Programista nie musi przekopywać się przez linie kodu, które nie mają znaczenia w celu zrozumienia zawartości klasy.

Użyte adontacje:

* @RooJavaBean

Generuje metody związane z ideą JavaBean – gettery i setery dla pól w pliku \*\_Roo\_JavaBean.aj

* @RooToString

Generuje metodę toString() korzystając z biblioteki org.apache.commons.lang3 \*\_Roo\_ToString.aj

* @RooJpaActiveRecord

Generuje metody odstępu do bazy danych (podstawowe: CRUD), a także umożliwia wygenerowanie bardziej wyspecjalizowanych finderów. Znajdują się one w pliku \*\_Roo\_Jpa\_ActiveRecord.aj, a informacje o id obiektu oraz wersji trzymane są w pliku \*\_Roo\_Jpa\_Entity.aj

* @Configurable

Jest to informacja dla środowiska Spring, o tym, że obiekty tej klasy powinny uczestniczyć w procesie wstrzykiwania zależności.

## LoadBalancer

Load balancer, czyli równe rozkładnie obciążenia na serwery aplikacji realizowany jest za pomocą serwera Apache. Konfiguracja jest stosunkowo prosta:

<Proxy balancer://mycluster>  
BalancerMember http://192.168.1.50:8080  
BalancerMember http://192.168.1.51:8080  
</Proxy>  
ProxyPass /test balancer://mycluster

Wymaga podania adresów serwerów aplikacji. Resztą zajmuje się serwer. Udostępnione są 3 możliwe algorytmy:

* Request Counting
* Weighted Traffic Counting
* Pending Request Counting

# Key abstractions

* MVC
* Aspektowość
* Wstrzykiwanie zależności – odwrotna kontrola zależności

[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.]

# Architectural views

Diagram komponentów

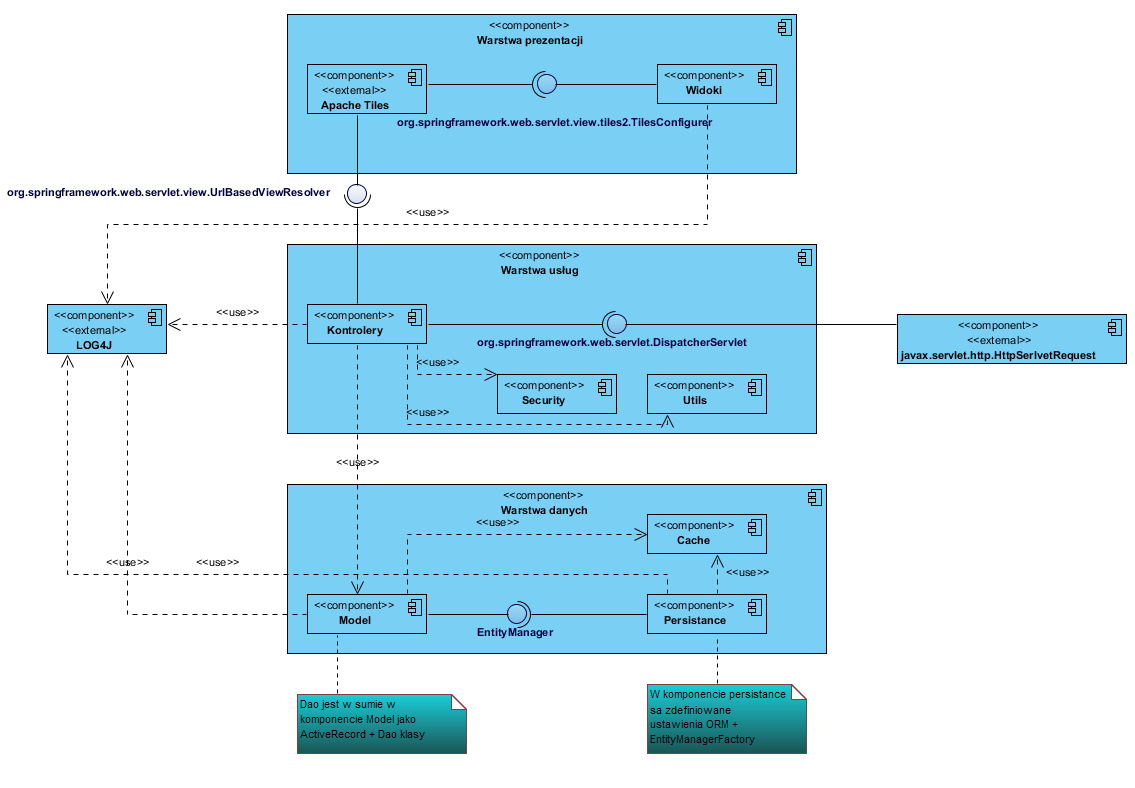


Diagram rozmieszczenia

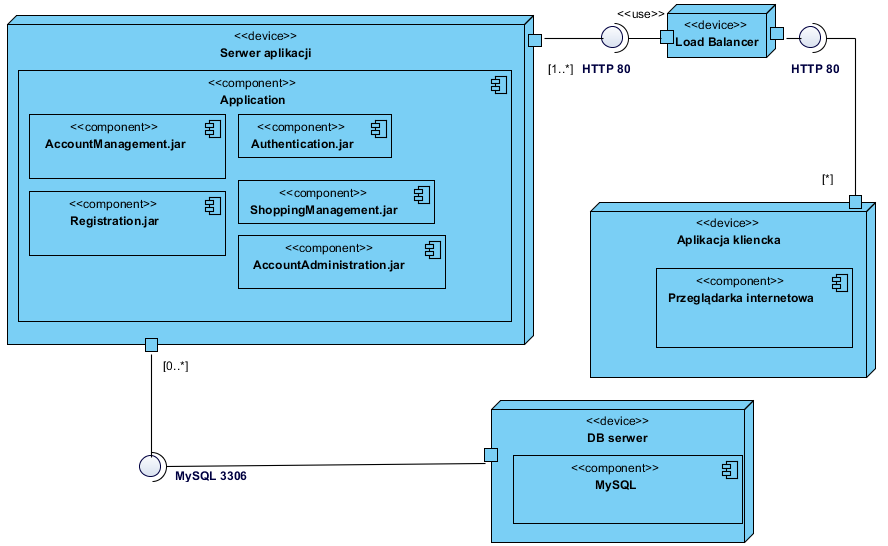


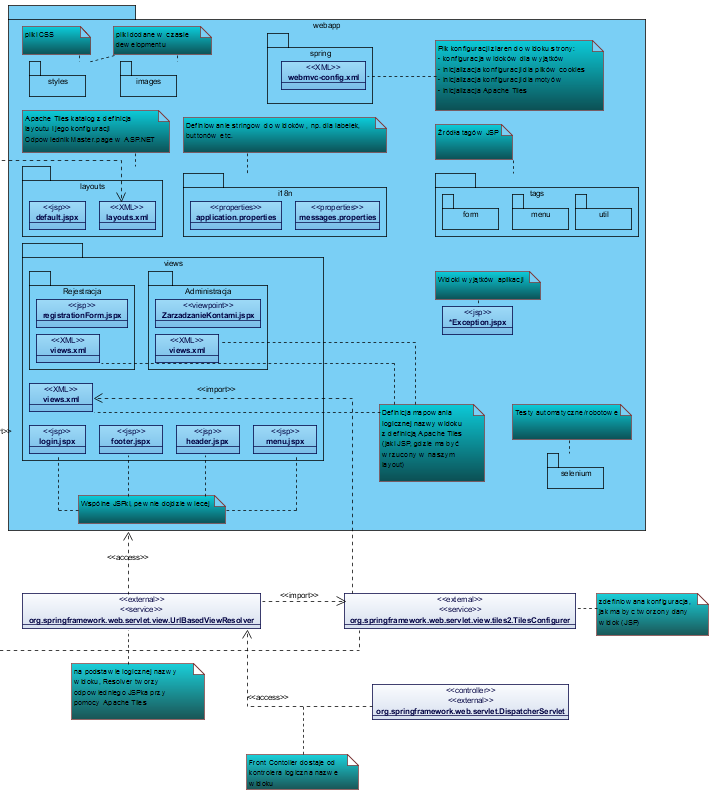
Diagram pakietów - widoki

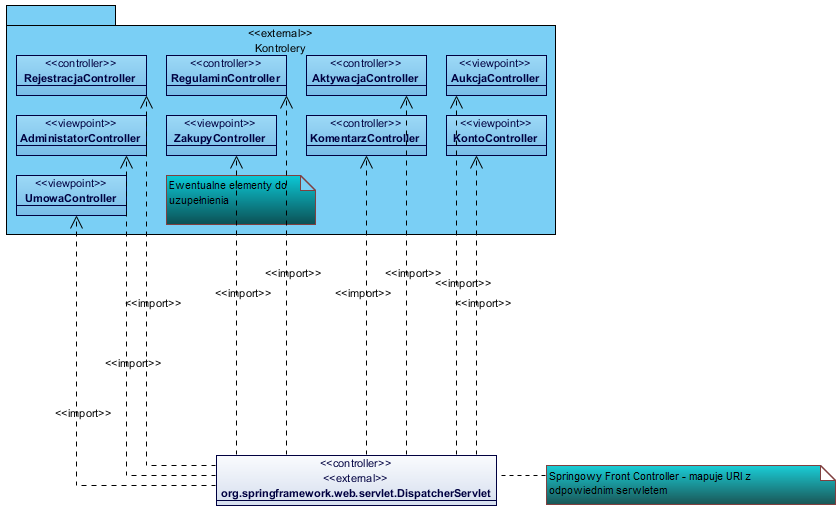
Diagram pakietów – kontrolery

Diagram pakietów – model

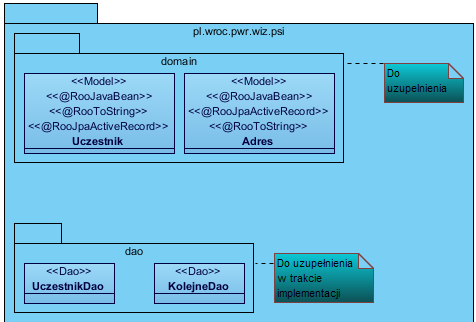


Diagram pakietów – persistance

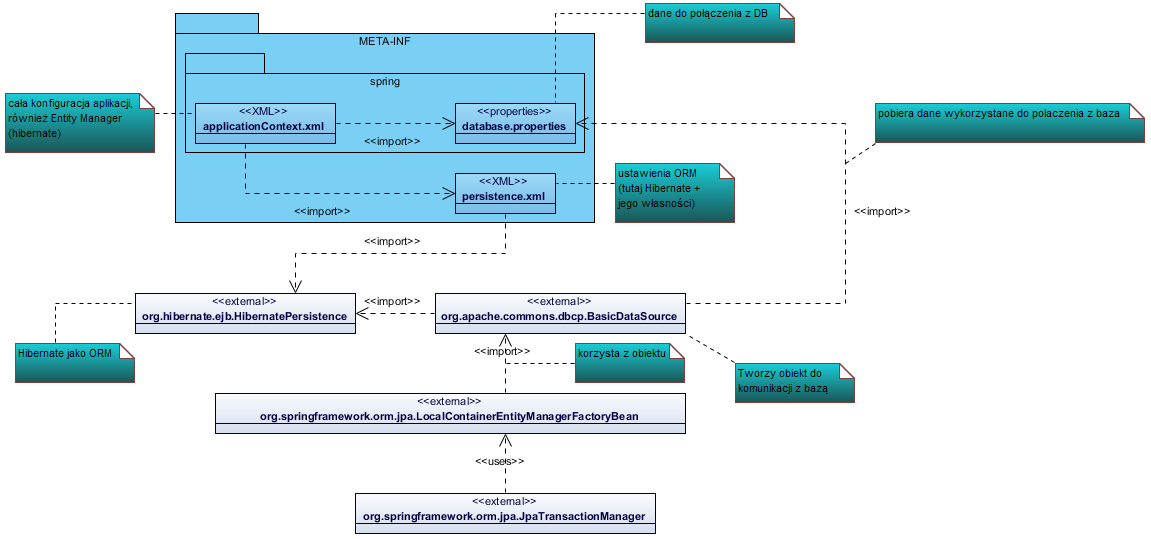


Diagram pakietów – security

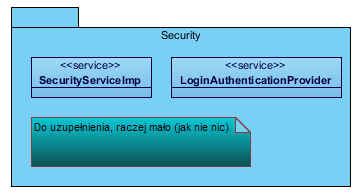


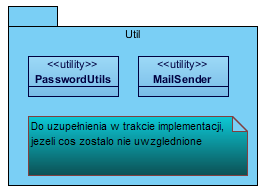
Diagram pakietów – utils  


Diagram pakietów – walidatory

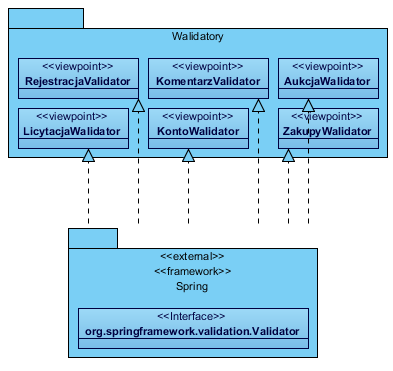
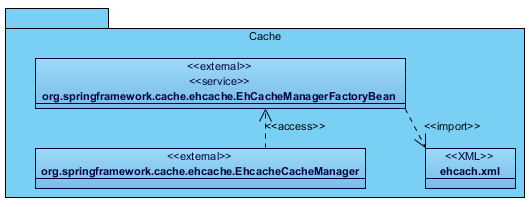


Diagram pakietów – cache



[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.]

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