|  |
| --- |
| Wydział Fizyki i Informatyki Stosowanej |
| **Gabinet Weterynaryjny** |
| Dokumentacja architektury systemu |
|  |
| Kantor Rafał  Kiszka Artur  Tomaszewski Sylwester |
| 2015-01-14 |

|  |
| --- |
| Poniższy dokument prezentuje przegląd architektury systemu którego robocza nazwa to „Gabinet Weterynaryjny”. |

[1 Wprowadzenie 3](#_Toc259470617)

[1.1 Zakres architektury systemu 3](#_Toc259470618)

[1.2 Cele i ograniczenia architektury 3](#_Toc259470619)

[2 Model środowiska docelowego 4](#_Toc259470620)

[3 Model architektury 4](#_Toc259470621)

[4 Realizacja przypadków użycia 5](#_Toc259470622)

[5 Wskazówki do implementacji 8](#_Toc259470624)

[5.1 Wykorzystywane biblioteki i gotowe komponenty 8](#_Toc259470625)

[6 Model danych 9](#_Toc259470626)

[7 Model testu 9](#_Toc259470627)

[8 Problemy i ryzyka 9](#_Toc259470628)

# Wprowadzenie

W skład architektury systemu wchodzą struktury oprogramowania oraz komponentów z których system będzie zbudowany. Dodatkowo architektura definiuje w jaki sposób komponenty porozumiewają się ze sobą.

## Zakres architektury systemu

Moduły z których składa się system:

 Aplikacja Webowa którą zawiera moduły:

 moduł autoryzacji użytkownika

 moduł terminarz

 moduł stan magazyny

 moduł wizyta

 moduł vademecum leków

 moduł generacja raportów

 baza danych MySQL

## Cele i ograniczenia architektury

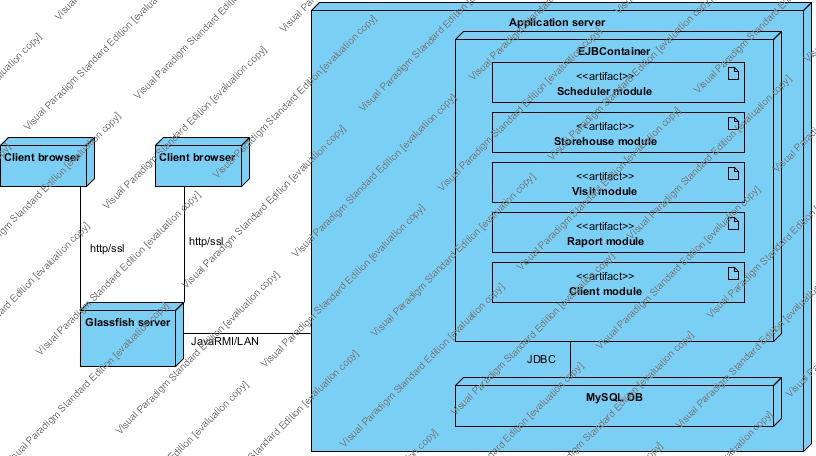
Tworzona w ramach projektu aplikacja ma oferować komfortowy styl pracy, który będzie spełniał oczekiwania tzw. użytkownika casualowego.

Interfejs musi być intuicyjny i prosty w obsłudze, a sama obsługa aplikacji nie powinna wymagać od klienta żadnej wiedzy na temat pracy aplikacji internetowych.

Wymagania sprzętowe muszą być bardzo niskie, a program powinien oferować maksymalną przenośność – możliwość uruchomienia na komputerach różnego typu i pod dowolnymi systemami operacyjnymi.

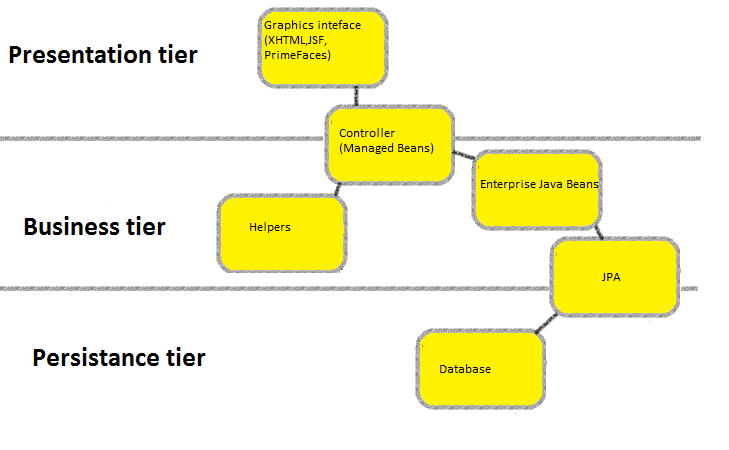
* Czas ładowania każdej strony powinien być niższy niż 3 sekundy
* Niedozwolone są jakiekolwiek wycieki danych – testy penetracyjne
* Wymaganie tworzenia instrukcji obsługi

# Model środowiska docelowego



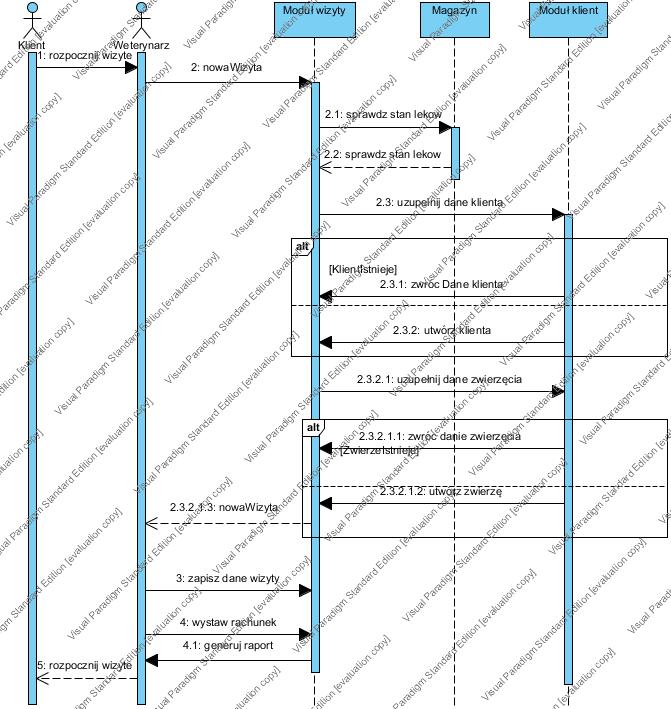
# Model architektury

Każdy wyżej wymieniony moduł można podzielić na osobne warstwy :

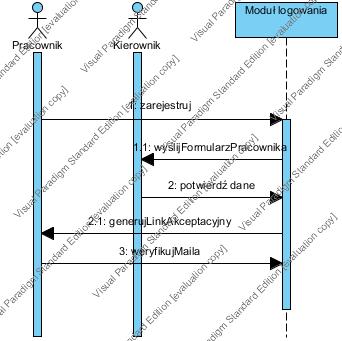


# Realizacja przypadków użycia

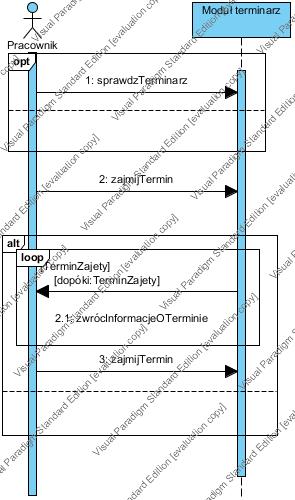
## GW0001 – Nowa wizyta



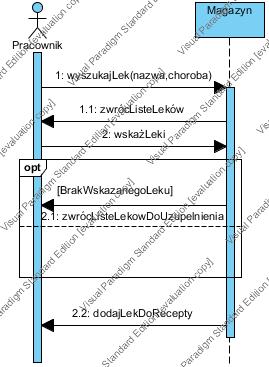
## GW0002 - Rejestracja w systemie

****

## GW0003 – Terminarz

****

## GW0004 – Dodaj lek



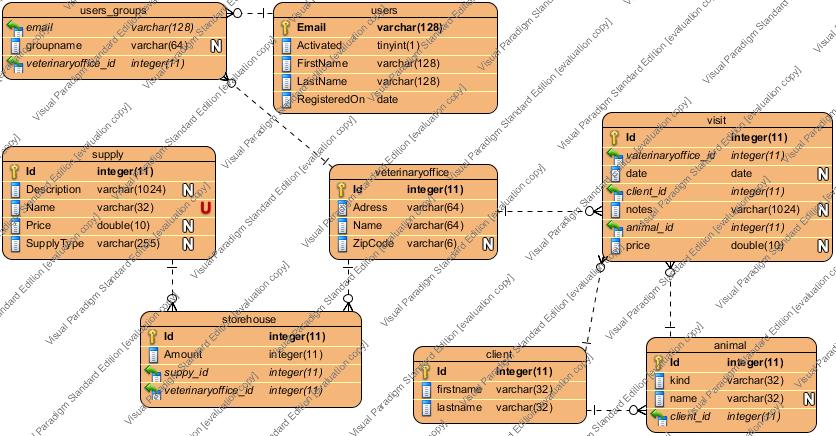
# Wskazówki do implementacji

* Należy pamiętać aby aby każda tablica która zawiera referencję do gabinetu weterynaryjnego była po tym polu indeksowana.
* Skupienie się głównie na obsłudze standardowych przypadków + możliwość łatwej konfiguracji/dodania funkcjonalności w przypadku gdy klient często obsługuje nietypowe przypadki.
* Należy zachować wyżej opisanie rozdzielenie na warstwy – nie pisać logiki biznesowej w widokach itp.

## Wykorzystywane biblioteki i gotowe komponenty

Wykorzystanie biblioteki pakietów komponentów PrimeFaces dzięki której zaoszczędzimy czas na tworzenie interfejsu użytkownika od zera. Dodatkowym plusem tej biblioteki jest to że jest ona całkowicie darmowa.

# Model danych



# Model testu

Podstawowym sposobem testowania systemu będą testy funkcjonalne. Środowiskiem testowym będzie założony „sztuczny” gabinet na którym dedykowana osoba będzie testowała funkcjonalności systemu. Dodatkowo podczas procesu rozwoju aplikacji będzie wdrożona ciągła integracja i testy regresyjne aby zmiany w systemie nie interferowały z poprzednio wdrożonymi funkcjonalnościami. W pewnym stopniu zostanie wdrożony lekko zmodyfikowany tak zwany „model V”. Zostaną również wykonane testy penetracyjne sprawdzające bezpieczeństwo aplikacji.

# Problemy i ryzyka

* Produkt może nie przyjąć się na rynku.
* Istnieje konkurencja : możliwość rozwoju oferty przez konkurencje.
* Produkt celuje w małe gabinety weterynaryjne : możliwość że specyfika pracy w każdym z nich jest na tyle różnorodna że ciężko będzie wdrożyć produkt bez istotnych modyfikacji
* Ograniczony czas na development może spowodować to że aplikacja będzie trudna w utrzymaniu.
* Wiąże się z poprzednim punktem : niektóre sytuacje mogą być trudne do przeprocesowania za pomocą standardowych mechanizmów zaimplementowanych w sytuacji; potrzeba przewidzenia obsługi przypadków nietypowych.