



Polsko-Japońska Wyższa Szkoła Technik Komputerowych
Wydział Zamiejscowy Informatyki w Gdańsku

Imię i nazwisko dyplomanta: Artur ŻURAWSKI Szymon KOWALCZYK Adam SMAKULSKI
Nr albumu: s7258 s7289 s7260
Kierunek studiów: APLIKACJE INTERNETOWE I SIECI
Rodzaj studiów: NIESTACJONARNE

PRACA DYPLOMOWA

Temat pracy: REZERWACJA I DOSTĘP POKOI HOTELOWYCH PRZY
UŻYCIU KODÓW QR.
Temat pracy w
języku angielskim: HOTEL ROOMS RESERVATION AND ACCESS USING QR
CODES.
Opiekun Pracy: dr inż. Stanisław SZEJKO
Wykonawcy: Szymon KOWALCZYK
Adam SMAKULSKI
Artur ŻURAWSKI

Streszczenie:

Celem projektu QRMagicKey było zaprojektowanie i stworzenie systemu umożliwiającego rejestrację pokoi hotelowych i dostęp do nich za pomocą zintegrowanego z systemem czytnika kodów QR. Przy tworzeniu systemu wykorzystano metodykę kaskadowo-przyrostową, w skład której weszły dwa wydania. System powstał w oparciu o język Java i Perl. System rezerwacji pokoi hotelowych współpracuje z czytnikiem kodów QR zbudowanym w oparciu o platformę Raspberry Pi.

Gdańsk, 2013

Oświadczenie autora pracy dyplomowej

Świadom odpowiedzialności prawnej oświadczam, że niniejszą pracę dyplomową w zakresie przeze mnie przedstawionym wykonałem samodzielnie i nie zawiera ona treści uzyskanych w sposób niezgodny z obowiązującymi przepisami.

Oświadczam również, że praca w przedstawionym przeze mnie zakresie nie była wcześniej przedmiotem procedur związanych z uzyskaniem tytułu ukończenia studiów wyższych.

Oświadczam ponadto, że niniejsza wersja pracy dyplomowej jest identyczna z załączoną wersją elektroniczną.

Temat projektu: Rezerwacja i dostęp pokoi hotelowych przy użyciu kodów QR.	Akronim: QR Magic Hotel
	Data ustalenia tematu: 2012.10.06
Opiekun: dr inż. Stanisław Szejko	Konsultanci: mgr. Adam Urbanowicz
Cel projektu: Stworzenie systemu umożliwiającego rejestrację pokoi hotelowych i dostęp do nich za pomocą zintegrowanego z systemem czytnika kodów QR.	
Rezultaty projektu: Gotowy system w postaci aplikacji webowej, współpracującej z czytnikiem kodów QR umożliwiających dostęp do pokoi.	
Miary sukcesu: Stworzenie systemu posiadającego funkcjonalności opisane w Specyfikacji Wymagań Systemowych. Wykonanie projektu wraz z dokumentacją techniczną do 2013.09.06	

Wykonawca	Numer albumu	Specjalizacja	Tryb studiów
Szymon Kowalczyk	7289	Aplikacje Internetowe i Sieci	niestacjonarne
Adam Smakulski	7260	Aplikacje Internetowe i Sieci	niestacjonarne
Artur Żurawski	7258	Aplikacje Internetowe i Sieci	niestacjonarne

Oceny Projektu

Semestr Dyplomowy I

Wykonawca	Nr albumu	Ocena sem. I	Data oceny	Podpis
Szymon Kowalczyk	7289			
Adam Smakulski	7260			
Artur Żurawski	7258			

Semestr Dyplomowy II

Wykonawca	Numer albumu	Ocena promotora	Podpis	Ocena I Recenzenta	Podpis	Ocena II Recenzenta	Podpis	Data oceny
Szymon Kowalczyk	7289							
Adam Smakulski	7260							
Artur Żurawski	7258							

Spis treści

Spis treści

1	Wstęp	1
2	Opis problemu	2
2.1	Przedstawienie problemu	2
2.2	Proponowane rozwiązanie	2
2.3	Wzbogacony wizerunek problemu i rozwiązania	4
3	Planowanie	5
3.1	Kontekst systemu	5
3.2	Zakres Projektu	6
3.3	Wizja konstrukcyjna	6
3.3.1	Założenia	6
3.3.2	Technologia i zamierzone środowisko	7
3.3.3	Wybór procesu wytwórczego	7
3.3.4	Zespół projektowy	8
3.3.5	Ograniczenia	8
3.3.6	Harmonogram	9
3.3.6.1	Pierwsze wydanie	9
3.3.6.1.1	Zakres projektu	9
3.3.6.1.2	Zakres systemu	9
3.3.6.2	Drugie wydanie	9
3.3.6.2.1	Zakres projektu	9
3.3.6.2.2	Zakres systemu	10
4	Analiza	11
4.1	Źródła i proces pozyskiwania wymagań	11
4.1.1	Szablon opisu wymagań	11
4.1.2	Aktorzy systemu	12
4.2	Udziałowcy	12
4.3	Wymagania	14
4.3.1	Wymagania funkcjonalne	14

4.3.2	Diagram przypadków użycia	22
4.3.3	Diagram sekwencji	28
4.3.4	Wymagania нефункционалне	29
4.4	Model logiczny	30
4.4.1	Diagram klas	30
4.4.2	Opis klas	31
4.4.3	Diagram stanów - Pokój	33
4.4.4	Diagram stanów - Rezerwacja	34
5	Projektowanie	35
5.1	Architektura aplikacji	35
5.2	Dekompozycja systemu na podsystemy	36
5.2.1	Wykorzystane wzorce projektowe	36
5.2.2	Diagram pakietów	37
5.2.3	Opis pakietów	38
5.3	Projekt struktury i algorytmy podsystemów	39
5.3.1	Diagram klas	39
5.3.2	Opis wybranych klas	46
5.3.3	Wybrane diagramy sekwencji	51
5.4	Decyzje projektowe	55
5.4.1	Języki programowania	55
5.4.2	Raspberry Pi	55
5.4.3	Framework Vaadin	55
5.4.4	Baza danych MySQL	56
5.4.5	Serwer aplikacji	56
5.5	Projekt bazy danych	57
5.5.1	Diagram ERD	58
5.5.2	Specyfikacja tabel	59
5.6	Projekt interfejsu użytkownika	62
5.6.1	Diagramy nawigacji	63
5.6.2	Szkielet strony - rozmieszczenie paneli	65
5.6.3	Wygląd ekranów	69
6	Wydanie I	72
6.1	Implementacja	72
6.1.1	Implementacja bazy danych	72
6.1.2	Moduły	73
6.1.2.1	Moduł rezerwacji pokoi	73
6.1.2.2	Moduł generowania i wysyłania kodu QR	73
6.1.3	Napotkane problemy	73
6.1.4	Wybrane algorytmy	73
6.1.5	Raspberry Pi	76
6.1.5.1	Specyfikacja	76

6.1.5.2	Urządzenia peryferyjne	76
6.1.5.3	System operacyjny	77
6.1.5.4	Przygotowanie środowiska	77
6.1.5.5	Moduł walidacji poprawności kodu QR	78
6.2	Testy	78
6.2.1	Testy jednostkowe	78
6.2.2	Testy integracyjne	78
6.2.3	Testy walidacyjne/systemowe	78
6.3	Raport końcowy	79
6.3.1	Porównanie założeń z realizacją	79
6.3.2	Rozkład pracy	79
6.3.2.1	Planowanie	80
6.3.2.2	Analiza	80
6.3.2.3	Projektowanie	81
6.3.2.4	Implementacja	81
6.3.2.5	Testowanie	82
6.3.2.6	Rozkład całkowity	82
7	Wydanie II	83
7.1	Implementacja	83
7.1.1	Moduły	83
7.1.1.1	Rejestracja	83
7.1.1.2	Graficzny wybór pokoi	83
7.1.1.3	Moduł osoby sprzątającej	83
7.1.1.4	Moduł administratora	83
7.1.1.5	Moduł managera	84
7.1.1.5.1	Raporty	84
7.1.2	Napotkane problemy	85
7.1.3	Przykładowe algorytmy	85
7.1.4	RaspberryPi	87
7.1.4.1	Urządzenia peryferyjne	87
7.2	Testy	88
7.2.1	Testy walidacyjne/systemowe	88
7.3	Raport końcowy	95
7.3.1	Moduły	95
7.3.2	Rozkład pracy	96
7.3.2.1	Analiza	96
7.3.2.2	Implementacja	97
7.3.2.3	Projektowanie	97
7.3.2.4	Testowanie	98
7.3.2.5	Rozkład całkowity prac	98
8	Prezentacja systemu	99

9 Wkład własny - Artur Żurawski	106
10 Wkład własny - Szymon Kowalczyk	113
11 Wkład własny- Adam Smakulski	119
Spis rysunków	123
A Lista plików załączonych na płycie CD	125
Bibliografia	126

Rozdział 1

Wstęp

Celem projektu było stworzenie systemu, który umożliwia rezerwację pokoju hotelowego udostępniając graficzną mapę dostępnych pokoi w wybranym hotelu. Po dokonaniu opłaty za wybrany okres rezerwacji system wysyła mailem kod QR w formie obrazka. Kod ten służy jako klucz otwierający drzwi zarezerwowanego pokoju. Do walidacji kodu został wykorzystany czytnik zbudowany w oparciu o komputer Raspberry Pi i podłączoną do niego kamerę USB.

Pomysł wziął się z chęci połączenia elementów niskopoziomowego sprzętu fizycznego współpracującego z wysokopoziomową aplikacją webową. System ma na celu zaprezentowanie możliwości wykorzystania kodów QR jako narzędzia do autoryzacji dostępu. Do jego działania został wykorzystany abstrakcyjny hotel, w którym elementy takie jak płatność czy system ryglowania drzwi, stanowią jedynie symulację. Do pozyskiwania wymagań zastosowana została metoda etnograficzna. Było to spowodowane brakiem podobnych rozwiązań dostępu do pokoi opartych na kodach QR.

Przy tworzeniu aplikacji internetowej zostało wykorzystane podejście obiektowe. W celu zapewnienia prawidłowego działania systemu przeprowadzone zostały testy integracyjne czytnika kodów QR oraz aplikacji webowej.

W efekcie powstał system rezerwacji pokoi hotelowych współpracujący z czytnikiem kodów QR zbudowanym w oparciu o platformę Raspberry Pi.

W skład grupy realizującej projekt wchodził: Szymon Kowalczyk, Adam Smakulski oraz Artur Żurawski.

Rozdział 2

Opis problemu

2.1 Przedstawienie problemu

Większość hoteli nie obsługuje Klientów w godzinach nocnych ponieważ wiąże się to z dodatkowymi kosztami (zatrudnienie osoby pracującej w recepcji w godzinach nocnych). Kolejnym problemem jest niewystarczająca ilość kluczy na pokój w przypadku gdy z pokoju miałyby korzystać więcej niż jedna osoba. Klucz jest kolejną fizyczną rzeczą o której trzeba pamiętać i pilnować. W przypadku zgubienia klucza hotel lub klient musi ponieść związane z tym koszty.

2.2 Proponowane rozwiązanie

System “QR Magic Key” jest w stanie obsłużyć Klienta o każdej porze dnia i nocy od momentu rezerwacji pokoju, do otworzenia pokoju. Wszystko za pomocą smartfona z połączeniem internetowym.

Zastąpienie go kodem QR, który można przechowywać w telefonie eliminuje ten problem a zastosowane w nim algorytmy zapewniają.

W dzisiejszych czasach telefonem płaci się za zakupy w sklepie, odprawia na lotnisku, steruje domowymi urządzeniami Audio/Video. W związku z tym nasz system jest kolejnym krokiem do ułatwienia życia codziennego.

System ma zastąpić kodami QR tradycyjne klucze, które powoli stają się reliktem minionego wieku. Użyte przez nas kody QR są bezpieczniejsze, tańsze, prostsze w użyciu a ich dystrybucja jest o wiele łatwiejsza.

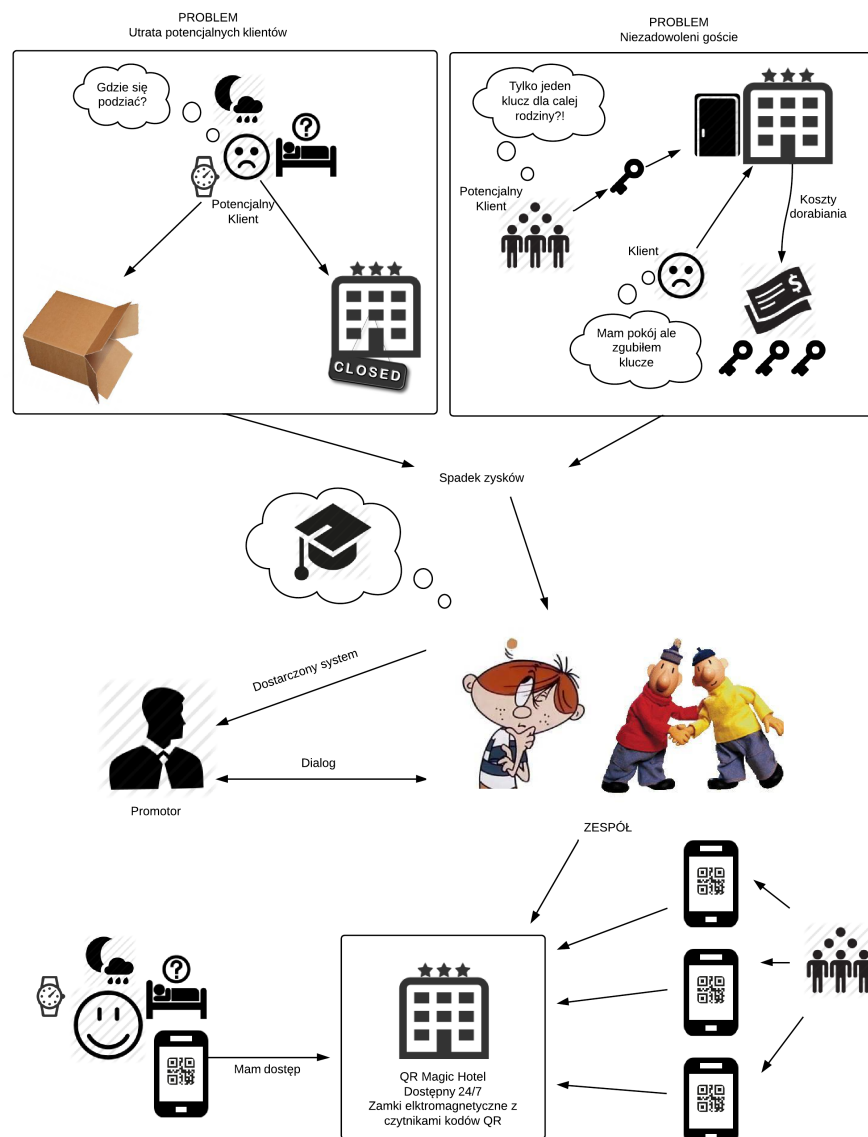
Aplikacja webowa pozwala na wybór położenia rezerwowanego pokoju, co nie jest oczywiste przy standardowych rozwiązaniach.

Poniższy opis przedstawia przewidywany proces rezerwacji pokoju przy wykorzystaniu naszego systemu:

- Użytkownik wybiera termin rezerwacji oraz parametry pokoju na stronie rezerwacji

- Użytkownik wybiera pokój z mapy przedstawiającej wszystkie dostępne pokoje o parametrach odpowiadających wybranym w poprzednim kroku
- Użytkownik wypełnienia formularz płatności i potwierdza płatność
- Użytkownik otrzymuje drogą mailową kod QR umożliwiający dostęp do zarezerwowanego pokoju
- Wraz z kodem Użytkownik udaje się do hotelu
- Użytkownik waliduje otrzymany kodu QR poprzez przyłożenie do czytnika zmieszczonego przy drzwiach pokoju hotelowego
- Po udanej walidacji kodu przez system Użytkownik uzyskuje dostęp do pokoju

2.3 Wzbogacony wizerunek problemu i rozwiązania



Rysunek 2.1: Rich Picture - Opis problemu i rozwiązanie

Głównym problemem w przeciętnym hotelu jest zależność gościa hotelowego od pracowników recepcji, którzy nie zawsze dostępni są w godzinach nocnych. Dzięki naszemu systemowi użytkownik uzyskuje pełną kontrolę na procesem rezerwacji i dostępu do pokoju niezależnie od pory dnia i nocy.

Rozdział 3

Planowanie

W trakcie fazy planowania powstał Wstępny Plan Projektu, w którym zawarty został opis celu, zakresu projektu oraz wizja naszego rozwiązania.

Niniejszy rozdział powstał na bazie znajdującego się w dodatku A, załączonego dokumentu WPP.

3.1 Kontekst systemu

System wykorzystuje wewnętrzną niewspółdzieloną bazę danych, z której korzysta interfejs rejestracyjny i poszczególne systemy walidacyjne zamków elektronicznych.

System dzieli się na dwie części: programową, na którą składa się aplikacja webowa wraz z bazą danych. sprzętową, na którą składa się czytnik kodów QR wraz z jednostką sterującą zamkiem elektrycznym. Jednostka sterująca komunikuje się z bazą danych walidując poprawność kodu QR.

Uprawnienia:

- Użytkownik nieposiadający konta w serwisie (niezalogowany) - Rejestracja w systemie, jak również korzystanie z wyszukiwarki dostępnych pokoi i analiza ich dostępności za pomocą graficznego interfejsu.
- Użytkownik posiadający konto w serwisie (zalogowany) – Rezerwacja pokoju, anulowanie rezerwacji, edycja danych kontaktowych.
- Manager - Generowanie raportów dotyczących stanu pokoi oraz historii wynajmu pokoiów.
- Osoba sprzątająca - Przyjmuje informacje o pokojach zwolnionych przez klientów w celu ich sprzątnięcia.
- Administrator - Tworzenie i usuwanie kont w systemie o różnym poziomie dostępu.

3.2 Zakres Projektu

Zakres projektu obejmuje:

- stworzenie pełnej dokumentacji projektowej
- implementację oraz testowanie aplikacji
- wprowadzenie przykładowych danych do bazy
- zaprogramowanie urządzenia walidującego kody QR

Najważniejsze funkcje przyszłego systemu to:

- możliwość wyszukania dostępności pokoi w podanych ramach czasowych
- możliwość rezerwacji wybranego pokoju na wybrany okres czasu (graficzna wizualizacja)
- możliwość rezygnacji z rezerwacji
- generowanie unikalnych kodów QR zastępujących standardowe klucze
- walidacja kodów QR
- nadawanie pokojom stanów (wolny, zajęty, do posprzątania, wyłączony z użytku)
- możliwość generowania raportów (wolne, zajęte, wyłączone_z_ruchu, lista klientów za dany okres)
- symulacja procesu płatności

3.3 Wizja konstrukcyjna

3.3.1 Założenia

Gość hotelowy zainteresowany rezerwacją pokoju hotelowego w pierwszej kolejności musi skorzystać z aplikacji webowej systemu. Niezalogowany użytkownik posiada jedynie możliwość wyszukiwania dostępnych pokoi. Aby z niej skorzystać użytkownik musi wybrać interesujący go okres rezerwacji pokoju. Po wybraniu daty prezentowana jest graficzna wizualizacja wolnych i zajętych pokoi w hotelu. Chcąc zarezerwować pokój klient hotelu będzie musiał założyć konto w systemie za pomocą formularza rejestracyjnego. Tym razem zalogowaniu użytkownik będzie posiadał możliwość wyboru interesującego go pokoju oraz przejścia do panelu płatności. Po dokonaniu opłaty wygenerowany zostanie kod QR umożliwiający dostęp do wybranego pokoju hotelowego. W kodzie będzie zawarty unikalny ciąg znaków oraz identyfikator rezerwacji. Obie informacje jednoznacznie pozwalają stwierdzić, czy użytkownik posługujący się danym kodem QR, w danym momencie ma

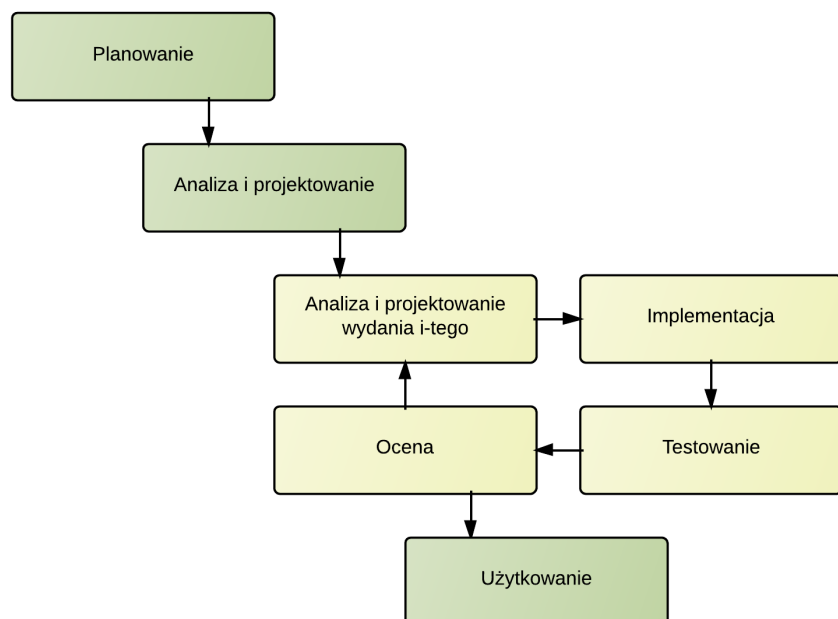
mieć możliwość dostępu do danego pokoju. Kod QR zostanie przesłany pod wskazany przy rejestracji adres mailowy. Gość hotelowy posiadający wygenerowany kod QR, w zdefiniowanym wcześniej terminie będzie miał możliwość otwarcia drzwi danego pokoju dowolną ilość razy aż do upływu terminu ważności danego kodu. Kod może być kopiowany dowolną ilość razy przez osobę rezerwującą pokój w celu umożliwienia dostępu do pokoju pozostałym gościom danego pokoju. W założeniu przy każdym pokoju hotelowym zamontowany zostanie czytnik kodów QR zintegrowany z elektrycznym rygłem drzwi. Po zbliżeniu kodu do czytnika nastąpi walidacja i w razie powodzenia zwolniona zostanie blokada zamka elektrycznego. Drzwi będą zamykane automatycznie. W praktyce symulacja będzie przeprowadzana na jednym egzemplarzu czytnika kodów QR z góry ustalonym (lecz edytowalnym) numerem odpowiadającego mu pokoju.

3.3.2 Technologia i zamierzone środowisko

Czytnik kodów QR wymaga użycia komputera pracującego pod kontrolą systemu Linux/Unix z możliwością obsługi języka Perl oraz podłączoną kamerą dostępną z poziomu systemu jako urządzenie blokowe. W naszym przypadku użyto komputera RaspberryPi z zainstalowanym systemem Debian/Linux z podłączoną kamerą internetową poprzez port USB. Aplikacja webowa wymaga serwera aplikacji umożliwiającego obsługę serwetów oraz serwera bazy danych. W naszym przypadku użyto serwera pracującego pod kontrolą systemu Linux CentOS. Jako serwer aplikacji posłużył nam Tomcat wraz z bazą danych MySQL. Aplikacja zostanie stworzona w oparciu o języki: Java, Html, Javascript, AJAX oraz Perl.

3.3.3 Wybór procesu wytwórczego

Wybrany przez nas model to połączenie modelu kaskadowego z modelem przyrostowym [1]. Wybór spowodowany jest faktem braku posiadania klienta, co za tym idzie brak możliwości zbierania informacji zwrotnych i sugestii. Kaskadowy model, jako główny podyktowany jest faktem posiadania wymagań, które są z góry ustalone przed przystąpieniem do fazy projektowania systemu. Model przyrostowy z kolei rozciąga się na etapy implementacji i testowania. Spowodowane jest to przyjętym przez nas planem, który zakłada w pierwszej kolejności utworzenie bardzo podstawowych funkcjonalności działającego systemu a następnie ich rozbudowywanie aż do momentu spełnienia wszystkich założonych przez nas wymagań. Model przyrostowy pozwala nam również na prowadzenie równoczesnych prac nad kilkoma modułami systemu oraz stopniową ich integrację i testowanie. Takie podejście daje nam także możliwość wprowadzenia ewentualnych zmian odnośnie wymagań, które mogą wynikać w trakcie rozbudowywania funkcjonalności systemu. Równie ważnym jest fakt, iż przy takich założeniach w prawie każdym momencie dysponować będziemy działającym projektem w zdegradowanej formie.



Rysunek 3.1: Model Procesu Wytwórczego

3.3.4 Zespół projektowy

W skład zespołu projektowego wchodziły 3 osoby. W trakcie realizacji decyzje dotyczące planowania czy implementacji były podejmowane wspólnie. Ogólny podział zakresu obowiązków został zrealizowany następująco:

Szymon Kowalczyk - główny programista aplikacji internetowej, dokumentacja

Adam Smakulski - programista aplikacji internetowej, testy, baza danych, dokumentacja

Artur Żurawski - programista czytnika kodów qr, tester, dokumentacja, serwery, integracja

3.3.5 Ograniczenia

- Z uwagi na nieprodukcyjną wersję systemu system opłat nie będzie zintegrowany z żadnym zewnętrznym systemem płatności. Do symulacji płatności zostanie użyty prosty formularz wymagający jedynie potwierdzenia dokonania płatności przez użytkownika.
- Czas realizacji: 14 czerwiec 2013
- Budżet: 300zł kosztów własnych
- Zasoby ludzkie: 3 studentów informatyki.

3.3.6 Harmonogram

3.3.6.1 Pierwsze wydanie

3.3.6.1.1 Zakres projektu

- Dokument Założeń Wstępnych [2012.11.03]
- Wstępny Plan Projektu [2012.11.17]
- Specyfikacja Wymagań Systemowych [2012.12.01]
- Projektowanie systemu [2013.01.17]
- Walidacja oprogramowania [2013.02.01]
- Dokumentacja testów [2013.02.09]

3.3.6.1.2 Zakres systemu

- Logowanie użytkownika do aplikacji internetowej [2012.12.22]
- Wyszukiwanie pokoju na podstawie podanych dat [2013.12.30]
- Wybór pokoju [2013.12.30]
- Rezerwacja pokoju [2013.01.12]
- Generowanie kodu QR [2013.01.12]
- Symulacja płatności [2013.01.20]
- Wysyłanie kodu QR na adres email zalogowanego użytkownika [2013.01.20]
- Walidacja kodu QR przy pomocy czytnika z zamontowaną kamerą [2013.01.26]

3.3.6.2 Drugie wydanie

3.3.6.2.1 Zakres projektu

- Architektura aplikacji [2012.02.05]
- Model logiczny [2013.02.26]
- Model projektowy [2013.03.12]
- Dokumentacja testów [2013.06.19]
- Nadanie końcowej formy i kształtu dokumentacji [2013.06.26]

3.3.6.2.2 Zakres systemu

- Graficzny wybór pokoi [2013.05.05]
- Panele osób sprzątających, administratora i managera [2013.05.19]
- Optymalizacja kodu i algorytmu czytnika kodów QR [2013.05.19]
- Generowanie raportów [2013.06.15]

Rozdział 4

Analiza

4.1 Źródła i proces pozyskiwania wymagań

Podstawą definiowania wymagań była metoda etnograficzna polegająca na pozyskiwaniu danych na podstawie własnych obserwacji i doświadczeń podczas korzystania z usług różnych hoteli na całym świecie. Nie ma na rynku podobnego rozwiązania jeżeli chodzi o dostęp do pokoi za pomocą kodu QR, z tego powodu autorzy projektu byli jednocześnie twórcami oraz odbiorcami produktu.

4.1.1 Szablon opisu wymagań

Szablon użytych poniżej tabel pochodzi z materiałów przedstawionych na przedmiocie „Projektowanie systemów informacyjnych” prowadzonym w ramach semestru V naszego rocznika, na uczelni PJWSTK.

- W polu identyfikator użyto następujących symboli:
 - UOB** - Udziałowiec ożywiony bezpośredni
 - UOP** - Udziałowiec ożywiony pośredni
 - UNP** - Udziałowiec nieożywiony pośredni
 - F** - wymaganie funkcjonalne
 - N** - wymaganie niefunkcjonalne
- W polu wymagania podane zostały wymagania, których źródłem był dany udziałowiec.
- W polu status znajduje się priorytet dla danego wymagania. Możliwymi wartościami są: kluczowy, pożądaný oraz opcjonalny.
- Pole sytuacje wyjątkowe opisuje sytuację w której dla konkretnego przypadku użycia istnieje alternatywny przebieg, inny niż główny.

- Pole udziałowiec opisuje podmiot (ożywiony bądź nieożywiony) powiązany z danym wymaganiem.
- Pole źródło opisuje osoby, które miały wpływ na definicję i jej kształt.

4.1.2 Aktorzy systemu

Użytkownik niezalogowany - każda niezalogowana osoba odwiedzająca serwis.

Użytkownik zalogowany - osoba posiadająca konto w serwisie, zalogowana posiadająca jedno z czterech uprawnień: gość hotelowy, manager, osoba sprzątająca, administrator.

Gość hotelowy - użytkownik zalogowany, posiadający uprawnienia do rejestracji pokoju, administrowania swoimi rezerwacjami.

Osoba sprzątająca - użytkownik zalogowany posiadający uprawnienia do podglądu listy pokoi wymaganych do posprzątania oraz zmiany ich stanu.

Manager hotelu - użytkownik zalogowany posiadający uprawnienia do generowania raportów, uproszczonej edycji właściwości pokoi (cena, stan, opis).

Administrator - użytkownik zalogowany posiadający uprawnienia do edycji kont użytkowników oraz szczegółowej edycji pokoi (dodanie nowego pokoju, numer pokoju).

4.2 Udziałowcy

Identyfikator	UOB.1
Nazwa	Promotor
Opis	Określa termin wykonania projektu, może wprowadzać nowe wymagania.
Typ	ożywiony, bezpośredni
Wymagania	F05

Identyfikator	UOB.2
Nazwa	Manager hotelu
Opis	Osoba zajmująca się generowaniem raportów dotyczących stanu pokoi oraz historii wynajmu pokoi. Ma możliwość manualnego przypisywania stanu pokojom.
Typ	ożywiony, bezpośredni
Wymagania	F16, F17, F18, F19, F19.1, F19.2, N01, N02, N04

Identyfikator	UOB.3
Nazwa	Twórcy
Opis	Tworzą, modyfikują funkcje systemu. Definiują wymagania.
Typ	ożywiony, bezpośredni

Identyfikator	UOB.4
Nazwa	Administrator
Opis	Osoba usuwająca oraz tworząca konta o różnym poziomie dostępu.
Typ	ożywiony, bezpośredni
Wymagania	F15, F15.1, F15.2, F15.3, F15.4, N01, N02, N04

Identyfikator	UOP.1
Nazwa	Osoba sprzątająca
Opis	Osoba otrzymująca informację o pokojach zwolnionych przez Klienta, do sprzątania.
Typ	ożywiony, pośredni
Wymagania	F19, F19.2, F20, F20.1, F21, N01, N02, N04

Identyfikator	UOP.2
Nazwa	Klient Hotelu
Opis	Osoba wynajmująca pokój za pośrednictwem strony internetowej wymagającej rejestracji oraz logowania. Ma możliwość otwierania drzwi przy pomocy otrzymanego kodu QR.
Typ	ożywiony, pośredni
Wymagania	F01, F01.1, F02, F03, F04, F05, F06, F07, F07.1, F07.2, F07.3, F08, F09, F10, F11, F12, F13, F14, F20, F20.2, N01, N02, N03, N04

Identyfikator	UNP.1
Nazwa	Kod QR
Opis	Medium zawierające zakodowane informacje, wymagające przetworzenia i walidacji.
Typ	nieożywiony, pośredni
Wymagania	F10, F12, F13, F14

4.3 Wymagania

4.3.1 Wymagania funkcjonalne

Identyfikator	F01
Status	Kluczowe
Nazwa	Rejestracja
Opis	System tworzy konto na podstawie podanych przez użytkownika danych (adres poczty elektronicznej oraz hasło). Konto jest niezbędne użytkownikowi do logowania do systemu.
Sytuacje wyjątkowe	Jeżeli adres e-mail istnieje w bazie danych użytkownik zostanie o tym poinformowany poprzez wyświetlenie informacji “konto o podanym adresie już istnieje” na stronie rejestracji.
Udziałowiec	Klient hotelu
Źródło	Twórcy

Identyfikator	F01.1
Status	Kluczowe
Nazwa	Weryfikacja pól rejestracji
Opis	System waliduje pola rejestracji pod względem zawartości oraz ilości znaków.
Sytuacje wyjątkowe	Jeżeli walidacja nie powiedzie się zostanie wyświetlona informacja obok pól które nie przeszły walidacji. Informacja będzie wyświetlana natychmiast po przejściu do następnego pola.
Udziałowiec	Klient hotelu
Źródło	Twórcy

Identyfikator	F02
Status	Kluczowe
Nazwa	Logowanie
Opis	Użytkownik powinien się logować przy użyciu formatki zawierającej pola: Login(adres e-mail), Hasło.
Sytuacje wyjątkowe	Jeżeli adres e-mail i/lub hasło nie zgadzają się z wpisem w bazie danych użytkownik zostanie o tym poinformowany poprzez wyświetlenie informacji “błędny login lub hasło” na stronie rejestracji.
Udziałowiec	Klient hotelu
Źródło	Twórcy

Identyfikator	F03
Status	Kluczowe
Nazwa	Rezerwacja pokoju
Opis	Funkcja pozwala na zarezerwowanie wybranego pokoju, w wybranym okresie czasu po uprzednim dokonaniu wpłaty.
Sytuacje wyjątkowe	Żądany pokój jest zarezerwowany. Brak wolnych pokoi, brak środków na koncie.
Udziałowiec	Klient hotelu
Źródło	Twórcy

Identyfikator	F04
Status	Kluczowe
Nazwa	Przedstawienie dostępności pokoi
Opis	Funkcja pozwala na wyświetlenie dostępnych pokoi w podanym terminie.
Sytuacje wyjątkowe	Brak dostępnych pokoi w podanym terminie. Wyświetlenie stosownej informacji.
Udziałowiec	Klient hotelu
Źródło	Twórcy

Identyfikator	F05
Status	Pożądane
Nazwa	Wybór pokoju (graficznie)
Opis	Funkcja prezentuje graficznie dostępność pokoi.
Sytuacje wyjątkowe	Brak
Udziałowiec	Klient hotelu
Źródło	Twórcy

Identyfikator	F06
Status	Kluczowe
Nazwa	Płatność online
Opis	System symuluje płatność online. Do symulacji płatności zostanie użyty prosty formularz wymagający jedynie potwierdzenia dokonania płatności przez użytkownika.
Sytuacje wyjątkowe	Brak dostępnych środków na wirtualnym koncie.
Udziałowiec	Klient hotelu
Źródło	Twórcy

Identyfikator	F07
Status	Pożądane
Nazwa	Zarządzanie rezerwacjami
Opis	Funkcja pozwala na zarządzanie dokonanymi rezerwacjami (Przedłużanie, anulowanie rezerwacji oraz funkcja ponownego wysłania kodu).
Sytuacje wyjątkowe	Panel zarządzania wyświetli pustą listę w przypadku, gdy żadna rezerwacja nie została dokonana.
Udziałowiec	Klient hotelu
Źródło	Twórcy

Identyfikator	F07.1
Status	Pożądane
Nazwa	Przedłużenie rezerwacji
Opis	Funkcja pozwala na przesunięcie terminu zdania pokoju o czas zdefiniowany przez gościa.
Sytuacje wyjątkowe	Przedłużenie nie jest możliwe w przypadku, gdy pokój został już zarezerwowany przez innego Klienta w danym terminie.
Udziałowiec	Klient hotelu
Źródło	Twórcy

Identyfikator	F07.2
Status	Pożądane
Nazwa	Anulowanie rezerwacji
Opis	Funkcja pozwala na anulowanie dokonanej wcześniej rezerwacji. Anulowanie rezerwacji nie może nastąpić później niż 2 dni przed terminem.
Sytuacje wyjątkowe	Próba anulowania rezerwacji później niż 2 dni przed terminem rezerwacji. Wyświetlenie informacji o Braku możliwości anulowania rezerwacji.
Udziałowiec	Klient hotelu
Źródło	Twórcy

Identyfikator	F07.3
Status	Pożądane
Nazwa	Funkcja ponownego wysłania kodu
Opis	Użytkownik ma możliwość ponownego wysłania kodu QR na podany adres email. Umożliwi ona dzielenie się kluczem pomiędzy kilkoma lokatorami.
Sytuacje wyjątkowe	Brak
Udziałowiec	Klient hotelu
Źródło	Twórcy

Identyfikator	F08
Status	Kluczowe
Nazwa	Chwilowa blokada pokoju
Opis	Po wskazaniu pokoju i zaakceptowaniu wyboru przez użytkownika pokój zostanie zablokowany na 10 minut, aby uniemożliwić rezerwację tego samego pokoju przez więcej niż jedną osobę. Jeżeli w ciągu dziesięciu minut płatność za rezerwację zostanie potwierdzona status pokoju zostanie zmieniony na wynajęty.
Sytuacje wyjątkowe	Brak
Udziałowiec	Klient hotelu
Źródło	Twórcy

Identyfikator	F09
Status	Pożądane
Nazwa	Edycja konta użytkownika
Opis	Funkcja pozwala na zmianę adresu e-mail, telefonu kontaktowego oraz hasła.
Sytuacje wyjątkowe	Brak
Udziałowiec	Klient hotelu
Źródło	Twórcy

Identyfikator	F10
Status	Kluczowe
Nazwa	Generowanie kodu QR
Opis	System generuje kod QR zawierający numer pokoju i losowo wygenerowany ciąg znaków, który musi zgadzać się z tym przechowywanym w bazie danych.
Sytuacje wyjątkowe	Brak
Udziałowiec	Klient hotelu, Kod QR
Źródło	Twórcy

Identyfikator	F11
Status	Kluczowe
Nazwa	Wysyłanie kodu QR
Opis	System automatycznie wysyła kod QR na adres email potwierdzony przy rejestracji po pomyślnym zakończeniu procesu rezerwacji.
Sytuacje wyjątkowe	Brak
Udziałowiec	Klient hotelu
Źródło	Twórcy
Wymagania	powiązane

Identyfikator	F12
Status	Kluczowe
Nazwa	Walidacja kodu QR
Opis	System przetwarza dane zawarte w obrazku zawierającym kod QR, porównuje dane z wartościami w bazie danych.
Sytuacje wyjątkowe	Dane niezgodne.
Udziałowiec	Klient hotelu, Kod QR
Źródło	Twórcy

Identyfikator	F13
Status	Kluczowe
Nazwa	System otwiera drzwi za pomocą kodu QR
Opis	System przetwarza dane zawarte w obrazku zawierającym kod QR, porównuje dane z wartościami w bazie danych i w przypadku zgodności danych wysyła impuls elektryczny otwierający zamek u drzwi.
Sytuacje wyjątkowe	Dane niezgodne. Użytkownik zostanie poinformowany poprzez zapalenie się czerwonej diody.
Udziałowiec	Klient hotelu, Kod QR
Źródło	Twórcy

Identyfikator	F14
Status	Pożądane
Nazwa	Informacja o prawidłowości kodu QR
Opis	System informuje o przyjęciu / odrzuceniu (graficznie, dźwiękowo) kodu QR podczas skanowania czytnikiem.
Sytuacje wyjątkowe	Brak
Udziałowiec	Klient hotelu, Kod QR
Źródło	Twórcy

Identyfikator	F15
Status	Kluczowe
Nazwa	Zarządzanie kontami
Opis	Możliwość zalogowania się do panelu udostępniającego funkcjonalność dla administratora.
Sytuacje wyjątkowe	Brak
Udziałowiec	Administrator
Źródło	Twórcy

Identyfikator	F15.1
Status	Kluczowe
Nazwa	Ustawienie uprawnień użytkowników
Opis	Administrator ma możliwość przypisania do konta wybranych uprawnień decydujących o funkcjonalności danego konta.
Sytuacje wyjątkowe	Brak
Udziałowiec	Administrator
Źródło	Twórcy

Identyfikator	F15.2
Status	Kluczowe
Nazwa	Tworzenie kont użytkowników
Opis	Administrator ma możliwość tworzenia nowych kont.
Sytuacje wyjątkowe	Brak
Udziałowiec	Administrator
Źródło	Twórcy

Identyfikator	F15.3
Status	Kluczowe
Nazwa	Edycja kont użytkowników
Opis	Administrator ma możliwość edycji dowolnych kont.
Sytuacje wyjątkowe	Brak
Udziałowiec	Administrator
Źródło	Twórcy

Identyfikator	F16
Status	Opcjonalne
Nazwa	Podgląd historii rezerwacji pokoi
Opis	Funkcja pozwala na podgląd historii rezerwacji pokoi w zadanym przedziale czasowym. Funkcja dostępna tylko dla konta z uprawnieniami Managera Hotelu.
Sytuacje wyjątkowe	Brak
Udziałowiec	Manager hotelu
Źródło	Twórcy

Identyfikator	F17
Status	Opcjonalne
Nazwa	Podgląd stanu pokoi
Opis	Funkcja pozwala na podgląd stanu pokoi. Funkcja dostępna tylko dla konta z uprawnieniami Managera Hotelu.
Sytuacje wyjątkowe	Brak
Udziałowiec	Manager hotelu
Źródło	Twórcy

Identyfikator	F18
Status	Opcjonalne
Nazwa	Drukowanie raportów
Opis	Generowanie raportów (lista rezerwacji, przychody, historia sprzątnięcia pokoi, użytkownicy) z możliwością filtrowania oraz eksportu do formatu pdf.
Sytuacje wyjątkowe	Brak
Udziałowiec	Manager hotelu
Źródło	Twórcy

Identyfikator	F19
Status	Kluczowe
Nazwa	Manualne przypisywanie stanu pokojom
Opis	Stany pokoju będzie ustawiał użytkownik z prawami dostępu "Manager hotelu" lub 'Osoba sprzątajaca' według potrzeby.
Sytuacje wyjątkowe	Brak
Udziałowiec	Manager hotelu, Osoba sprzątajaca
Źródło	Twórcy

Identyfikator	F19.1
Status	Kluczowe
Nazwa	Zmiana stanu pokoju na 'wyłączony z użytku'
Opis	Stan będzie ustawiał użytkownik z prawami dostępu "Manager hotelu" w przypadku gdy pokój nie powinien być dopuszczony do rezerwacji.
Sytuacje wyjątkowe	Brak
Udziałowiec	Manager hotelu
Źródło	Twórcy

Identyfikator	F19.2
Status	Kluczowe
Nazwa	Zmiana stanu pokoju na 'posprzątany'
Opis	Stan będzie ustawiał użytkownik z prawami dostępu "Osoba sprzątająca" w przypadku gdy pokój zostanie posprzątany
Sytuacje wyjątkowe	Brak
Udziałowiec	Osoba sprzątająca
Źródło	Twórcy

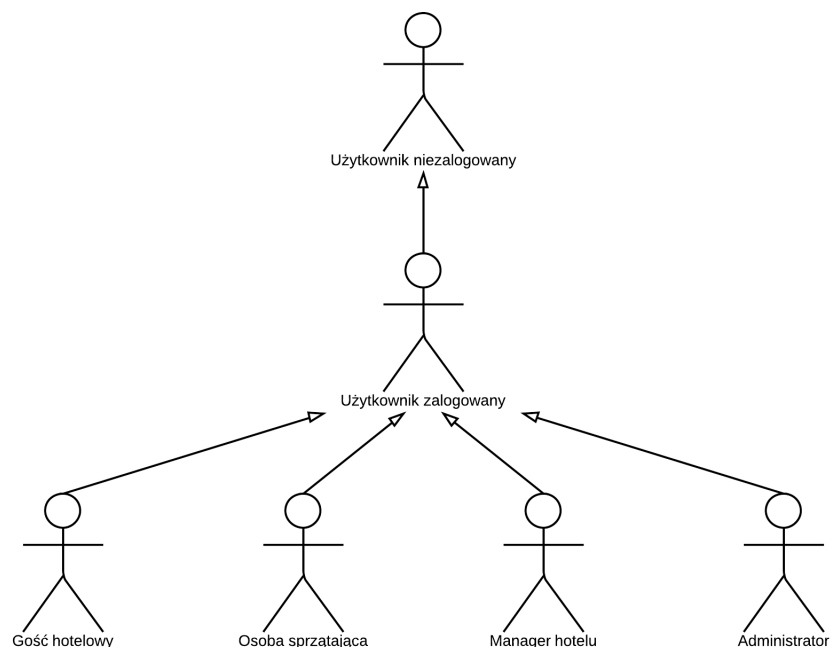
Identyfikator	F20
Status	Kluczowe
Nazwa	Zmiana stanu pokoju na 'do posprzątania'
Opis	Stan pokoju do posprzątania będzie ustawiany automatycznie przez system w przypadku gdy pokój był wynajęty i minęła doba hotelowa.
Sytuacje wyjątkowe	Brak
Udziałowiec	Osoba sprzątająca
Źródło	Twórcy

Identyfikator	F21
Status	Pożądane
Nazwa	Lista pokoi do posprzątania w panelu sprzątaczk.
Opis	Lista będzie wyświetlać pokoje wymagające sprzątnięcia.
Sytuacje wyjątkowe	Brak
Udziałowiec	Osoba sprzątająca
Źródło	Twórcy

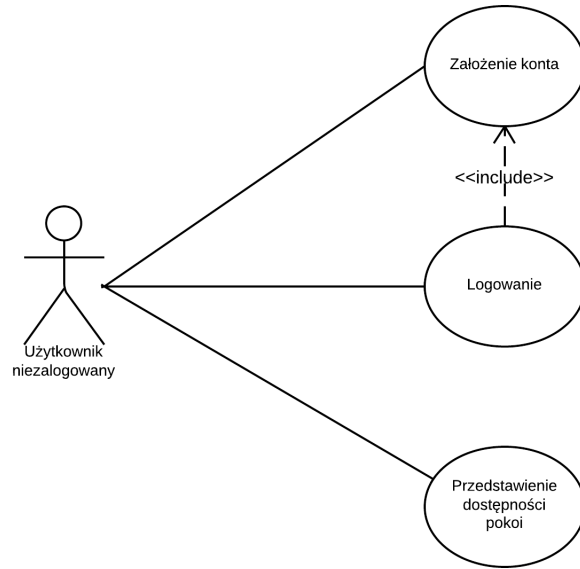
Identyfikator	F22
Status	Pożądane
Nazwa	Zmiana statusu pokoju przez sprzątaczkę
Opis	Osoba sprzątająca będzie miała możliwość zmiany stanu pokoju z "do posprzątania" na "posprzątany"
Sytuacje wyjątkowe	Brak
Udziałowiec	Osoba sprzątająca
Źródło	Twórcy

Identyfikator	F23
Status	Pożądane
Nazwa	Historia sprzątania pokoi
Opis	Lista będzie wyświetlać historię zmiany statusu pokoi z "do posprzątania" na "posprzątany".
Sytuacje wyjątkowe	Brak
Udziałowiec	Osoba sprzątająca
Źródło	Twórcy

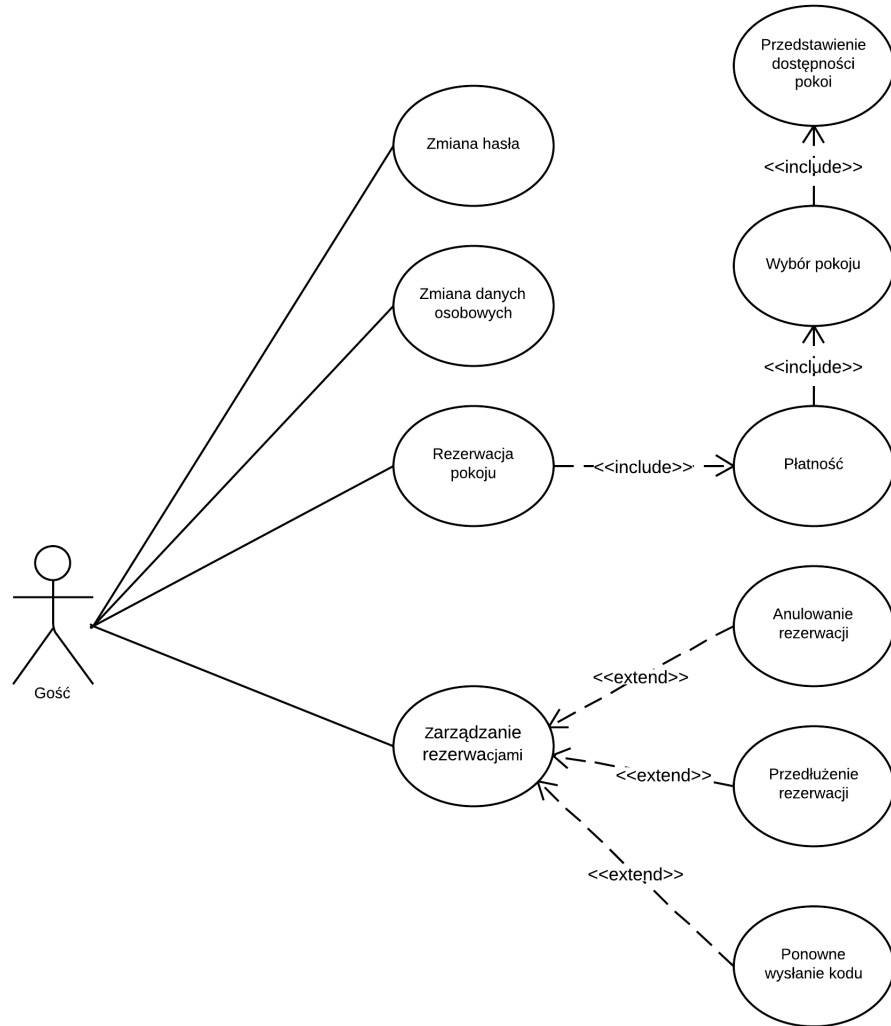
4.3.2 Diagram przypadków użycia



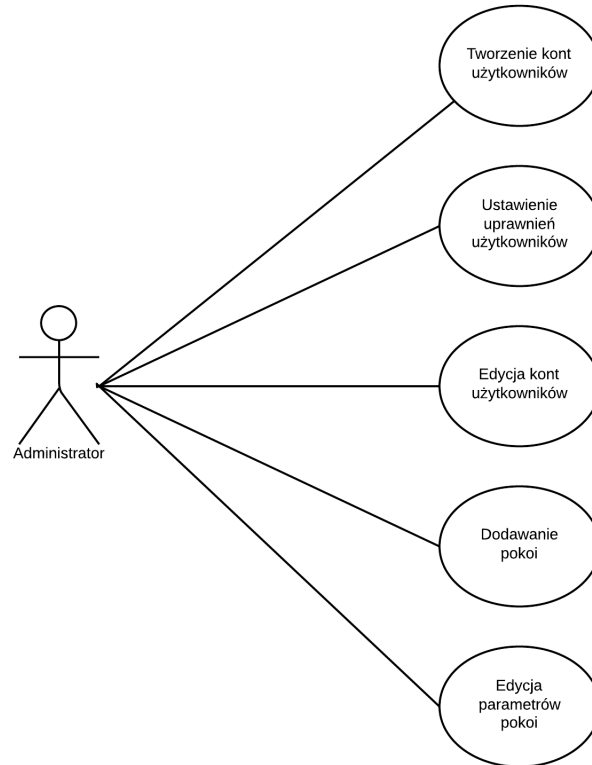
Rysunek 4.1: Diagram przypadków użycia - Aktorzy



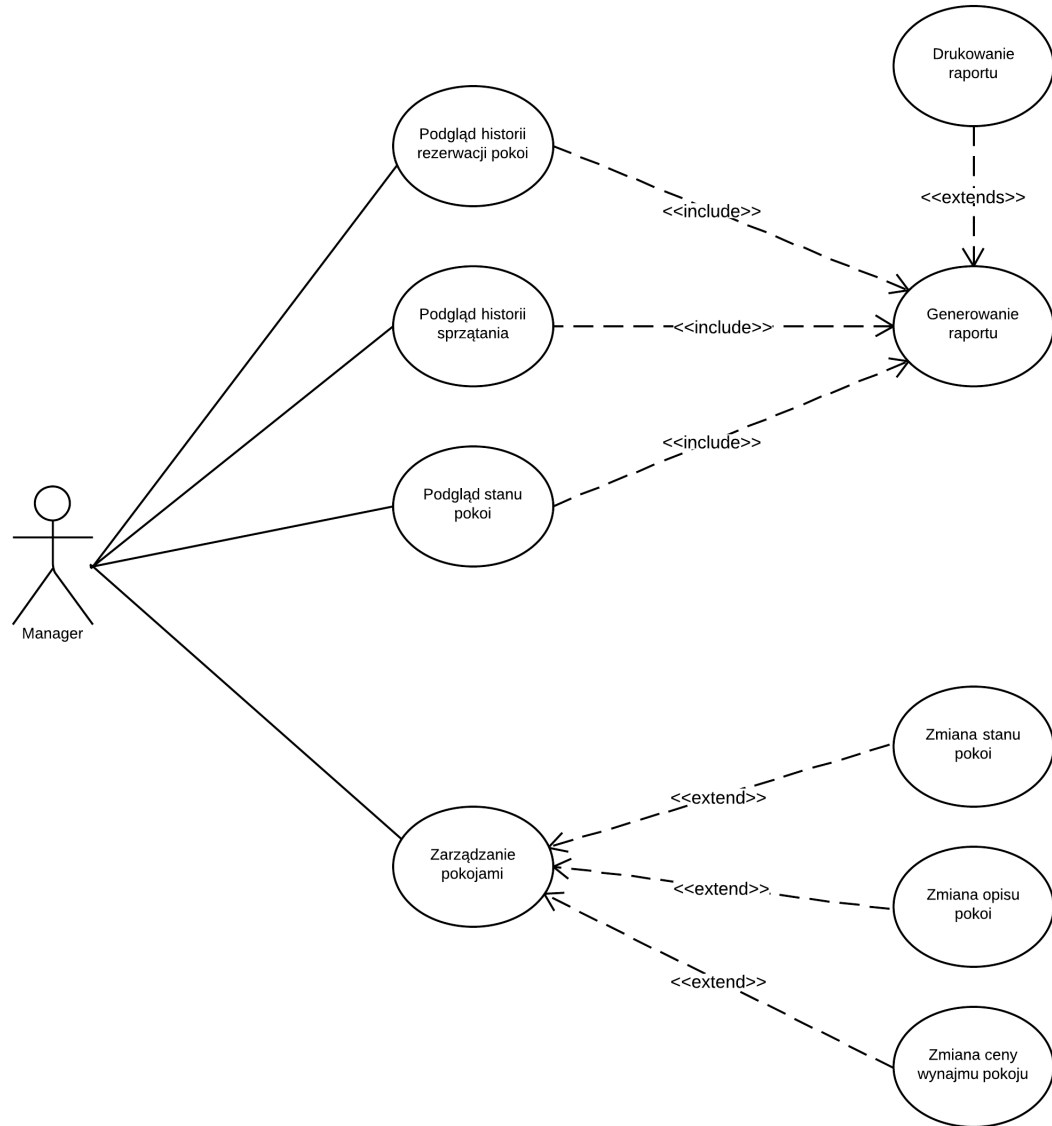
Rysunek 4.2: Diagram przypadków użycia - Niezalogowany Gość



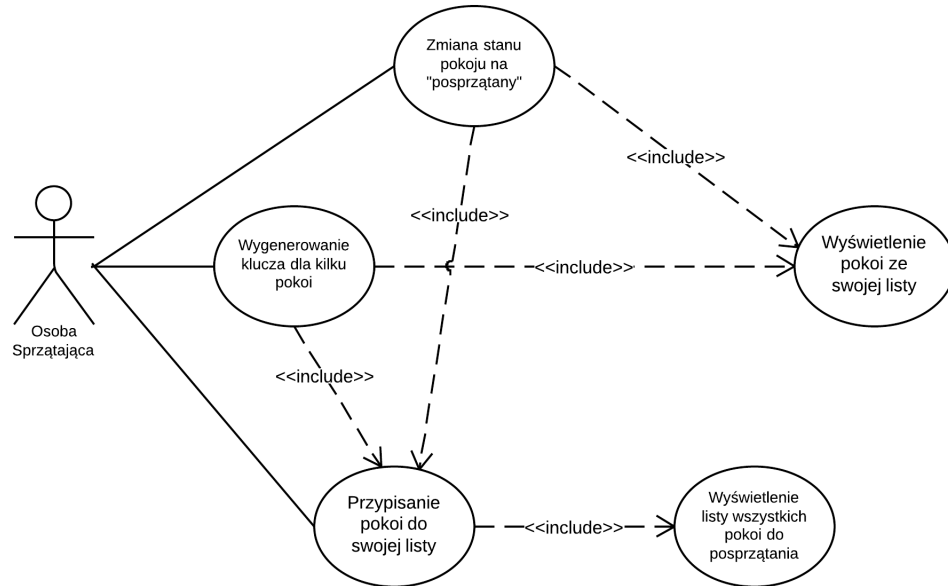
Rysunek 4.3: Diagram przypadków użycia - Zalogowany Gość



Rysunek 4.4: Diagram przypadków użycia - Zalogowany Administrator

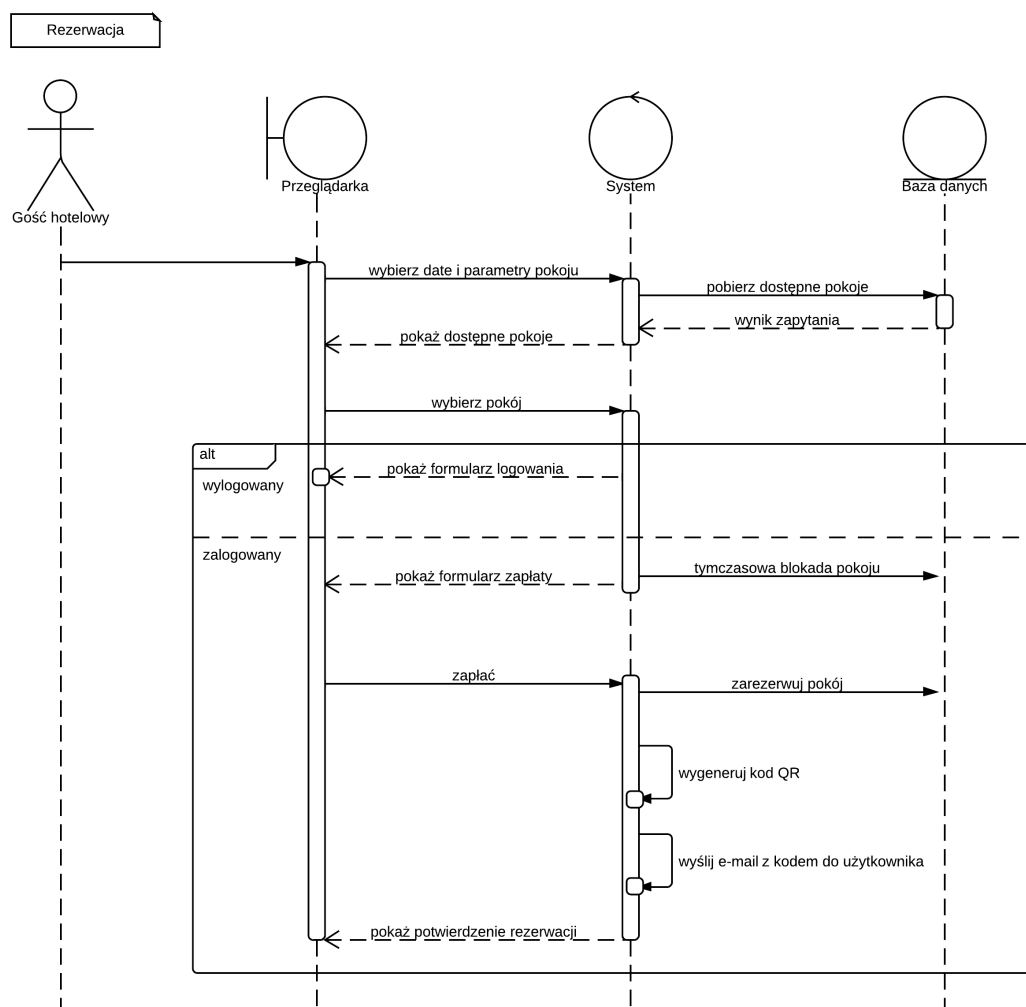


Rysunek 4.5: Diagram przypadków użycia - Zalogowany Manager



Rysunek 4.6: Diagram przypadków użycia - Zalogowana Osoba sprzątająca

4.3.3 Diagram sekwencji



Rysunek 4.7: Diagram sekwencji przebiegu rezerwacji

4.3.4 Wymagania нефункционалне

Identyfikator	N01
Status	Pożądane
Nazwa	Cross-Browser
Opis	System powinien być obsługiwany w następujących przeglądarkach: Firefox 19+, Chrome 26+, Internet Explorer 9+.
Sytuacje wyjątkowe	Brak
Udziałowiec	Klient hotelu, Manager Hotelu, Administrator, Osoba sprzątająca
Źródło	Twórcy
Wymagania powiązane	Brak

Identyfikator	N02
Status	Pożądane
Nazwa	Wydażność systemu nie jest ograniczona logiką działania systemu
Opis	Szybkość działania systemu i liczba obsługiwanych jednocześnie sesji będzie zależeć wyłącznie od wydajności systemu który będzie hostował aplikacje i bazę danych. Logika działania systemu nie będzie narzucać niepotrzebnych ograniczeń.
Sytuacje wyjątkowe	Brak
Udziałowiec	Klient hotelu, Manager Hotelu, Administrator, Osoba sprzątająca
Źródło	Twórcy
Wymagania powiązane	Brak

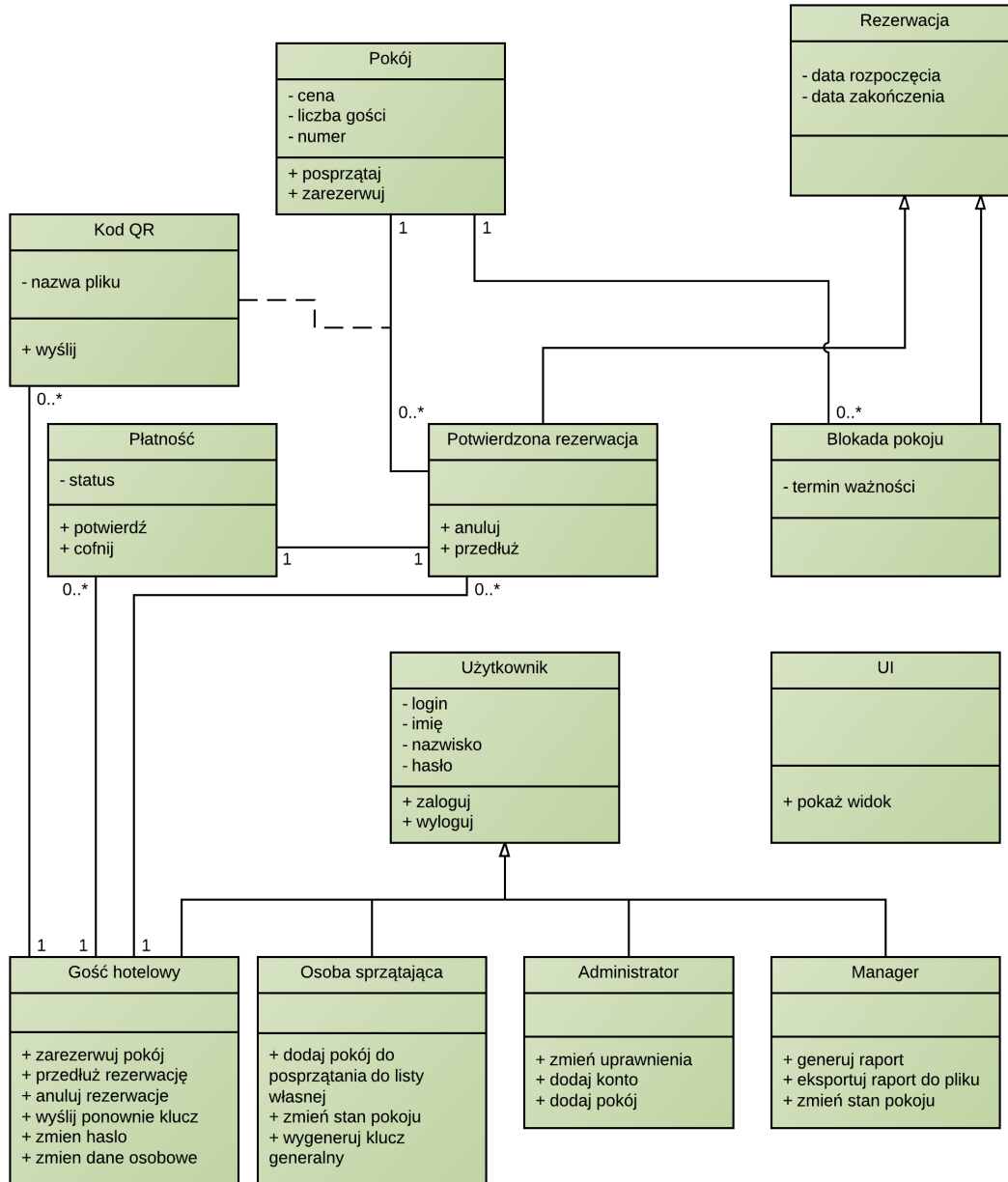
Identyfikator	N03
Status	Pożądane
Nazwa	Dostępność systemu 24/7
Opis	System ma umożliwić rezerwację pokoju o każdej porze dnia i nocy.
Sytuacje wyjątkowe	Brak prądu
Udziałowiec	Klient hotelu
Źródło	Twórcy
Wymagania powiązane	Brak

Identyfikator	N04
Status	Opcjonalne
Nazwa	Usability
Opis	Obsługa powinna być zrozumiała i nieskomplikowana dla Użytkownika, Osoby sprzątającej oraz Administratora. Funkcje powinny być łatwo dostępne aby ułatwić i przyspieszyć obsługę.
Sytuacje wyjątkowe	Brak
Udziałowiec	Manager hotelu, Klient hotelu, Osoba sprzątająca, Administrator
Źródło	Twórcy
Wymagania powiązane	Brak

4.4 Model logiczny

4.4.1 Diagram klas

Poniższy diagram oraz opis klas utworzony był w początkowym etapie projektowania. Wraz z ewolucją projektu uległ znaczącym zmianą i jego obecna, szczegółowa wersja wraz z metodami oraz atrybutami znajduje się na stronie 39.



Rysunek 4.8: Diagram Klas

4.4.2 Opis klas

Room - Klasa odpowiedzialna jest za prezentację stanu pokoju. Pokój może przybierać następujące stany: wolny, do sprzątnięcia, do posprzątania, zarezerwowany.

Reservation - Klasa odpowiedzialna za rezerwacje pokoju.

QRCode - Klasa odpowiedzialna za generowanie i wysyłkę kodu QR na podstawie danych z rezerwacji.

Payment - Klasa odpowiedzialna za opłacenie zarezerwowanego pokoju.

ConfirmedReservation - Klasa odpowiedzialna za potwierdzenie prawidłowego dokonania płatności.

TmpReservation - Klasa odpowiedzialna za czasową blokadę pokoju w celu zapobieżenia rezerwacji tego samego pokoju przez wielu użytkowników.

User - Klasa odpowiedzialna za edycje danych użytkowników oraz rezerwację pokoi.

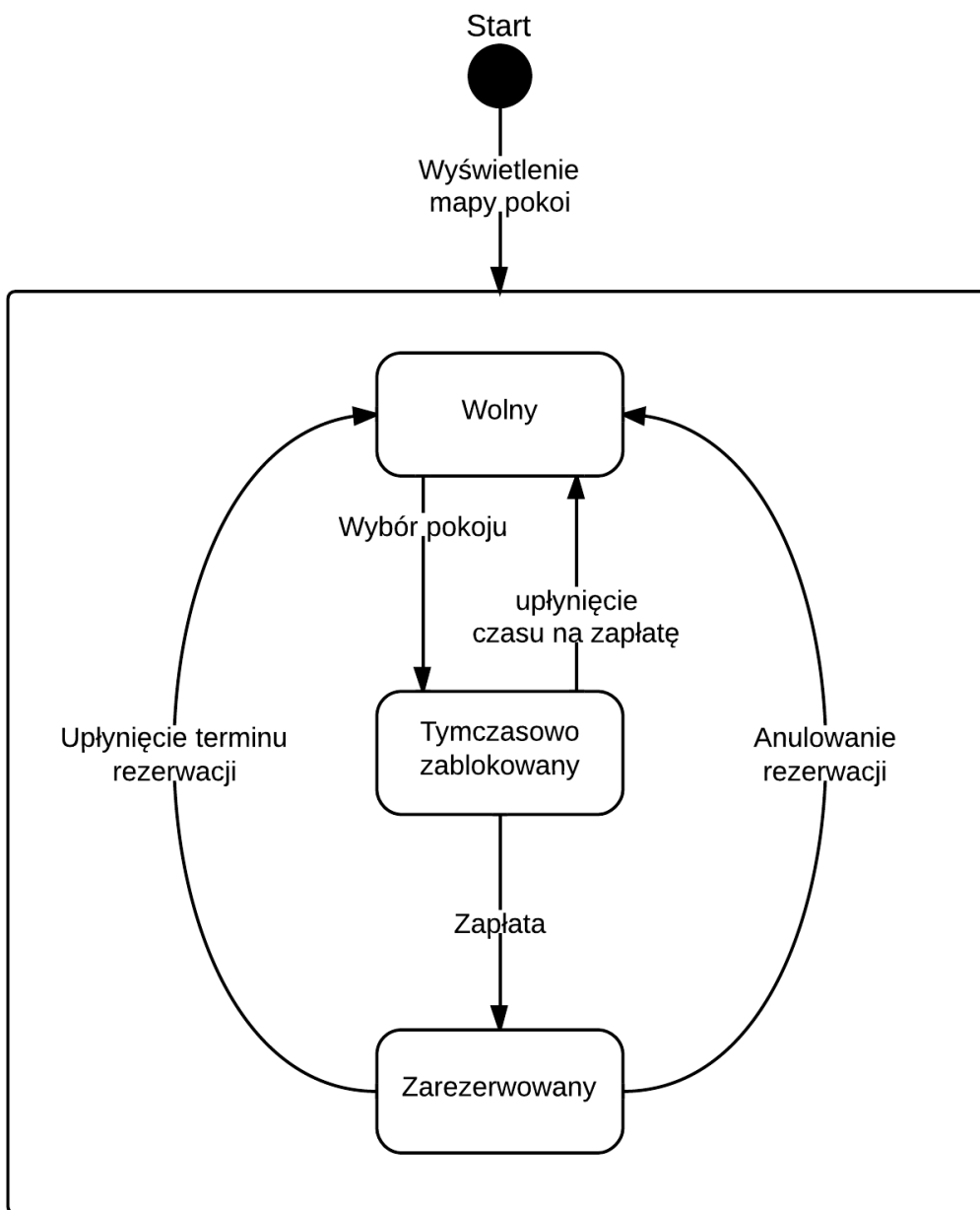
HotelGuest - Klasa odpowiedzialna za przeglądanie dostępnych pokoi, bez możliwości przejścia do etapu rezerwacji.

Cleaner - Klasa odpowiedzialna za zmianę stanu pokoju z „do posprzątania” na „posprzątany”.

Administrator - Klasa odpowiedzialna za edycję kont użytkowników, edycję pokoi.

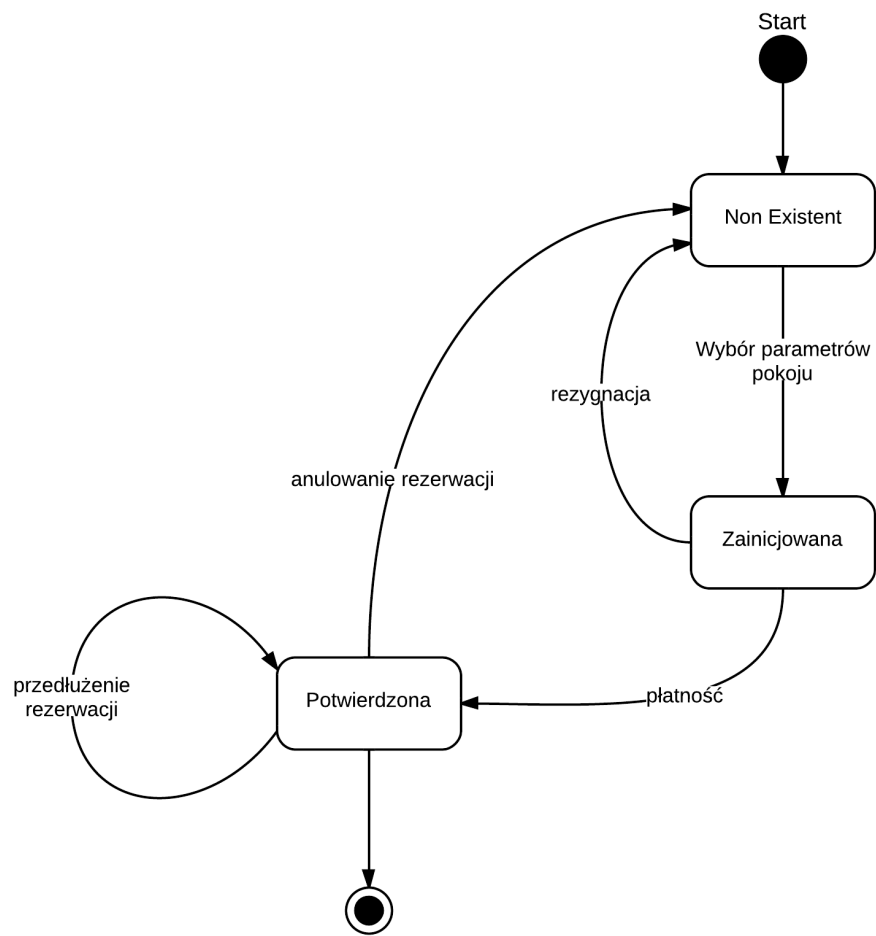
HotelManager - Klasa odpowiedzialna za generowanie raportów oraz możliwość zarządzania pokojami.

4.4.3 Diagram stanów - Pokój



Rysunek 4.9: Diagram stanów - Pokój

4.4.4 Diagram stanów - Rezerwacja

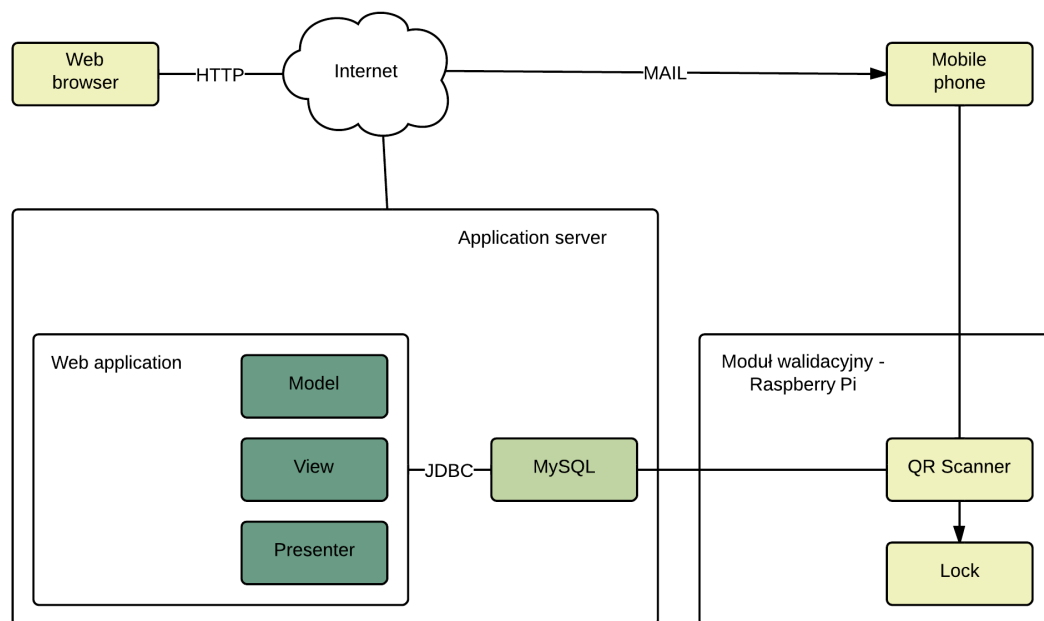


Rysunek 4.10: Diagram stanów - Rezerwacja

Rozdział 5

Projektowanie

5.1 Architektura aplikacji



Rysunek 5.1: Architektura aplikacji

Application server - serwer aplikacji

Web application - komponent aplikacji internetowej zbudowany na bazie wzorca MVP

MySQL - baza danych połączona z aplikacją za pomocą interfejsu JDBC

Mobile phone - urządzenie przenośne z możliwością wyświetlania kodu na ekranie. Służy do wyświetlenia obrazka kodu kamerze modułu walidacyjnego

Moduł walidacyjny Raspberry Pi - komputer Raspberry Pi z kamerą służący jako moduł do walidacji kodów i otwierania zamka klucza

5.2 Dekompozycja systemu na podsystemy

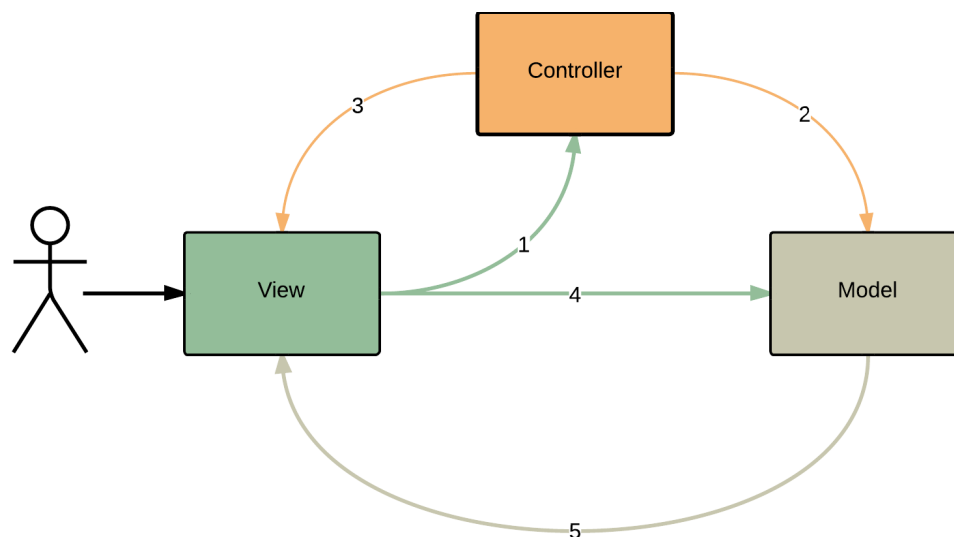
5.2.1 Wykorzystane wzorce projektowe

Początkowo próbowano umieścić projekt w ramy wzorca MVC. Jak się później okazało przy próbie stworzenia diagramu zależności podsystemów, aplikacja jest klasycznym MVP.

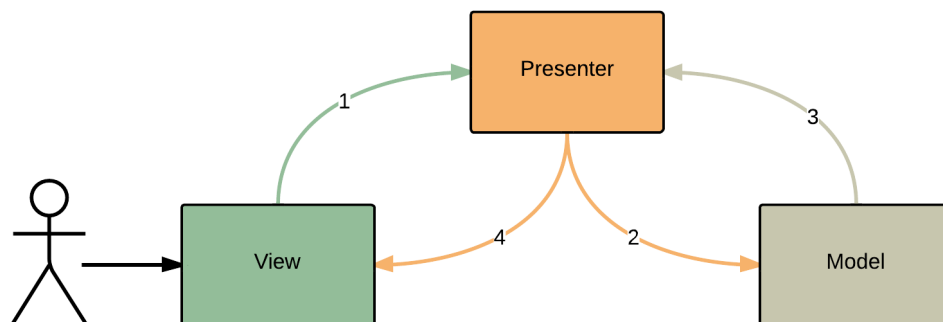
Wzorzec ten był rekomendowany przez wykładowcę Jakuba Neumanna jak i przez twórców Vaadina czym utwierdziło w przekonaniu, że nowe, niesprawdzone nie zawsze musi być złe.

MVP, czyli z ang. Model-View-Presenter (Model-Widok-Prezenter) jest pochodną wzorca MVC. Modyfikacja polega na tym, że kontroler z MVC jest prezenterem. To oznacza, że wszelkie wyniki logiki biznesowej aplikacji przesyłane są właśnie z tego prezentera, a nie jak w klasycznym wzorcu MVC - z modelu.

Porównanie idei MVC i MVP:



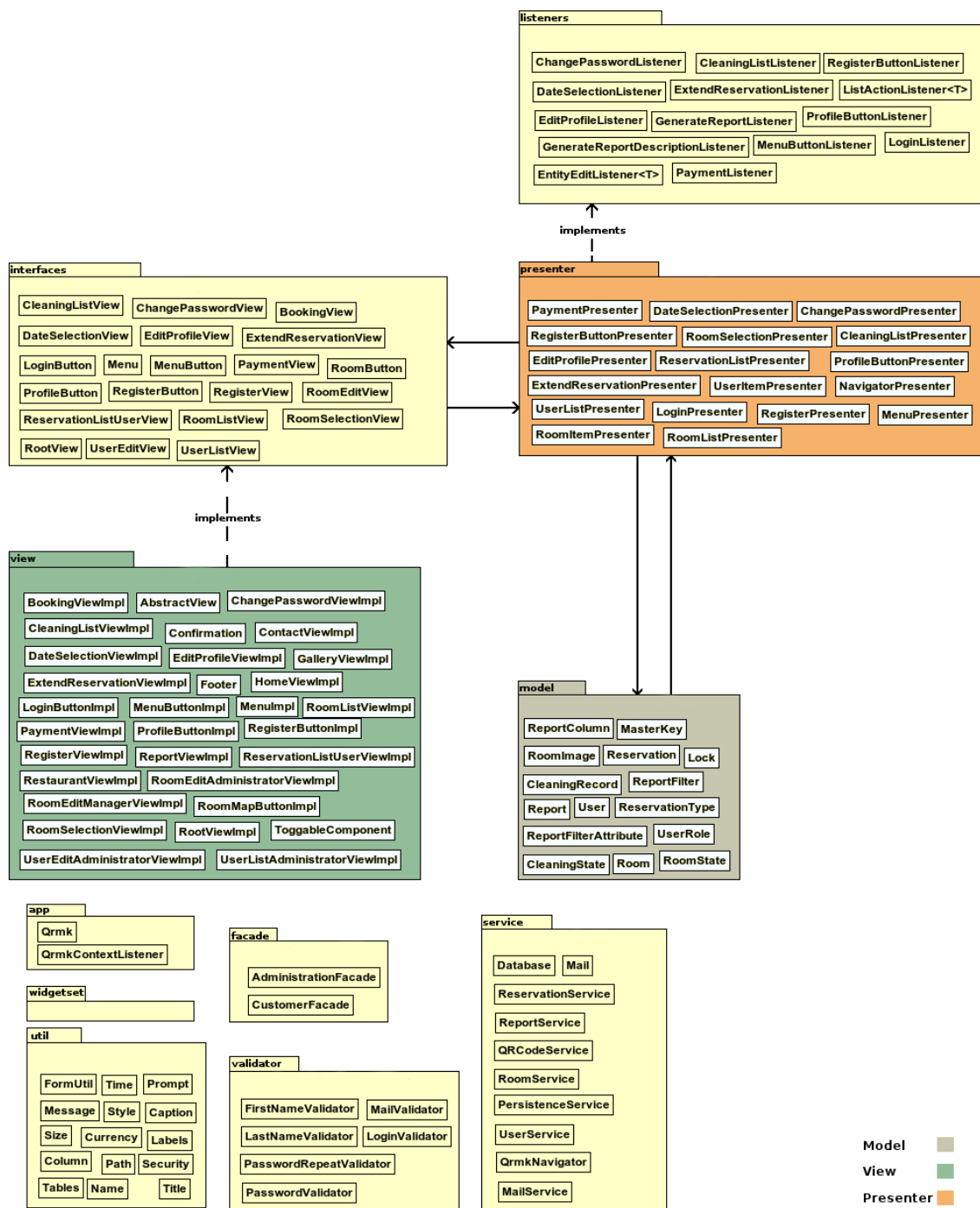
Rysunek 5.2: Wzorzec MVC



Rysunek 5.3: Wzorzec MVP

5.2.2 Diagram pakietów

Poniższy diagram pokazuje w jaki sposób wykorzystany przez nas wzorzec MVP (Model, View, Presenter) przekłada się na strukturę pakietów w systemie QR Magic Key.



Rysunek 5.4: Diagram pakietów

5.2.3 Opis pakietów

app - pakiet zawierający główną klasę aplikacji wraz z interfejsem.

facade - pakiet zawierający klasy służące do ujednolicenia dostępu do systemu poprzez wystawienie uproszczonego interfejsu.

model - pakiet zawierający klasy mapowane na obiekty bazy danych.

presenter - pakiet zawierający klasy prezenterów. Mają na celu pośredniczenie pomiędzy widokiem a modelem.

view - pakiet zawierający klasy tworzące interfejs użytkownika z wykorzystaniem frameworku Vaadin. Zgodnie ze wzorcem MVP każdy widok posiada swój interfejs.

util - klasy zawierające statyczne metody pomocnicze oraz stałe globalne.

5.3 Projekt struktury i algorytmy podsystemów

5.3.1 Diagram klas

Przy projektowaniu systemu wykorzystany został wzorec projektowy MVP. Jest to wzorec bardzo często stosowany przy tworzeniu interfejsów użytkownika. We wzorcu MVP Prezenter przyjmuje rolę pośrednika pomiędzy widokami a modelem. Reagując na akcje użytkownika modyfikuje i pobiera potrzebne dane z modelu logicznego aplikacji. Na podstawie otrzymanych danych jest w stanie obsłużyć widok dzięki dostępowi do jego interfejsu.

Opis wybranych klas znajduje się w punkcie 5.3.2

Oprócz pakietów wynikających z zastosowania wzorca MVP wydzielone zostały także pakiety:

facade - pakiet tworzący uproszczony interfejs do komunikacji pomiędzy prezenterami a modelem logicznym aplikacji

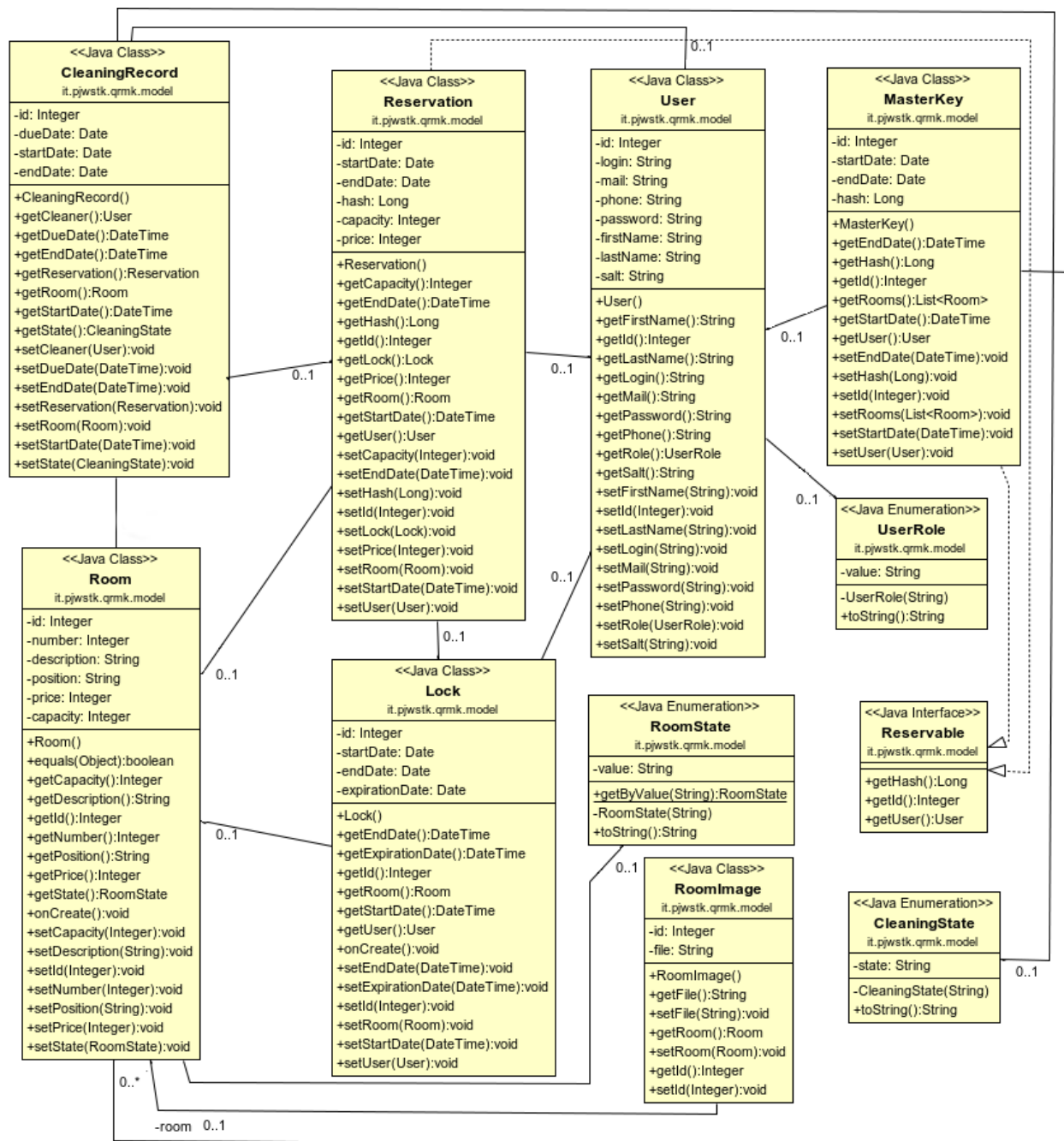
service - znajdują się w nim klasy zawierające metody do komunikacji z bazą danych oraz metody do generowania kodów QR i wysyłania maili do użytkowników

util - pakiet zawierający zbiór klas pomocniczych wykorzystywanych w całej aplikacji

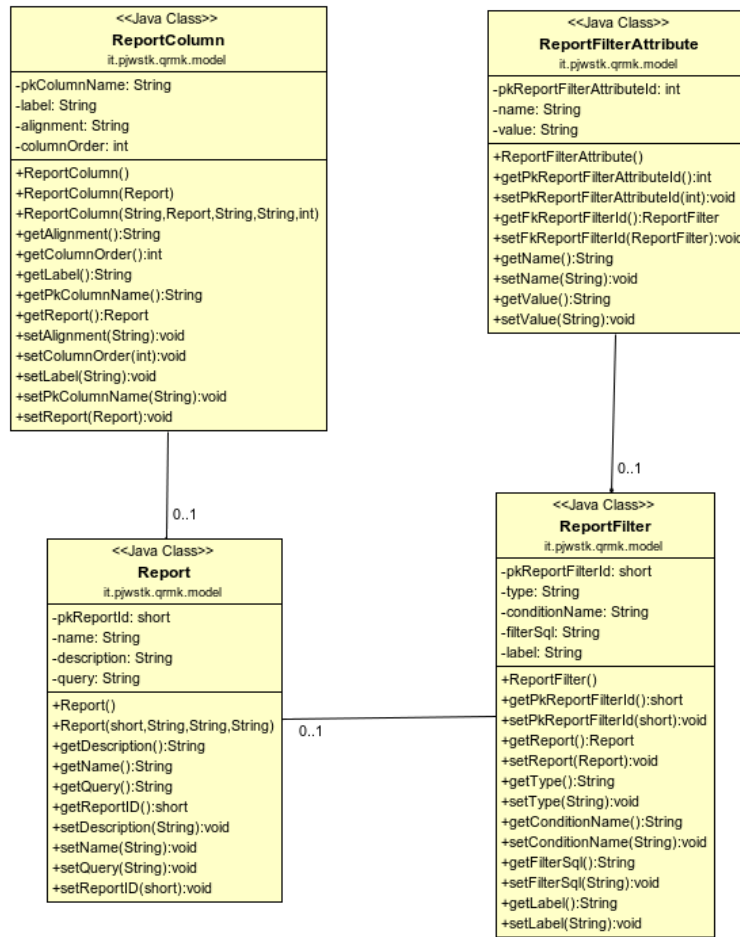
Poniżej zaprezentowane zostały uproszczone diagramy klas rozdzielone na poszczególne pakiety. Niektóre szczegóły zostały pominięte ze względu na zachowanie przejrzystości diagramów. Każda klasa widoku zgodnie z założeniami wzorca MVP posiada swój interfejs służący do komunikacji z prezenterem odpowiedzialnym za dany widok. Dodatkowo diagram reprezentujący pakiet zawierający klasy widoków został rozdzielony na dwie części

- widoki do których użytkownik ma dostęp poprzez wprowadzenie adresu w przeglądarce
- widoki będące podwidokami wyżej wymienionych lub komponenty składowe widoków

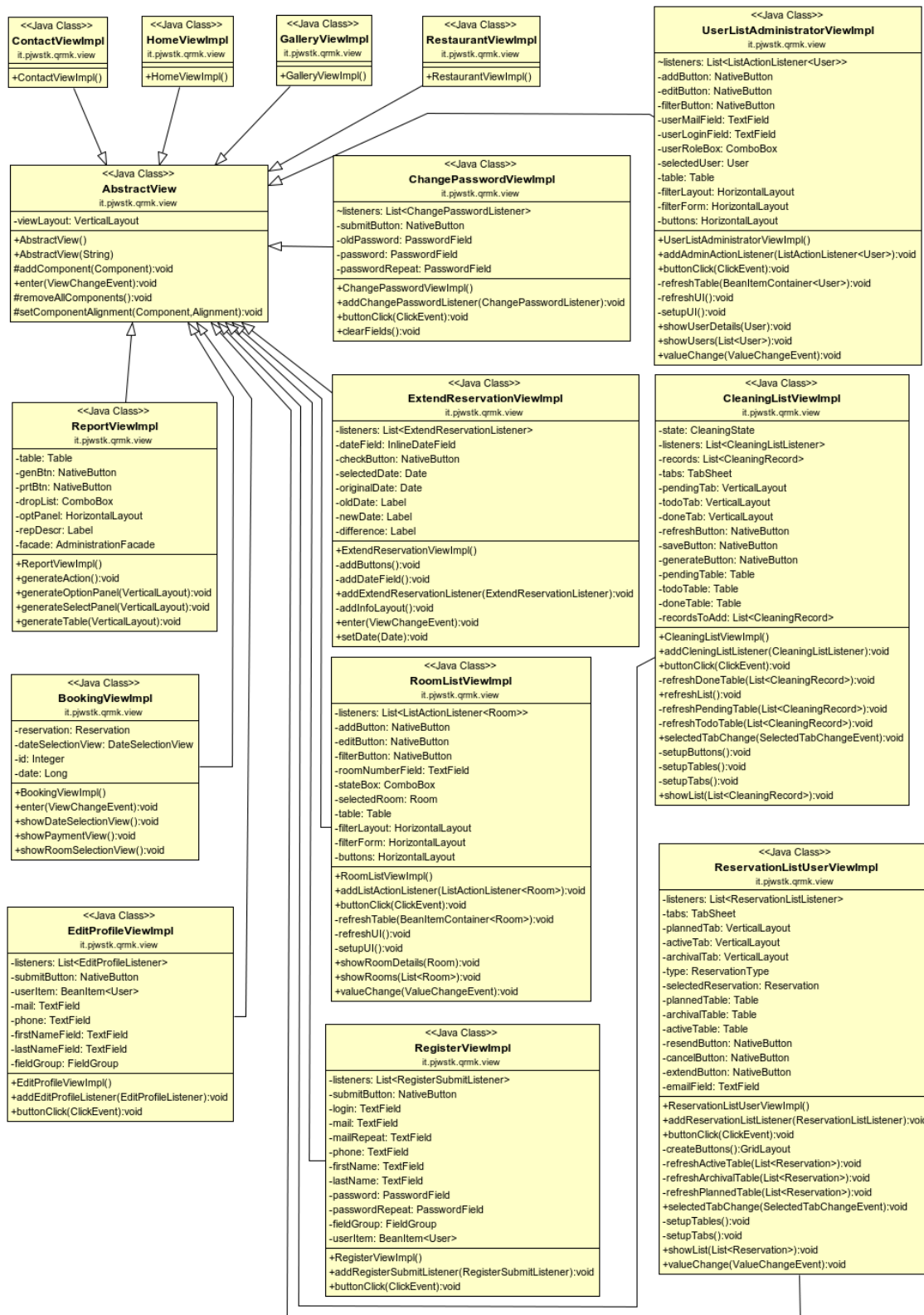
Przy tworzeniu diagramu klas został użyty plugin dla środowiska Eclipse. Ze względu na to, iż atrybuty zostały dodane automatycznie, na diagramie znalazły się ID dla klas z pakietu it.pjwstk.qrmk.model., które przy tego typu diagramu powinny zostać pominięte.



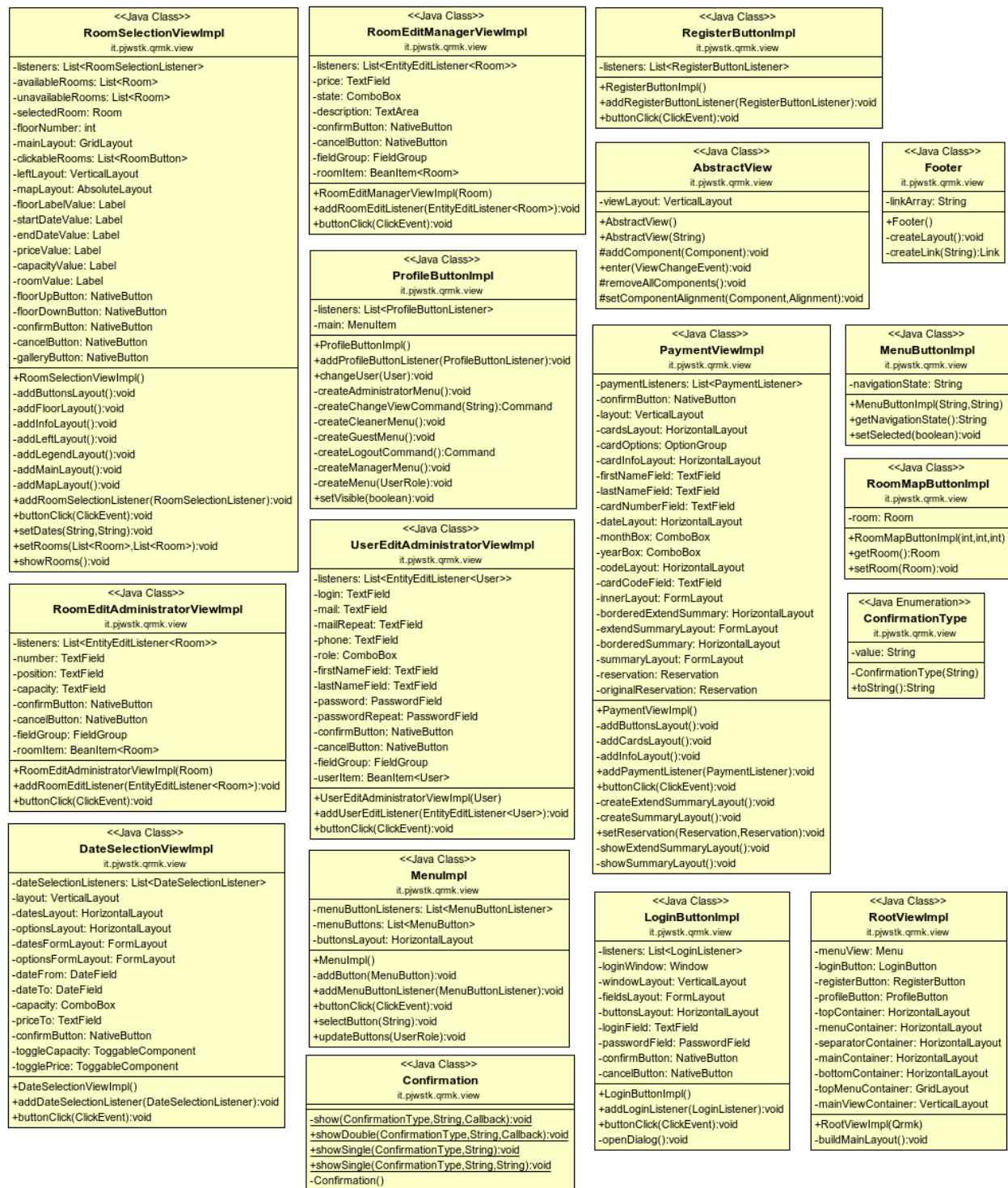
Rysunek 5.5: Diagram klas pakietu it.pjwstk.qrmk.model



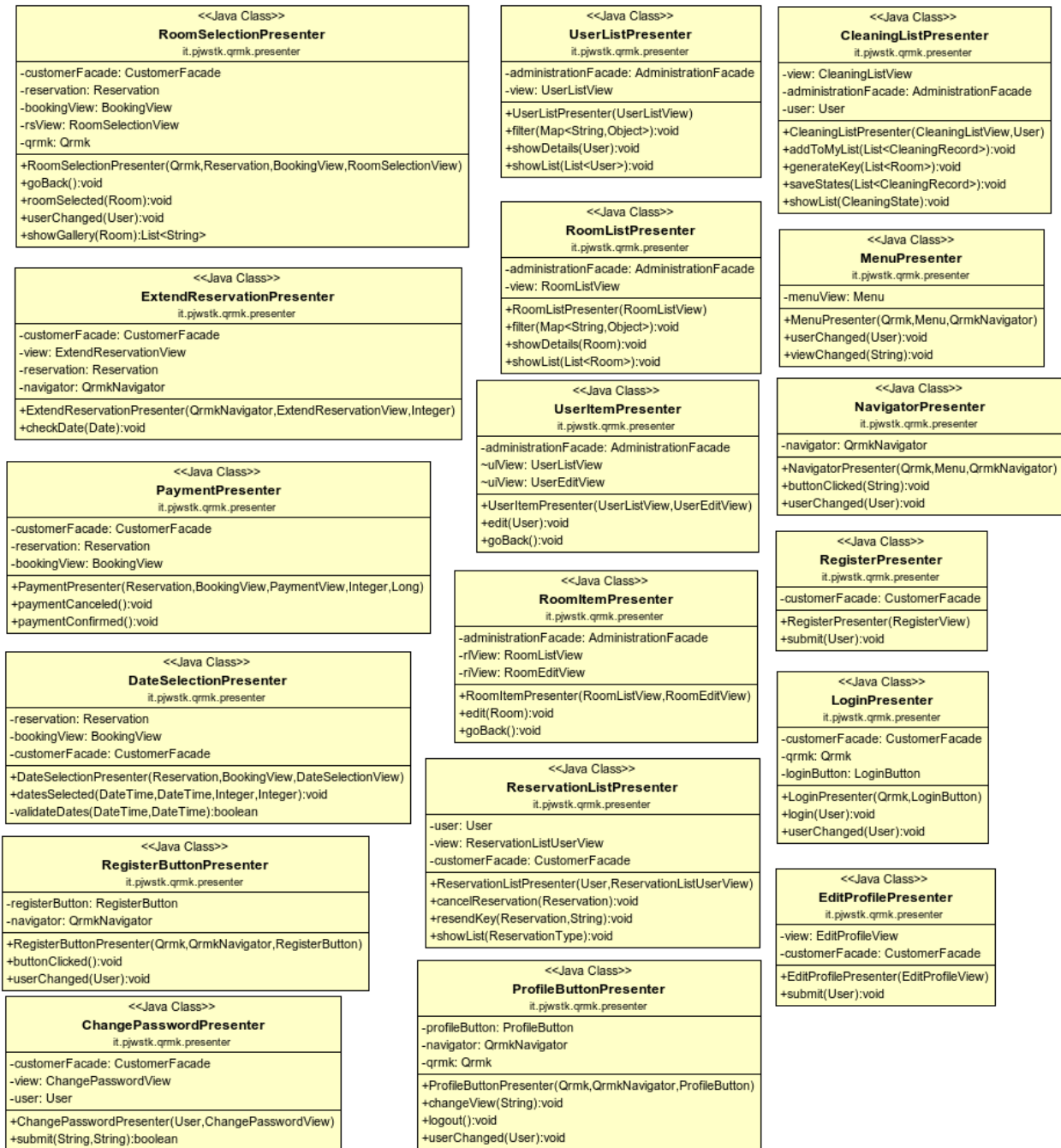
Rysunek 5.6: Diagram klas pakietu it.pjwstk.qrmk.model c.d.



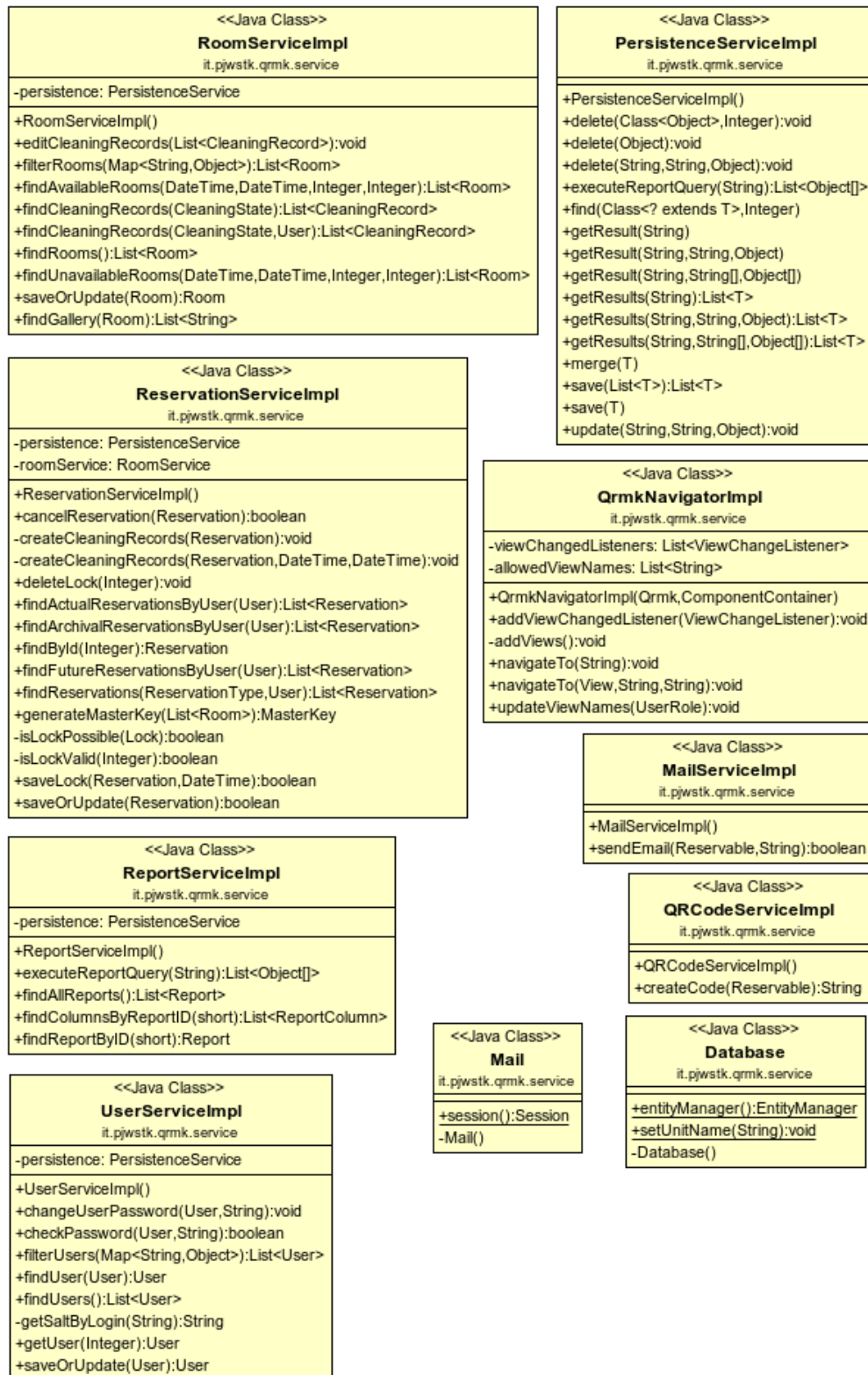
Rysunek 5.7: Diagram klas pakietu it.pjwstk.qrmk.view



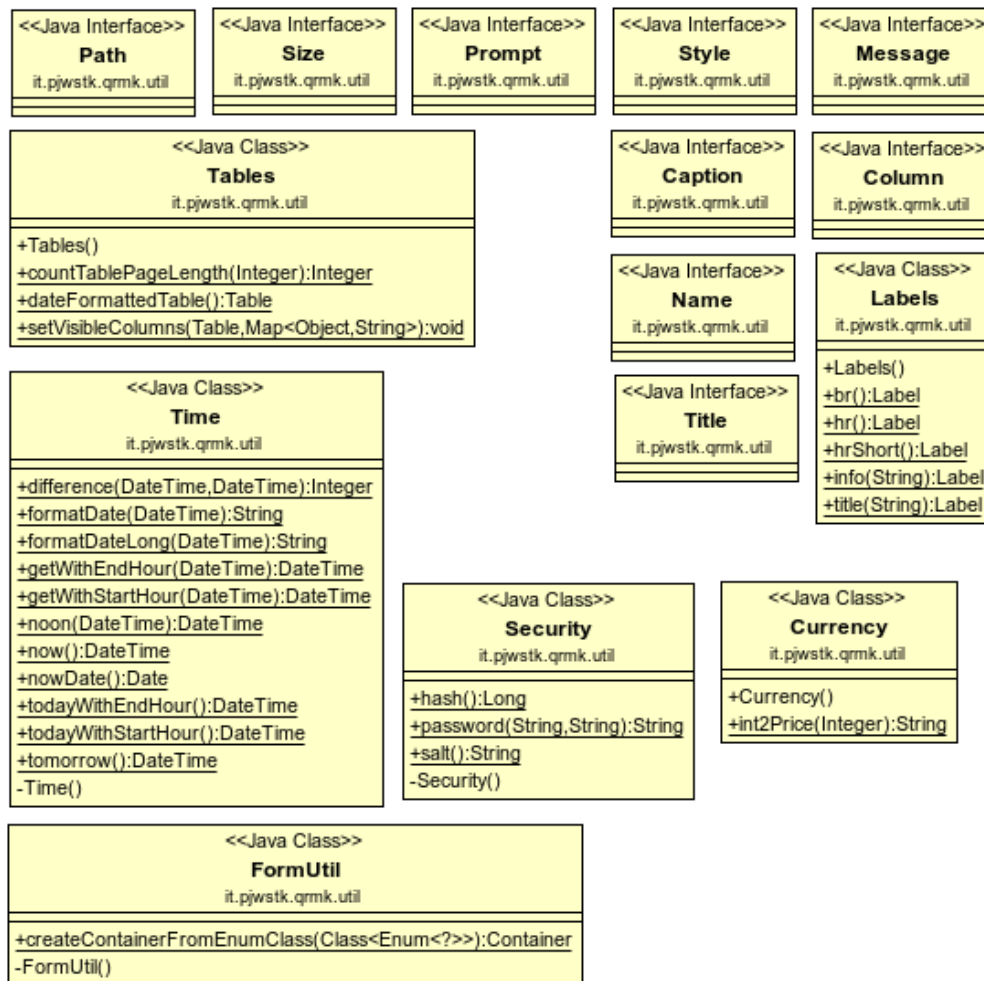
Rysunek 5.8: Diagram klas pakietu it.pjwstk.qrmk.view c.d.



Rysunek 5.9: Diagram klas pakietu it.pjwstk.qrmk.presenter



Rysunek 5.10: Diagram klas pakietu it.pjwstk.qrmk.service



Rysunek 5.11: Diagram klas pakietu it.pjwstk.qrmk.util

5.3.2 Opis wybranych klas

AbstractView - Klasa rozszerzana przez wszystkie widoki do których użytkownik ma dostęp poprzez wpisanie adresu w przeglądarce. Klasa ta implementuje interfejs View pochodzący z frameworku Vaadin. Interfejs ten stanowi kontrakt niezbędny do użycia danego widoku przez klasę nawigatora odpowiadające za zmianę widoków aplikacji. Widoki rozszerzające tę klasę rejestrowane są w klasie QrmkNavigatorImpl, aby następnie mogły być tworzone ich instancje w momencie zmiany widoku przez użytkownika.

Atrybuty - Klasa AbstractView posiada atrybuty oraz metody typowe dla komponentów dostarczanych przez framework Vaadin.

Metody :

- addComponent() - dodaje nowy komponent do widoku

- `enter()` - metoda uruchamiana w momencie przejścia na dany widok
- `removeAllComponents()` - usuwa wszystkie komponenty z widoku
- `setComponentAlignment()` - ustawia pozycję komponentu w widoku

RoomSelectionViewImpl - klasa zawierająca implementację widoku służącego do wyboru pokoju przez użytkownika. Interfejs tej klasy tak jak w przypadku innych klas widoków przekazywany jest do odpowiedniego prezentera. Prezenter uzyskuje w ten sposób dostęp do metod potrzebnych, aby obsłużyć dany widok, a także rejestruje siebie jako słuchacza, który będzie informowany o akcjach użytkownika. Interfejs widoku posiada między innymi metodę służącą do wyświetlenia listy pokoi przekazanych z prezentera jako mapy pokoi oraz metody służące do prezentacji dat wybranych w poprzednim kroku procesu rezerwacji pokoju.

Atrybuty :

- `listeners` - lista słuchaczy zarejestrowanych w widoku
- `availableRooms` - lista dostępnych pokoi wyświetlanych na mapie pokoi
- `unavailableRooms` - lista niedostępnych pokoi wyświetlanych na mapie pokoi
- `selectedRoom` - aktualnie zaznaczony pokój
- `floorNumber` - aktualnie wybrane piętro
- `mainLayout`, `leftLayout`, `mapLayout` - layouty używane w widoku
- `floorLabelValue`, `startDateValue`, `endDateValue`, `priceValue`, `capacityValue`, `roomValue` - wartości poszczególnych parametrów rezerwacji
- `floorUpButton`, `floorDownButton`
- `confirmButton`, `cancelButton`, `galleryButton` - przyciski używane w widoku

Metody :

- `setDates()` - ustawia informacje o datach rezerwacji
- `setRooms()` - ustawia listę pokoi na podstawie której utworzone zostanie mapa pokoi
- `showRooms()` - wyświetla mapę pokoi
- `addRoomSelectionListener()` - rejestruje słuchacza w widoku

RoomSelectionPresenter - klasa prezentera będącego pośrednikiem pomiędzy widokiem wyboru pokoju przez użytkownika a modelem logicznym aplikacji. W momencie tworzenia obiektu tej klasy do konstruktora przekazany zostaje interfejs odpowiedniego widoku tak aby prezenter miał dostęp do metod niezbędnych do obsługi widoku. Dodatkowo prezenter rejestruje siebie jako słuchacza dzięki implementacji interfejsu typu listener. W ten sposób prezenter jest informowany interakcji użytkownika z widokiem wyboru pokoi. Przykładowo w momencie akceptacji wybranego pokoju poprzez kliknięcie w odpowiedni przycisk prezenter zostaje o tym poinformowany. Odpytuje wówczas odpowiedni model korzystając z fasady o dodatkowe dane potrzebne do przeprowadzenia procesu wyboru pokoju. Po

otrzymaniu informacji zwrotnych z modelu prezwter może wywołać odpowiednią metodę widoku mówiąc w ten sposób jak widok ma się zachować. Taki schemat postępowania wynika z zastosowanego wzorca projektowego, którego prezwter jest nieodzowną częścią.

Atrybuty :

- customerFacade - fasada obejmująca funkcjonalność dostępną dla klientów hotelu
- reservation - obiekt rezerwacji zawierający informacje na temat wybranych przez użytkownika parametrów

Metody :

- goBack() - powrót do poprzedniego widoku
- roomSelected() - pokój wybrany przez użytkownika
- userChanged() - metoda wywoływana w momencie zalogowania lub wylogowania użytkownika
- showGallery() - metoda służąca do zaprezentowania listy pokoi na mapie

User - typowa klasa reprezentująca encje w bazie danych stanowiąca model w aplikacji. Posiada atrybuty potrzebne do zidentyfikowania użytkownika systemu takie jak login, hasło, dane osobowe. Wykorzystywana jest w relacjach z innymi klasami modelowymi, które do istnienia wymagają relacji z użytkownikiem.

Atrybuty :

- login - login użytkownika
- mail - adres email użytkownika
- phone - telefon kontaktowy
- role - uprawnienia użytkownika
- password - hasło służące do zalogowania się w systemie
- firstName - imię użytkownika
- lastName - nazwisko użytkownika

Metody : Klasa zawiera metody umożliwiające pobieranie i ustawianie wszystkich pól.

Time - klasa z pakietu it.pjwstk.qrmk.util wspomagające operacje wymagające wykorzystanie dat. Ponieważ standardowe klasy Javy umożliwiające operacje na datach posiadają wiele wad zastosowana została dodatkowa biblioteka JodaTime. Stworzona przez nas klasa wspomagająca korzysta z tej biblioteki.

Atrybuty : Klasa zawiera zestaw stałych wymaganych do poprawnego funkcjonowania.

Metody :

- `formatDate()` - metoda formatująca datę według schematu 'dd MMMM yyyy'
- `formatDateLong()` - metoda formatująca datę według schematu 'dd MMMM yyyy : H:mm'
- `getWithEndHour()` - zwraca datę z ustawioną godziną odpowiadającą końcowi doby hotelowej
- `getWithStartHour()` - zwraca datę z ustawioną godziną odpowiadającą początkowi doby hotelowej
- `noon()` - zwraca datę z godziną ustawioną na 12:00
- `now()` - zwraca obecną godzinę
- `nowDate()` - zwraca obecną godzinę w formacie Date
- `todayWithEndHour()` - zwraca datę obecnego dnia z ustawioną godziną odpowiadającą końcowi doby hotelowej
- `todayWithStartHour()` - zwraca datę obecnego dnia z ustawioną godziną odpowiadającą początkowi doby hotelowej
- `tomorrow()` - zwraca obecną datę plus 1 dzień

CustomerFacadeImpl - klasa stanowiąca fasadę uproszczającą korzystanie z klas serwisowych, które łączą się z bazą danych. Dzięki zastosowaniu fasady wywoływanie odpowiednich metod z poziomu prezenterów jest bardziej przejrzyste. W systemie powstały dwie fasady odpowiednio dla metod potrzebnych z poziomu niezalogowanego użytkownika lub gościa hotelowego oraz fasada administracyjna dla metod związanych z obsługą widoków dostępnych dla pracowników hotelu. Fasada obejmująca obsługę zwykłych użytkowników posiada między innymi metody niezbędne do obsługi rejestracji kont, rezerwacji pokoi, logowania itp.

Atrybuty :

- `roomService` - obiekt klasy zawierającej metody związane z funkcjonalnością dotyczącą pokoi
- `reservationService` - obiekt klasy zawierającej metody związane z funkcjonalnością dotyczącą rezerwacji
- `userService` - obiekt klasy zawierającej metody związane z funkcjonalnością dotyczącą użytkowników
- `mailService` - obiekt klasy zawierającej metody związane z funkcjonalnością dotyczącą wysyłania wiadomości email
- `codeService` - obiekt klasy zawierającej metody związane z funkcjonalnością dotyczącą generowania kodów QR

Metody :

- `cancelReservation()` - Anuluje rezerwacje poprzez usunięcie wszystkich rekordów sprzątnięcia do niej przypisanych oraz jej samej
- `changeUserPassword()` - Zmienia hasło użytkownika na podane

- `boolean checkPassword()` - Sprawdza poprawność hasła dla danego użytkownika
- `createCode()` - Generuje kod QR na podstawie danych zawartych w obiekcie Rezerwacji
- `deleteLock()` - Usuwa tymczasową blokadę pokoju zapobiegającą rezerwacji pokoju przez więcej niż jednego użytkownika
- `editUser()` - Edycja użytkownika w bazie danych
- `findActualReservationsByUser()` - Zwraca listę aktualnych rezerwacji przypisanych do danego Użytkownika
- `findArchivalReservationsByUser()` - Zwraca listę archiwalnych rezerwacji przypisanych do danego Użytkownika
- `findAvailableRooms()` - Zwraca listę dostępnych pokoi po przefiltrowaniu według podanych parametrów
- `findFutureReservationsByUser()` - Zwraca listę planowanych rezerwacji przypisanych do danego Użytkownika
- `findReservationById()` - Zwraca obiekt rezerwacji o podanym id
- `findReservations()` - Zwraca listę rezerwacji o podanym typie oraz przypisanych do danego użytkownika
- `findUnavailableRooms()` - Zwraca listę niedostępnych pokoi po przefiltrowaniu według podanych parametrów
- `getUser()` - Wyszukuje użytkownika po podanym loginie i hasle
- `saveLock()` - Zapisuje blokadę dla pokoju objętego rezerwacją
- `saveOrUpdateReservation(final Reservation reservation)` - Zapisuje lub edytuje istniejącą rezerwację w bazie danych. Operacja może się nie powieść w przypadku gdy blokada dotycząca danej rezerwacji jest już nieaktualna
- `saveOrUpdateUser()` - Zapisuje lub edytuje użytkownika
- `sendEmail()` - Wysyła email z kodem QR do użytkownika. Dane potrzebne do przeprowadzenia operacji znajdują się w obiekcie rezerwacji
- `findRoomGallery()` - Wyszukuje listę nazw plików do załadowania w galerii dla danego pokoju

RoomServiceImpl - klasa serwisowa odwołująca się do bazy danych. Klasy serwisowe zostały podzielone w sposób pozwalający rozdzielenie metod dotyczących poszczególnych klas reprezentujących encje w bazie danych. Klasa `RoomServiceImpl` dotyczy metod związanych z operacjami wykonywanymi na pokojach. Za przykład można podać dodawanie pokoi przez administratora, edycja stanu pokoi przez managera itp.

Atrybuty :

- `persistence` - obiekt klasy zawierającej metody do zapisywania i pobierania obiektów z bazy danych

Metody :

- editCleaningRecords - edycja rekordów sprzątania przypisanych do danego pokoju
- filterRooms - zwraca listę pokoi po przefiltrowaniu
- findAvailableRooms - wyszukuje dostępne pokoje
- findCleaningRecords - wyszukuje rekordy sprzątania
- findRooms - zwraca listę wszystkich pokoi w hotelu
- findUnavailableRooms - zwraca listę niedostępnych pokoi
- saveOrUpdate - zapisuje lub edytuje istniejący wpis dotyczący pokoju
- findGallery - zwraca listę zdjęć wykorzystywanych w galerii danego pokoju

MailServiceImpl - inny przykład klasy serwisowej, której celem jest wysyłanie mail z kodami QR do użytkowników systemu.

Atrybuty : Brak

Metody :

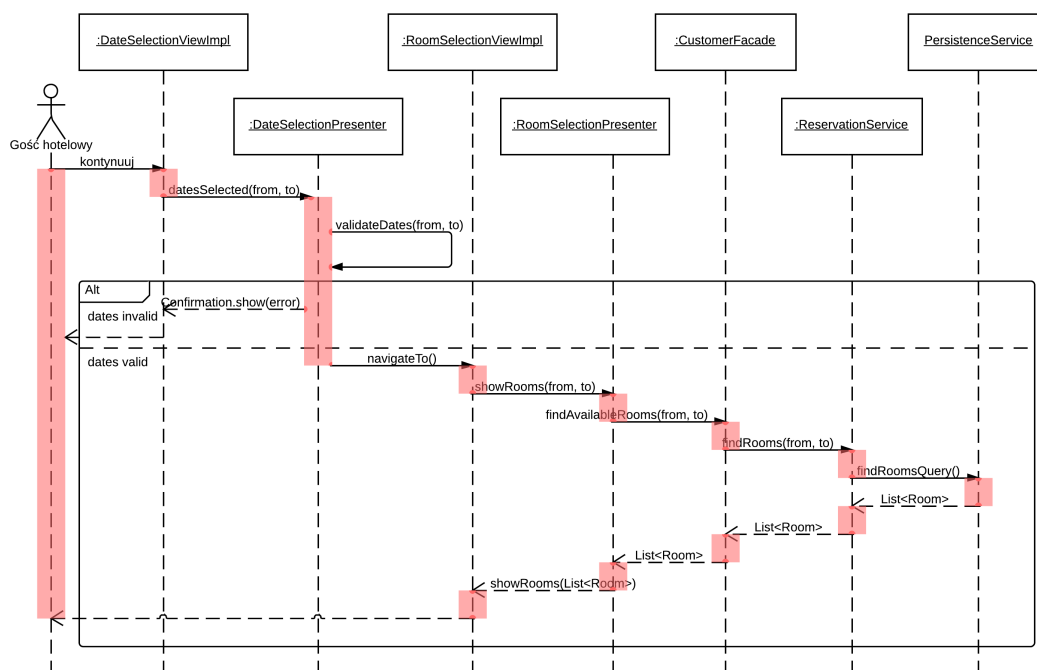
- sendEmail - wysyła wiadomość e-mail zawierającą wygenerowany kod QR do użytkownika

5.3.3 Wybrane diagramy sekwencji

Poniżej został przedstawiony proces rezerwacji pokoju. Z uwagi na złożoność całego procesu, na który składają się trzy osobne etapy, dla każdego etapu powstał oddzielny diagram sekwencji. Diagramy przedstawiają kolejne kroki jakie Użytkownik musi podjąć w celu zarezerwowania pokoju:

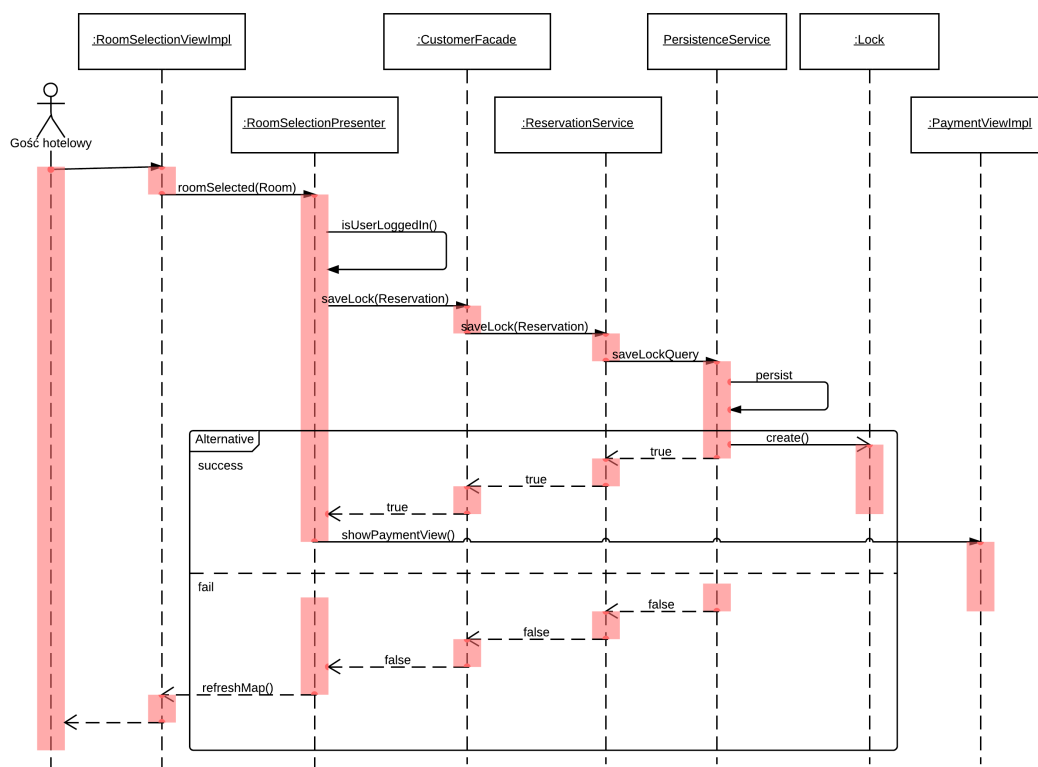
1. wybór daty rezerwacji
2. wybór pokoju
3. dokonanie płatności, która z uwagi na ograniczenia jest tylko symulacją prawdziwego procesu płatności. Pozwala jednak pokazać system tymczasowej blokady pokoju w celu uniemożliwienia jego rezerwacji przez więcej niż jednego użytkownika

Nazwa	Diagram sekwencji wyboru parametrów rezerwacji
Aktorzy	Gość hotelu
Opis	Gość hotelowy wybiera termin rezerwacji pokoju
Scenariusz	1. Użytkownik wybiera datę początkową oraz datę końcową dla rezerwacji 2. Użytkownik klika przycisk „Kontynuuj” i zostaje przekierowany do widoku wyboru pokoi
Stan końcowy	Użytkownik znajduje się w widoku wyboru pokoi, na którym wyświetlone są pokoje zgodnie z wybranymi przez niego parametrami



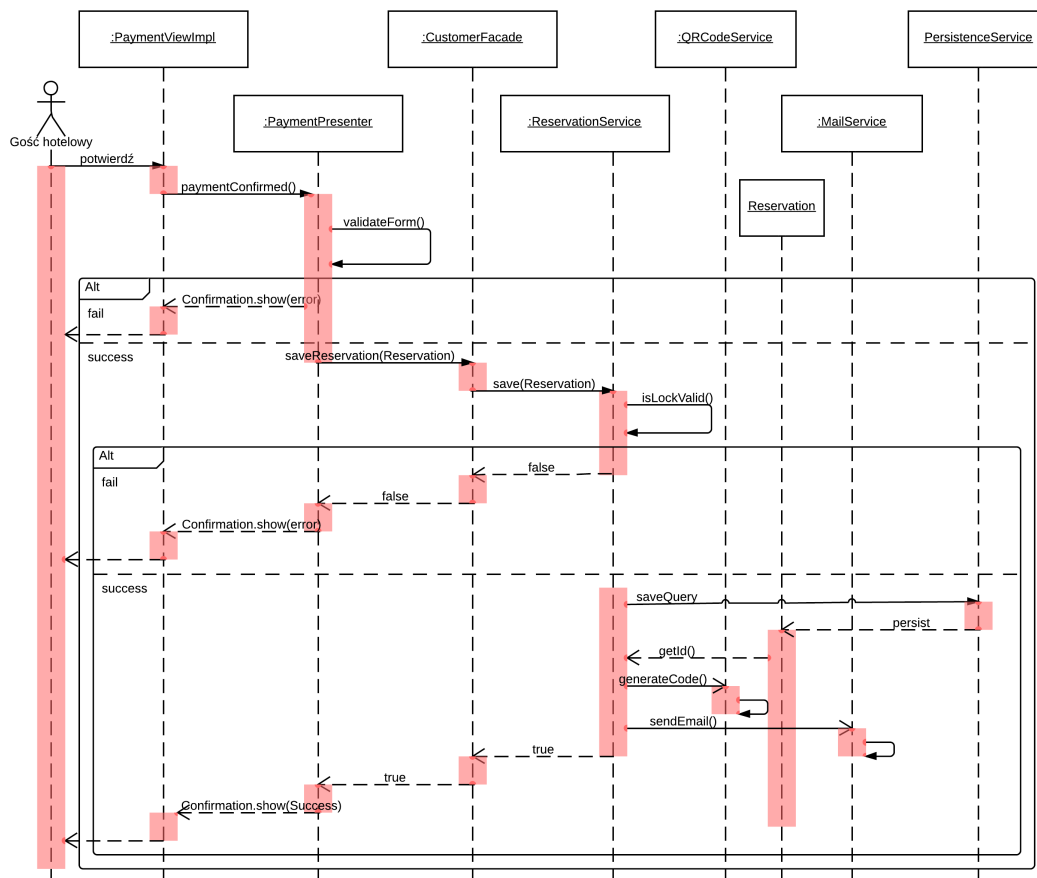
Rysunek 5.12: Diagram sekwencji wyboru parametrów rezerwacji

Nazwa	Diagram sekwencji wyboru pokoju
Aktorzy	Gość hotelu
Opis	Gość hotelowy wybiera pokój spośród dostępnych
Scenariusz	1. Użytkownik klika wabrany pokój na mapie 2. Użytkownik klika przycisk „Przejdź do płatności” 3. System tworzy blokadę dla wybranego pokoju tak, aby w trakcie dokonywania płatności pokój był niedostępny dla innych użytkowników 4. Użytkownik zostaje przekierowany do widoku płatności
Stan końcowy	Użytkownik znajduje się w widoku płatności



Rysunek 5.13: Diagram sekwencji wyboru pokoju

Nazwa	Diagram sekwencji płatności
Aktorzy	Gość hotelu
Opis	Gość hotelowy wypełnia i zatwierdza formularz płatności
Scenariusz	1. Użytkownik wypełnia dane w formularzu płatności 2. Użytkownik klika przycisk „potwierdź” 3. System generuje plik z kodem QR, a następnie wysyła go na adres podany przez użytkownika przy rejestracji 4. System wyświetla komunikat o powodzeniu operacji i następuje przekierowanie do widoku z listą rezerwacji planowanych użytkownika
Stan końcowy	Użytkownik otrzymał klucz uprawniający go do pobytu w hotelu w terminie podanym przy rezerwacji



Rysunek 5.14: Diagram sekwencji płatności

5.4 Decyzje projektowe

5.4.1 Języki programowania

Java Decyzję o wyborze języka Java podjęto ze względu na to, iż program studiów pozwolił na poznanie tego języka na poziomie umożliwiającym nam stworzenie zaawansowanej aplikacji. Decydując się na inny język musiano by poświęcić dodatkowy czas na głębsze zapoznanie się z nim.

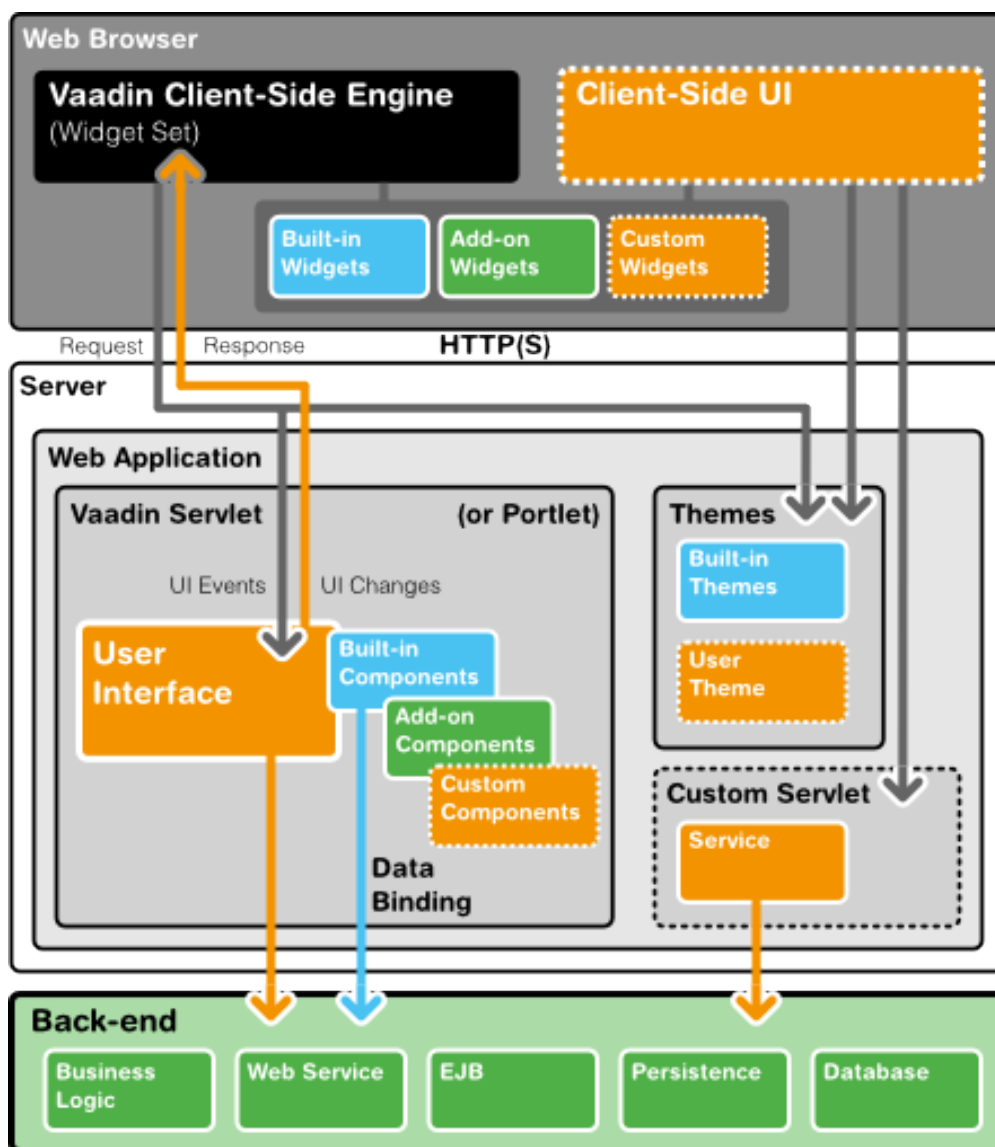
Perl Język perl został wybrany z powodu dostępności odpowiednich bibliotek obsługujących kody QR pod architekturę ARM. Perl posiada również małe wymagania odnośnie środowiska i zapewnia stabilną pracę. Nie bez znaczenia przy wyborze tego języka było praktyczne doświadczenie zdobyte przy pracy zawodowej i wcześniejszych projektach wykonywanych na uczelni.

5.4.2 Raspberry Pi

W charakterze czytnika kodów najlepiej sprawdził się komputer Raspberry Pi. Został on wybrany z powodu budowy w oparciu o architekturę ARM, niewielkiej ceny, możliwości uruchomienia systemu Linux wraz z bezproblemową obsługą języka Perl. Nie bez znaczenia było również posiadanie portów USB i portu GPIO za pomocą którego można sterować zewnętrznymi urządzeniami takimi jak diody LED lub elektryczny rygiel drzwi.

5.4.3 Framework Vaadin

Framework ten przedstawiony na przedmiocie Bogate Interfejsy Użytkownika. Okazał się łatwym w użyciu, a jednocześnie pozwalającym na osiągnięcie zadowalających efektów.



Rysunek 5.15: Schemat architektury Vaadin

5.4.4 Baza danych MySQL

MySQL jest bardzo szybkim, solidnym systemem zarządzania relacyjnymi bazami danych. Baza ta jest serwerem wielodostępnym i wielowątkowym. Wykorzystuje SQL (Structured Query Language), standardowy dla całego świata język zapytań bazy danych.

5.4.5 Serwer aplikacji

Skorzystano z popularnego serwera aplikacji Tomcat 7, gdyż zużywa małe ilości zasobów systemowych, jest prosty w administracji i wystarczający do framework'ów, które nie mają

potrzeby korzystania z pełnej zawartości serwera JavaEE.

5.5 Projekt bazy danych

Rysunek 5.16 przedstawia projekt bazy danych stworzony w oparciu o logiczny diagram klas. Pokazuje w jaki sposób klasy zostały zmapowane do bazy danych i jak współpracują ze sobą:

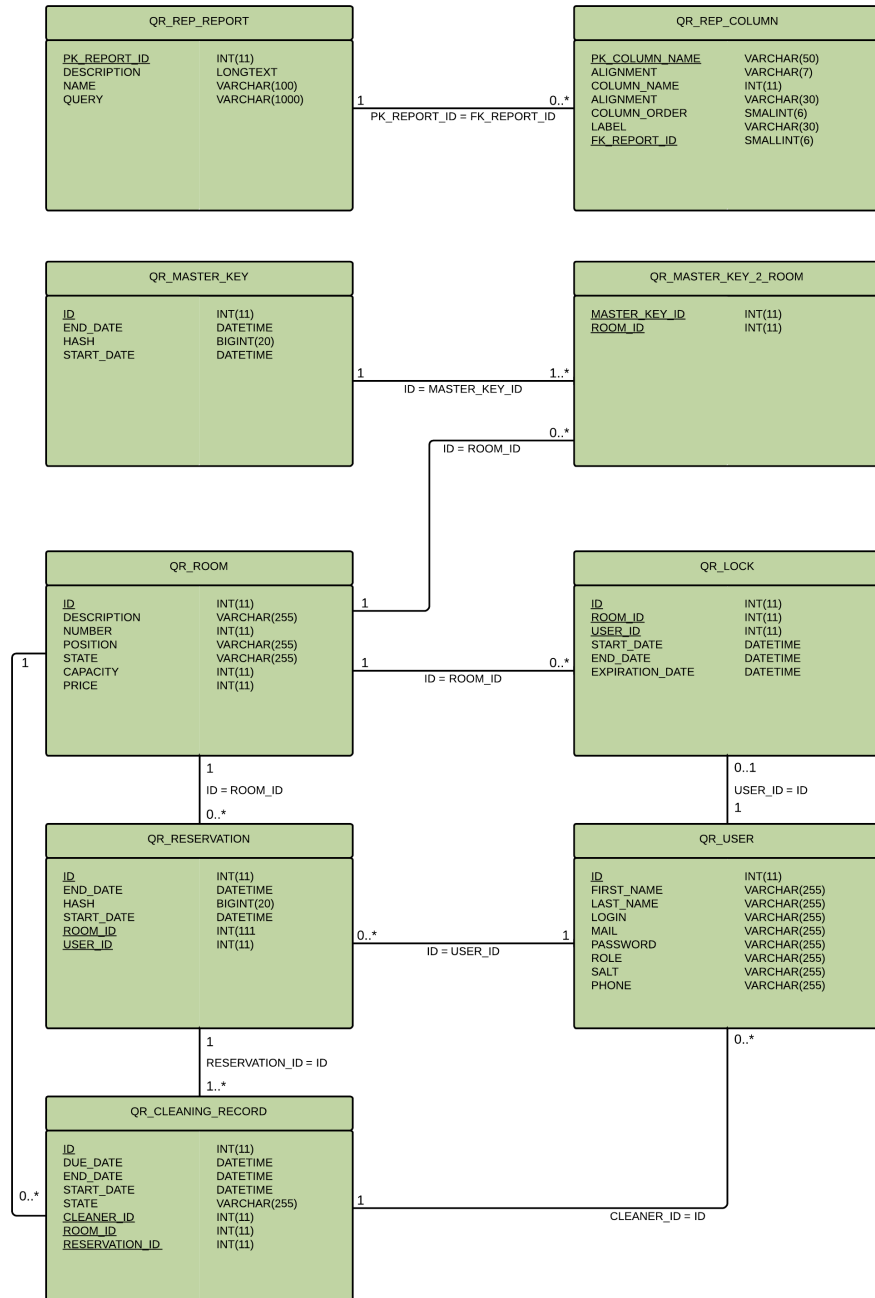
W momencie wybrania pokoju w aplikacji internetowej, zostaje założona tymczasowa blokada pokoju, zapisana w tabeli `QR_LOCK`, która powiązana jest z id pokoju z tabeli `QR_ROOM` oraz z id użytkownika z tabeli `QR_USER`.

W momencie potwierdzenia płatności, blokada zostaje usunięta z tabeli `QR_LOCK`, a rezerwacja zapisana w `QR_RESERVATION`. Tabela rezerwacji posiada w sobie daty rezerwacji oraz id użytkownika oraz id pokoju. Poza tym, w momencie rezerwacji, zostaje utworzony wpis do tabeli `QR_CLEANING_RECORD`, który posiada status `TO_CLEAN`, id pokoju oraz id rezerwacji. Codziennie taki pokój zostaje wypisany na liście w panelu osób sprzątających, tak aby można było taki pokój posprzątać.

W chwili gdy osoba sprzątajaca z tabeli `QR_USER` doda taki pokój do swojej listy, wpis w tabeli `QR_CLEANING_RECORD` przyjmuje stan `IN_PROGRESS` oraz zostaje mu przypisany id osoby sprzątajacej (użytkownika z tabeli `QR_USER` o roli `CLEANER`). Po posprzątaniu, osoba sprzątajaca, ręcznie zmienia w aplikacji stan faktyczny na posprzątany i wpis w `QR_CLEANING_RECORD` przyjmuje stan `CLEANED` oraz dodaje informacje o dacie posprzątania. Ta data jest później wykorzystana m.in. w generowaniu raportu do kontroli szybkości pracy osoby sprzątajacej (jest porównywana z datą rozpoczęcia sprzątania czyt. dodania przez `CLEANER`'a do swojej listy).

W bazie jest również tabela przechowująca dane o generowanych kluczach generalnych obsługujących wiele pokoi. W momencie gdy osoba sprzątajaca posiada na swojej liście wiele pokoi do posprzątania, może wygenerować sobie klucz generalny do tychże pokoi. Po tej operacji zostaje utworzony wpis w tabeli `QR_MASTER_KEY`, która jest powiązana z id pokoju z tabeli `QR_ROOM` oraz id użytkownika z tabeli `QR_USER`. Ze względu na to, iż tabele `QR_MASTER_KEY` oraz `QR_ROOM` są powiązane ze sobą w relacji wiele do wiele, pomiędzy nimi została dodana tabela asocjacyjna `QR_MASTER_KEY_2_ROOM`, która umożliwia taką implementację.

5.5.1 Diagram ERD



Rysunek 5.16: Diagram ERD

5.5.2 Specyfikacja tabel

QR_ROOM			
ID	PRIMARY KEY, NOT NULL, AUTO INCREMENT	INT(11)	Identyfikator
DESCRIPTION		VARCHAR(255)	Opis pokoju
NUMBER	NOT NULL	INT(11)	Numer pokoju
POSITION	NOT NULL	VARCHAR(255)	Pozycja pokoju na mapie
STATE	NOT NULL	VARCHAR(255)	Status pokoju. 'OUT_OF_ORDER' - wyłączony z użytku, 'READY' - gotowy do rezerwacji
CAPACITY	NOT NULL	INT(11)	Maksymalna liczba gości w pokoju
PRICE	NOT NULL	INT(11)	Cena za dobę

Tablica 5.1: Tabela Pokoi

QR_USER			
ID	PRIMARY KEY, NOT NULL, AUTO INCREMENT	INT(11)	Identyfikator
FIRST_NAME		VARCHAR(255)	Imię użytkownika
LAST_NAME		VARCHAR(255)	Nazwisko użytkownika
PASSWORD	NOT NULL	VARCHAR(255)	Hasło użytkownika
LOGIN	NOT NULL	VARCHAR(255)	Login użytkownika
MAIL	NOT NULL	VARCHAR(255)	Email użytkownika
ROLE	NOT NULL	VARCHAR(255)	Rodzaj konta
SALT	NOT NULL	VARCHAR(255)	Dane dodawane podczas szyfrowania. Szyfrowanie „salt” jest losowym modyfikatorem klucza lub tekstu jawnego wprowadzanym by ten sam tekst jawny szyfrowany wielokrotnie tym samym kluczem dawał zawsze inne kryptogramy
PHONE		VARCHAR(255)	Telefon kontaktowy użytkownika

Tablica 5.2: Tabela Użytkowników

QR_RESERVATION			
ID	PRIMARY KEY, NOT NULL, AUTO INCREMENT	INT(11)	Identyfikator
END_DATE	NOT NULL	DATETIME	Data zakończenia rezerwacji
START_DATE	NOT NULL	DATETIME	Data zakończenia rezerwacji
HASH	NOT NULL	BIGINT(20)	Losowy ciąg znaków umożliwiający walidację kodu przez czytnik
ROOM_ID	FOREIGN KEY, NOT NULL	INT(11)	Id pokoju, którego dotyczy rezerwacja
USER_ID	FOREIGN KEY, NOT NULL	INT(11)	Id użytkownika, którego dotyczy rezerwacja

Tablica 5.3: Tabela rezerwacji

QR_LOCK			
ID	PRIMARY KEY, NOT NULL, AUTO INCREMENT	INT(11)	Identyfikator
END_DATE	NOT NULL	DATETIME	Data zakończenia rezerwacji
START_DATE	NOT NULL	DATETIME	Data zakończenia rezerwacji
EXPIRATION_DATE	NOT NULL	DATETIME	Czas, po którym blokada pokoju traci ważność
ROOM_ID	FOREIGN KEY, NOT NULL	INT(11)	Id pokoju, którego dotyczy tymczasowa blokada
USER_ID	FOREIGN KEY, NOT NULL	INT(11)	Id użytkownika, do którego przypisana jest blokada

Tablica 5.4: Tabela tymczasowej blokady pokoju, uniemożliwiająca wybór tego samego pokoju przez więcej niż jednego użytkownika

QR_MASTER_KEY			
ID	PRIMARY KEY, NOT NULL, AUTO INCREMENT	INT(11)	Identyfikator
END_DATE	NOT NULL	DATETIME	Data zakończenia działania klucza
START_DATE	NOT NULL	DATETIME	Data rozpoczęcia działania klucza
HASH	NOT NULL	BIGINT(20)	Losowy ciąg znaków umożliwiający walidację kodu
USER_ID	FOREIGN KEY, NOT NULL	INT(11)	Id użytkownika, do którego przypisany jest klucz

Tablica 5.5: Tabela klucza generalnego

QR_MASTER_KEY 2 ROOM			
MASTER_KEY_ID	NOT NULL	INT(11)	Identyfikator klucza generalnego
ROOM_ID	NOT NULL	INT(11)	Identyfikator pokoju

Tablica 5.6: Tabela asocjacyjna łącząca QR_MASTER_KEY i QR_ROOM

QR_CLEANING_RECORD			
ID	PRIMARY KEY, NOT NULL, AUTO INCREMENT	INT(11)	Identyfikator
END_DATE	NOT NULL	DATETIME	Data zakończenia sprzątnia pokoju dodawana w momencie oznaczenia pokoju jak posprzątny przez osobę sprzątajacą
START_DATE	NOT NULL	DATETIME	Data rozpoczęcia sprzątnia pokoju dodawana w momencie przypisania do osoby sprzątajacej
DUE_DATE	NOT NULL	DATETIME	Termin w jakim pokój powinien zostać posprzątny
STATE	NOT NULL	VARCHAR(255)	Losowy ciąg znaków umożliwiający walidacje kodu
CLEANER_ID	FOREIGN KEY, NOT NULL	INT(11)	Id osoby sprzątajacej, do której przypisane jest posprzątanie pokoju objętego encją
ROOM_ID	FOREIGN KEY, NOT NULL	INT(11)	Id pokoju wymagającego posprzątania
RESERVATION_ID	FOREIGN KEY, NOT NULL	INT(11)	Id rezerwacji, której następstwem jest wymóg posprzątania danego pokoju. Umożliwia usunięcie wszystkich wymogów przypisanych do tego Id w przypadku anulowania rezerwacji przez użytkownika

Tablica 5.7: Encja reprezentująca wymóg posprzątania pokoju w danym dniu

QR_REP_REPORT			
PK_REPORT_ID	PRIMARY KEY, NOT NULL	SMALLINT(6)	Identyfikator raportu
DESCRIPTION		LONGTEXT	opis raportu
NAME	NOT NULL	VARCHAR(100)	Nazwa raportu
QUERY	NOT NULL	VARCHAR(1000)	Zapytanie SQL pobierające dane z bazy

Tablica 5.8: Tabela zawierająca informacje na temat raportów

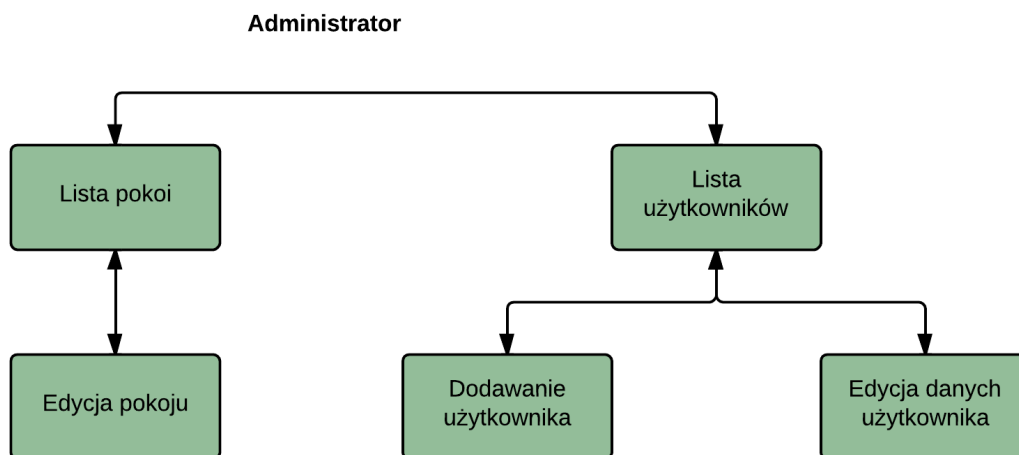
QR_REP_COLUMN			
PK_COLUMN_NAME	PRIMARY KEY, NOT NULL	VARCHAR(50)	Nazwa kolumny
ALIGNMENT	NOT NULL	VARCHAR(7)	Wyrównanie tekstu w komórce kolumny
COLUMN_ORDER	NOT NULL	INT(11)	Kolejność kolumny względem innych kolumn
LABEL	NOT NULL	VARCHAR(30)	Nagłówek kolumny
FK_REPORT_ID	FOREIGN KEY, NOT NULL	SMALLINT(6)	Identyfikator raportu, do którego przypisana jest kolumna

Tablica 5.9: Tabela zawierająca informacje na temat kolumn wykorzystywanych w raporcie

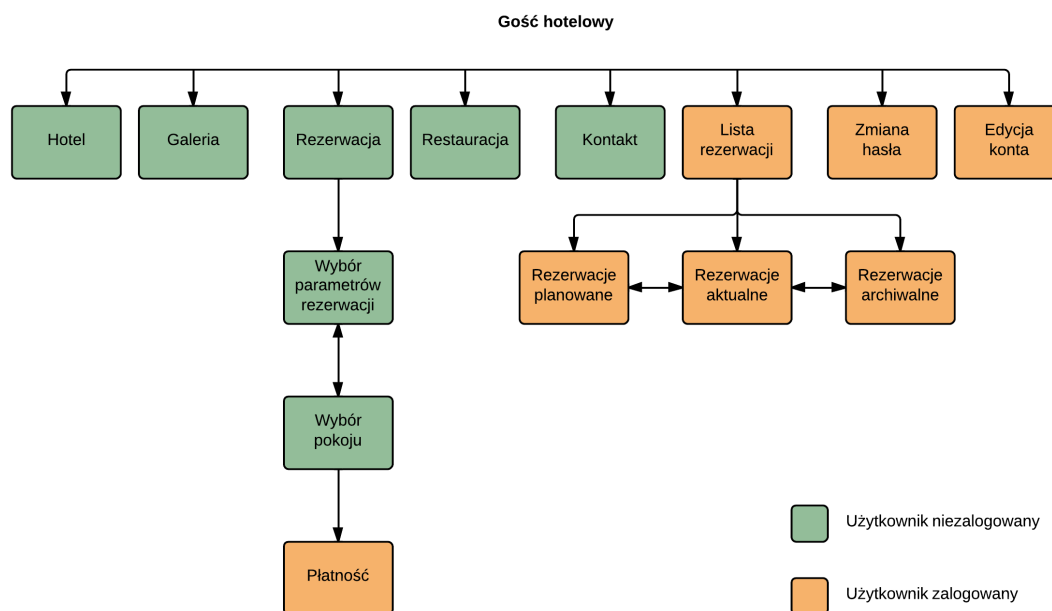
5.6 Projekt interfejsu użytkownika

W celu stworzenia interfejsu użytkownika zapoznaliśmy się z wyglądem wielu stron poświęconych tematyce hotelarskiej. Z naszych obserwacji wynikało, że najlepszym rozwiązaniem będzie zastosowanie poziomego menu głównego pozwalającego na przemieszczanie się pomiędzy panelami związanymi bezpośrednio z hotelem. Dodatkowo po zalogowaniu, użytkownik otrzymuje do dyspozycji menu umieszczone w przycisku znajdującym się w prawym górnym rogu ekranu. Menu udostępnia opcje odpowiednie dla uprawnień zalogowanego użytkownika. Przykładowo zalogowany gość hotelowy może przejść do widoku edycji konta lub listy swoich rezerwacji, a manager do widoków związanych z raportami i zarządzaniem parametrami pokoi. W przypadku zalogowanych pracowników hotelu dostęp do widoków przeznaczonych dla gości zostaje wyłączony w celu zachowania przejrzystości. Do stworzenia grafik użytych w aplikacji internetowej został wykorzystany darmowy program do tworzenia i obróbki grafiki - GIMP[12].

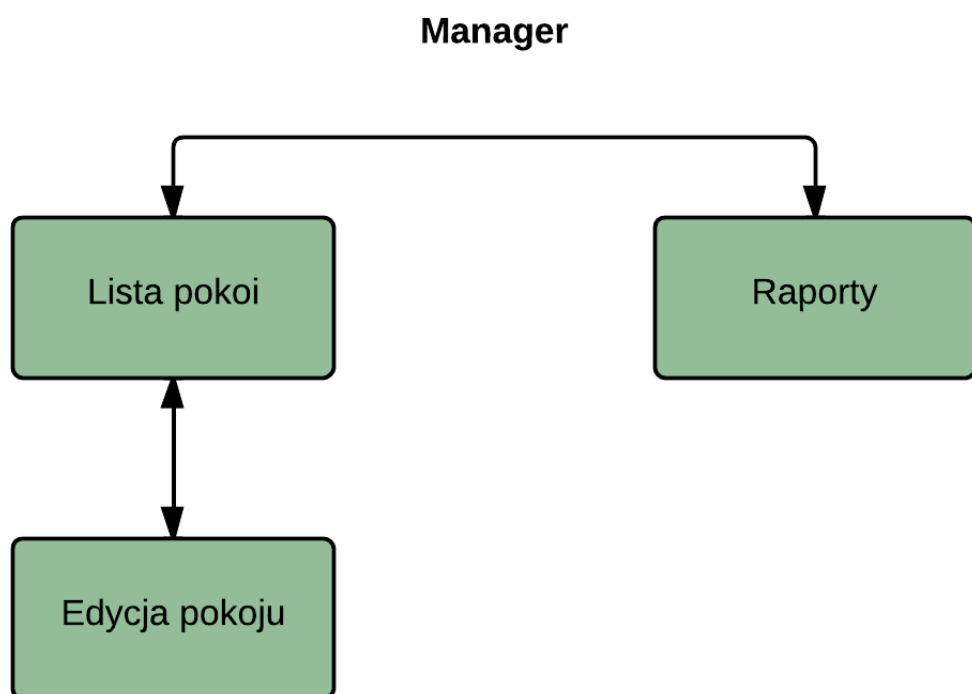
5.6.1 Diagramy nawigacji



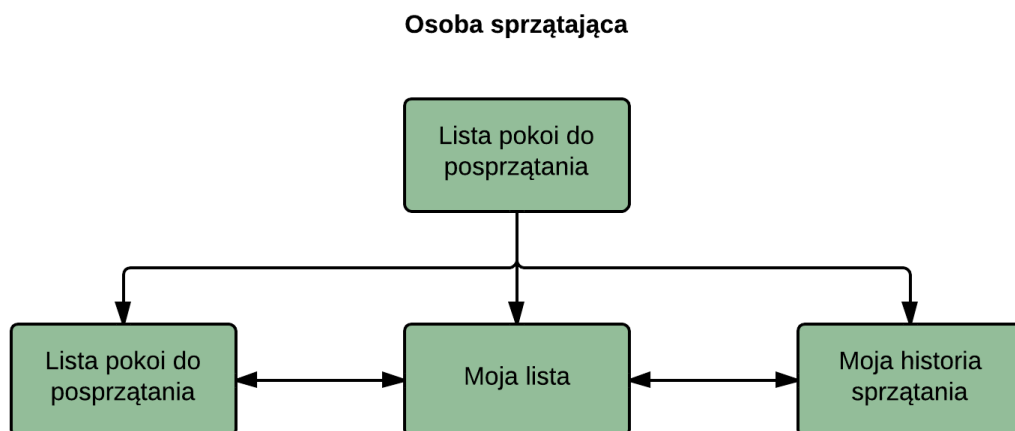
Rysunek 5.17: Diagram nawigacji dla Administratora



Rysunek 5.18: Diagram nawigacji dla Gościa hotelowego

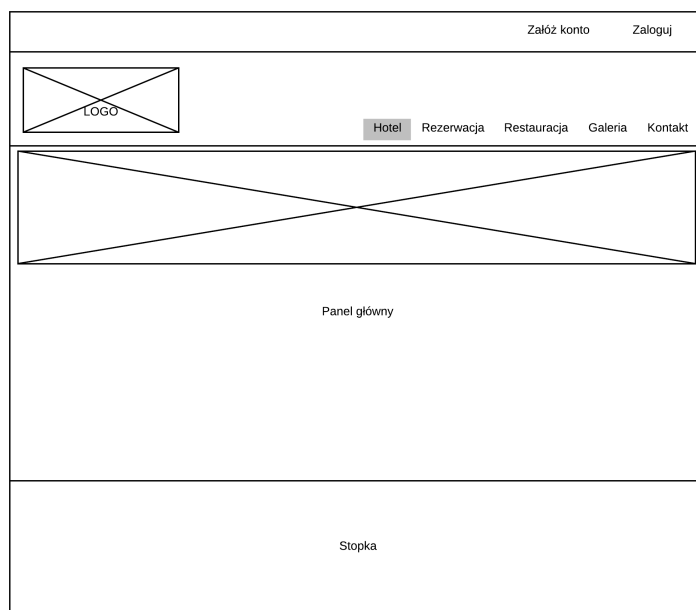


Rysunek 5.19: Diagram nawigacji dla Managera



Rysunek 5.20: Diagram nawigacji dla Osoby sprzątającej

5.6.2 Szkielet strony - rozmieszczenie paneli



Rysunek 5.21: Szkielet dla ekranu niezalogowanego użytkownika - ekran główny

Górny pasek opcji - zawiera przyciski do rejestracji i logowania. Po zalogowaniu, przycisk wzbogacony jest o listę opcji odpowiednich dla uprawnień zalogowanego użytkownika.

Menu główne - poziome menu zawierające przyciski służące do przemieszczania się pomiędzy panelami związanymi bezpośrednio z hotelem.

Panel główny - panel zawierający najważniejsze informacje dla danego widoku

Stopka - panel zawierający ważne odnośniki do stron internetowych oraz informację o prawach autorskich.

Witaj Adam

Edycja konta

Zmiana hasła

Rezerwacje

Wyloguj

Hotel

Rezerwacja

Galeria

Restauracja

Kontakt

LOGO

Informacje o:

-dacie przyjazdu

-dacie wyjazdu

-numrze pokoju

Mapa pokoi

Legenda mapy

Wybór piętra

Wróć

Przejdź do płatności

Stopka

Rysunek 5.22: Szkielet dla ekranu gościa - wybór pokoju

Witaj Adam

Raporty

Lista pokoi

Wyloguj

LOGO

Raporty

Wybór raportu

Opis wybranego typu raportu

Filtr A

Filtr B

Filtr C

Szukaj

Drukuj raport

Kolumna 1	Kolumna 2	Kolumna 3
Wartość 1	Wartość 2	Wartość 3
Wartość 4	Wartość 5	Wartość 6
Wartość 7	Wartość 8	Wartość 9
Wartość 10	Wartość 11	Wartość 12
Wartość 13	Wartość 14	Wartość 15
Wartość 16	Wartość 17	Wartość 18
Wartość 19	Wartość 20	Wartość 21

Stopka

Rysunek 5.23: Szkielet dla ekranu menedżera - raporty

66

Witaj Adam

Lista użytkowników
Lista pokoi
Wyloguj

LOGO

Lista pokoi

Numer pokoju

Status

▼

Szukaj

Dodaj

Edytuj

Kolumna 1	Kolumna 2	Kolumna 3
Wartość 1	Wartość 2	Wartość 3
Wartość 4	Wartość 5	Wartość 6
Wartość 7	Wartość 8	Wartość 9
Wartość 10	Wartość 11	Wartość 12
Wartość 13	Wartość 14	Wartość 15
Wartość 16	Wartość 17	Wartość 18
Wartość 19	Wartość 20	Wartość 21

Stopka

Rysunek 5.24: Szkielet dla ekranu managera i administratora - lista pokoi

Witaj Adam

Raporty
Lista pokoi
Wyloguj

LOGO

Edycja pokoju

Cena pokoju

Text

Stan

Wybór stanu

▼

Opis

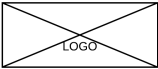
Text

Wróć

Zapisz

Stopka

Rysunek 5.25: Szkielet dla ekranu managera - edycja pokoju



Witaj Adam

Lista użytkowników
Lista pokoi
Wyloguj

Edycja pokoju

Numer

Text

Pozycja

Text (x, y, w, h)

Ilość osób

Text

Wróć

Zapisz

Stopka

Rysunek 5.26: Szkielet dla ekranu administratora - edycja pokoju



Witaj Adam

Edycja konta
Zmiana hasła
Rezerwacje
Wyloguj

Hotel

Rezerwacja

Galeria

Restauracja

Kontakt

Krok 1/3

Data przyjazdu

8/8/13

Data wyjazdu

15/8/13

Maksymalna ilość Gości

ilość gości

Maksymalna cena

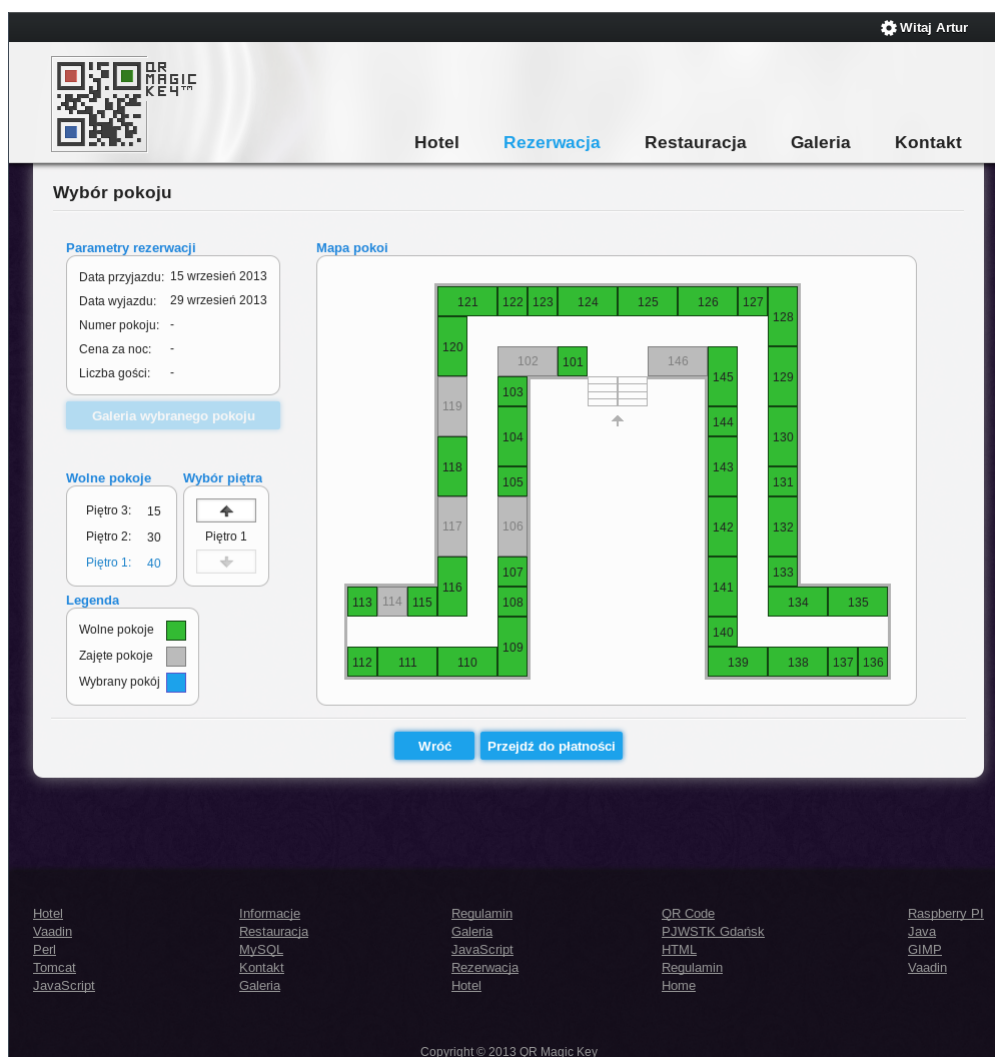
Text

Kontynuuj

Stopka

Rysunek 5.27: Szkielet dla ekranu dla zalogowanego i niezalogowanego użytkownika - parametry rezerwacji

5.6.3 Wygląd ekranów



Rysunek 5.28: Zrzut ekranu prezentujący ekran wyboru pokoju w widoku Gościa hotelowego

Witaj Adam

Wybierz raport

Aktualne rezerwacje
Przychód z ostatnich 7 dni
Przychód z ostatnich 30 dni
Lista najczęstszych gości
Raport ilości sprzątaných pokoi (30 dni)
Raport czasu sprzątaných pokoi (30 dni)
Użytkownicy QRMK

Opis raportu

Raport zawiera wszystkie aktualne rezerwacje

Generuj raport

			UŻYTKOWNIKA	NAZWISKO UŻYTKOWNIKA	DATA ROZPOCZĘCIA	DATA ZAKOŃCZENIA
				Żurawski	25-08-2013	30-08-2013
				Kowalczyk	24-08-2013	31-08-2013
				Smakulski	24-08-2013	29-08-2013
130	128	maniek	Marian	Stallone	24-08-2013	29-08-2013
134	206	malina	Malina	Zając	24-08-2013	06-09-2013
138	211	maniek	Marian	Stallone	24-08-2013	30-08-2013
120	132	adrianna	Adrianna	Woźniak	24-08-2013	04-09-2013
125	139	szyszy	Thierry	Henry	24-08-2013	28-08-2013
121	143	adrianna	Adrianna	Woźniak	24-08-2013	29-08-2013
116	129	szymon	Szymon	Kowalczyk	24-08-2013	01-09-2013
115	120	adam	Adam	Smakulski	23-08-2013	30-08-2013
118	126	radomil	Radomił	Nowak	22-08-2013	30-08-2013
109	101	szymon	Szymon	Kowalczyk	21-08-2013	28-08-2013
124	102	szyszy	Thierry	Henry	20-08-2013	03-09-2013

[Hotel](#)
[Vaadin](#)
[Perl](#)
[Tomcat](#)
[JavaScript](#)

[Informacje](#)
[Restauracja](#)
[MySQL](#)
[Kontakt](#)
[Galeria](#)

[Regulamin](#)
[Galeria](#)
[JavaScript](#)
[Rezerwacja](#)
[Hotel](#)

[QR Code](#)
[PJWSTK Gdansk](#)
[HTML](#)
[Regulamin](#)
[Home](#)

[Raspberry PI](#)
[Java](#)
[GIMP](#)
[Vaadin](#)

Copyright © 2013 QR Magic Key

Rysunek 5.29: Zrzut ekranu prezentujący ekran generowania raportów w widoku Managera



Witaj Szymon

Lista pokoi

NUMER POKOJU	CENA	LICZBA GOŚCI	POŁOŻENIE	STATUS POKOJU	OPIS
101	90	2	8.3.1.1	gotowy	
102	120	3	6.3.2.1	wyłączony z użytku	brak żarówek
103	95	2	6.4.1.1	gotowy	urwany kurek w umywalce
104	150	4	6.5.1.2	gotowy	
105	90	2	6.7.1.1	gotowy	
106	90	2	6.8.1.2	wyłączony z użytku	
107	95	2	6.10.1.1	gotowy	
108	90	2	6.11.1.1	gotowy	
109	110	2	6.12.1.2	gotowy	
110	90	2	4.13.2.1	gotowy	nowy opis dla pokoju
111	90	2	2.13.2.1	gotowy	
112	90	2	1.13.1.1	gotowy	
113	90	2	1.11.1.1	gotowy	
114	90	2	2.11.1.1	wyłączony z użytku	szambo wylało
115	90	2	3.11.1.1	gotowy	
116	90	4	4.10.1.2	gotowy	
117	90	2	4.8.1.2	wyłączony z użytku	trup w salonie
118	90	2	4.6.1.2	gotowy	
119	90	2	4.4.1.2	gotowy	
120	90	2	4.2.1.2	gotowy	

[Hotel](#)
[Vaadin](#)
[Perl](#)
[Tomcat](#)
[JavaScript](#)

[Informacje](#)
[Restauracja](#)
[MySQL](#)
[Kontakt](#)
[Galeria](#)

[Regulamin](#)
[Galeria](#)
[JavaScript](#)
[Rezerwacja](#)
[Hotel](#)

[QR Code](#)
[P.JWSTK Gdańsk](#)
[HTML](#)
[Regulamin](#)
[Home](#)

[Raspberry PI](#)
[Java](#)
[GIMP](#)
[Vaadin](#)

Copyright © 2013 QR Magic Key

Rysunek 5.30: Zrzut ekranu prezentujący ekran listy pokoi w widoku Administratora

Rozdział 6

Wydanie I

6.1 Implementacja

W wydaniu pierwszym oprócz implementacji modułów zawartych w harmonogramie, który znajduje się na stronie 9, powstał szkielet aplikacji internetowej oparty o wzorzec projektowy MVP. Korzystano ze środowiska programistycznego Eclipse oraz rozproszonego systemu kontroli wersji GIT, który ułatwił nam niezależną pracę. Jako łącze do bazy danych wykorzystano interfejs JDBC.

6.1.1 Implementacja bazy danych

Przy tworzeniu aplikacji skorzystano z biblioteki Hibernate. Jest to framework pozwalający realizować w prosty i szybki sposób obsługę przechowywania danych w bazie danych, niezależnie od systemu baz danych. Hibernate jest warstwą pośredniczącą pomiędzy systemem baz danych, a aplikacją i dostarcza translację, czy też mapowanie obiektów z Javy, na relacyjne bazy danych. Strukturę mapowania została zdefiniowana przy użyciu adnotacji w klasach.

Poniżej przedstawiono fragment przykładowej klasy z użyciem adnotacji:

```
1 @Entity
2 @Table(name = "rep_report")
3 @NamedQueries({
4     @NamedQuery(name = Report.FIND_ALL_REPORTS, query = "FROM Report"),
5     @NamedQuery(name = Report.FIND_REPORT_BY_ID, query = "FROM Report where pkReportId=:pkReportId")
6 })
7 public class Report implements Serializable {
8
9     private static final long serialVersionUID = 7421524697502172586L;
10
11     public static final String FIND_ALL_REPORTS = "Report.findAllReports";
12     public static final String FIND_REPORT_BY_ID = "Report.findReportById";
13
14     @Id
15     @Column(name = "pk_report_id")
16     private short pkReportId;
17
18     @Column(name = "name", length = 100, nullable = false)
```



```

19 |     private String      name;
20 |
21 |     @Column(name = "description", length = 500, nullable = true)
22 |     private String      description;
23 |
24 |     @Column(name = "query", length = 1000, nullable = false)
25 |     private String      query;
26 |
27 |     ...
28 | }

```

6.1.2 Moduły

W wydaniu pierwszym zaimplementowane zostały moduły wymagane do przeprowadzenia procesu rezerwacji oraz walidacji kodu przez czytnik.

6.1.2.1 Moduł rezerwacji pokoi

Moduł rezerwacji pokoi objął funkcjonalności dotyczące logowania użytkownika, wybór parametrów rezerwacji stanowiących jedynie datę przyjazdu oraz datę wyjazdu, prosty wybór pokoju z listy dostępnych dla podanych dat oraz symulację formularza płatności mającego na celu przetestowanie systemu blokady pokoju w celu uniemożliwienia rezerwacji przez więcej niż jednego użytkownika.

6.1.2.2 Moduł generowania i wysyłania kodu QR

Do generowania kodów QR wykorzystano bibliotekę ZXing [10]. Do wysyłania wiadomości e-mail zawierających wygenerowany kod został użyty interfejs JavaMail oraz bezpłatny serwis webmail - Gmail.

6.1.3 Napotkane problemy

Jedynym problemem podczas implementacji pierwszego wydania była zmiana wersji frameworku Vaadin z dotychczas używanej 6 na najnowszą wersję 7. Było to spowodowane chęcią poznania nowych funkcjonalności oferowanych przez framework. Uznane zostało, że zmiana nie spowoduje opóźnień względem harmonogramu.

6.1.4 Wybrane algorytmy

Poniżej przedstawiono fragment klasy prezentera wyboru dat:

```

1 | @Override
2 | public void datesSelected( final DateTime startDate, final DateTime endDate, final Integer capacity, final Integer price) {
3 |
4 |     if (!this.validateDates(startDate, endDate)) {
5 |         Confirmation.showSingle(Caption.INFORMATION, Message.INVALID_EXTEND_DATE);
6 |         return;
7 |     }
8 |
9 |     this.reservation.setStartDate(Time.getWithStartHour(startDate));

```

```

10 |     this.reservation.setEndDate(Time.getWithEndHour(endDate));
11 |     this.reservation.setPrice(price);
12 |     this.reservation.setCapacity(capacity);
13 |     this.bookingView.showRoomSelectionView();
14 | }
15 |
16 | private boolean validateDates(final DateTime startDate, final DateTime endDate) {
17 |
18 |     final DateTime now = DateTime.now();
19 |     final DateTime yesterday = DateTime.now().minusDays(1);
20 |
21 |     if (Time.noon(startDate).isBefore(Time.noon(yesterday)) ||
22 |         Time.noon(endDate).isBefore(Time.noon(now)) ||
23 |         !Time.noon(endDate).isAfter(Time.noon(startDate)) ||
24 |         (!Time.noon(startDate).isAfter(Time.noon(yesterday)) && now.getHourOfDay() > Time.LATE_LIMIT) ) {
25 |         return false;
26 |     }
27 |
28 |     return true;
29 | }

```

Poniżej przedstawiono metodę generującą kod QR z użyciem biblioteki ZXing:

```

1 | @Override
2 | public String createCode(final Reservable reservable) {
3 |
4 |     if (null == QRCodeServiceImpl.path) {
5 |         QRCodeServiceImpl.path = VaadinServlet.getCurrent().getServletContext().getRealPath("/") + "codes/";
6 |         final File dir = new File(QRCodeServiceImpl.path);
7 |         if (!dir.exists()) {
8 |             dir.mkdirs();
9 |         }
10 |    }
11 |
12 |    final StringBuilder qrHash = new StringBuilder(String.valueOf(reservable.getHash()).append(reservable.getId()));
13 |
14 |    final Charset charset = Charset.forName("UTF-8");
15 |
16 |    final CharsetParameter encoder = charset.newEncoder();
17 |
18 |    byte[] b = null;
19 |
20 |    try {
21 |        // Convert a string to ISO-8859-1 bytes in a ByteBuffer
22 |        final ByteBuffer bbuf = encoder.encode(CharBuffer.wrap(qrHash));
23 |        b = bbuf.array();
24 |    } catch (final CharacterCodingException e) {
25 |        QRCodeServiceImpl.LOGGER.error(null, e);
26 |        return null;
27 |    }
28 |
29 |    String data = null;
30 |    try {
31 |        data = new String(b, "UTF-8");
32 |    } catch (final UnsupportedEncodingException e) {
33 |        QRCodeServiceImpl.LOGGER.error(null, e);
34 |        return null;
35 |    }
36 |
37 |    // get a byte matrix for the data
38 |    BitMatrix matrix = null;
39 |    final int h = 300;
40 |    final int w = 300;
41 |    final com.google.zxing.Writer writer = new QRCodeWriter();

```

```

42
43     try {
44         final Map<EncodeHintType, Object> hints = new EnumMap<EncodeHintType, Object>(EncodeHintType.class);
45         hints.put(EncodeHintType.CHARACTER_SET, "UTF-8");
46         hints.put(EncodeHintType.MARGIN, 1);
47         matrix = writer.encode(data, BarcodeFormat.QR_CODE, w, h, hints);
48     } catch (final com.google.zxing.WriterException e) {
49         QRCodeServiceImpl.LOGGER.error(null, e);
50         return null;
51     }
52
53     final String fileName = qrHash + ".png";
54     final String filePath = QRCodeServiceImpl.path + fileName;
55     final File file = new File(filePath);
56     if (!file.exists()) {
57         try {
58             file.createNewFile();
59         } catch (final IOException e) {
60             QRCodeServiceImpl.LOGGER.error(null, e);
61             return null;
62         }
63     }
64
65     try {
66         MatrixToImageWriter.writeToFile(matrix, "PNG", file);
67     } catch (final IOException e) {
68         QRCodeServiceImpl.LOGGER.error(null, e);
69         return null;
70     }
71     return filePath;
72 }

```

6.1.5 Raspberry Pi

6.1.5.1 Specyfikacja



Rysunek 6.1: Platforma komputerowa Raspberry Pi

Komputer Raspberry Pi [11] jest urządzeniem stworzonym przez Raspberry Pi Foundation opartym na układzie Broadcom BCM2835 składającym się z procesora ARM1176JZF-S taktowanym zegarem 700 MHz oraz procesorem graficznym VideoCore IV. Ponadto urządzenie posiada m.in. 512 MB pamięci RAM współdzielonej z GPU, dwa porty USB2.0, port Ethernet 10/100 do komunikacji sieciowej oraz złącze General Purpose Input/Output (GPIO) służące do komunikacji komputera Raspberry pi z urządzeniami peryferyjnymi.

6.1.5.2 Urządzenia peryferyjne

Kamera W celu umożliwienia odczytu kodów QR użyta została kamera QuickCam for Notebooks Pro firmy Logitech. Kamera została wybrana z powodu bezproblemowej obsługi w systemie Linux.

6.1.5.3 System operacyjny

Komputer Raspberry pi pracuje pod kontrolą systemu Raspbian “wheezy” dostępnego pod adresem <http://www.raspberrypi.org/downloads> .

6.1.5.4 Przygotowanie środowiska

Po instalacji systemu wykonano następujące czynność mające na celu przygotowanie środowiska:

Instalacja wymaganych pakietów i bibliotek dla języka Perl

```
1 # apt-get install libdbd-mysql-perl libbarcode-zbar-perl libproc-daemon-perl screen libv4l-0
```

Pobranie i kompilacja bibliotek umożliwiających komunikację poprzez port GPIO

```
1 # sudo su
2 # cd /tmp/
3 # wget http://www.airspayce.com/mikem/bcm2835/bcm2835-1.26.tar.gz &&
4 # tar zxvf bcm2835-1.*.tar.gz && cd bcm2835-1.* && ./configure && make && make check && make install
5 #wget http://search.cpan.org/CPAN/authors/id/M/MI/MIKEM/Device-BCM2835-1.9.tar.gz &&
6 #tar zxvf Device-BCM2835-1.*.tar.gz && cd Device-BCM2835* && perl Makefile.PL && make install
```

Ustawienia prawidłowej strefy czasowej

```
1 # sudo su
2 # ln -sf /usr/share/zoneinfo/Europe/Warsaw /etc/localtime
```

Utworzenie klucza RSA. Utworzony klucz należy skopiować do repozytorium.

```
1 # sudo su
2 # ssh-keygen
3 # cat ~/.ssh/id_rsa.pub
```

Pobranie źródeł oprogramowania dla czytnika kodów QR

```
1 # sudo su
2 # mkdir /etc/qr_reader
3 # cd /etc/qr_reader/
4 # git init
5 # git pull git@bitbucket.org:pjwstk/qr_reader.git
```

Uruchamianie czytnika przy starcie systemu

```
1 # ln -sf /etc/qr_reader/qr_scanner /etc/init.d/
2 # update-rc.d qr_scanner defaults
```

6.1.5.5 Moduł walidacji poprawności kodu QR

Zaimplementowany został moduł walidacji poprawności kodu QR. Generowane przez system kody mają postać ciągu cyfr. W ciągu zawarty jest numer pokoju, ID rezerwacji w bazie danych oraz losowo wygenerowany hash. Aby ustrzec się przed wstrzyknięciem skryptu SQL (SQL Injection) poprzez kod QR lub też nadmiernej ilości zapytań kierowanych do bazy danych przez czytniki, każdorazowy odczyt kodu jest filtrowany przez funkcję wyrażenia regularnego (Regular expression). Jeżeli zawartość kodu QR posiada znaki inne niż cyfry zawartość kodu nie jest przekazywana do funkcji komunikującej się z bazą danych.

6.2 Testy

6.2.1 Testy jednostkowe

W celu przeprowadzenia testów jednostkowych wykorzystano bibliotekę JUnit 4. Celem testów jednostkowych było sprawdzenie poprawności działania pojedynczych metod oraz klas poprzez porównywanie oczekiwanych rezultatów z wynikami.

6.2.2 Testy integracyjne

Testy integracyjne zostały wykonane w celu wykrycia ewentualnych błędów w interakcji pomiędzy modułami. Zostały przeprowadzone dokładne testy integracji czytnika kodów z aplikacją. Generowane kody QR były sprawdzane pod kątem poprawności zawartych w nich danych i prawidłowej reakcji czytnika na nie.

6.2.3 Testy walidacyjne/systemowe

Podczas testów systemowych powstała do tej pory część systemu została sprawdzona pod kątem zgodności z wymaganiami obejmującymi wydanie pierwsze.

Nazwa testu	Logowanie użytkownika
Opis testowanego przypadku	Logowanie użytkownika poprzez wypełnienie formularza logowania
Oczekiwane wartości	Po zalogowaniu użytkownik powinien uzyskać dostęp do części dostępnej tylko dla zalogowanych użytkowników typu 'guest'
Wartości uzyskane	Dostęp do wyżej wymienionej części aplikacji
Wynik testu	pozytywny

Nazwa testu	Wyszukanie dostępnych pokoi
Opis testowanego przypadku	Wyszukiwanie dostępnych pokoi poprzez podanie daty przyjazdu oraz daty wyjazdu
Oczekiwane wartości	Przejdzie do widoku umożliwiającego wybranie jednego z dostępnych pokoi
Wartości uzyskane	Użytkownik został przekierowany do wyżej wymienionego widoku
Wynik testu	pozytywny

Nazwa testu	Rezerwacja pokoju
Opis testowanego przypadku	Wybór jednego z dostępnych pokoi oraz przejście przez symulowaną płatność
Oczekiwane wartości	Wysłanie kodu QR na adres email przypisany do użytkownika dokonującego rezerwacji
Wartości uzyskane	Użytkownik otrzymał kod QR drogą mailową
Wynik testu	pozytywny

6.3 Raport końcowy

Moduły wykonane w pierwszym wydaniu to:

- Logowanie
- Rezerwacja pokoju
- Wysłanie kodu QR
- Walidacja kodu QR

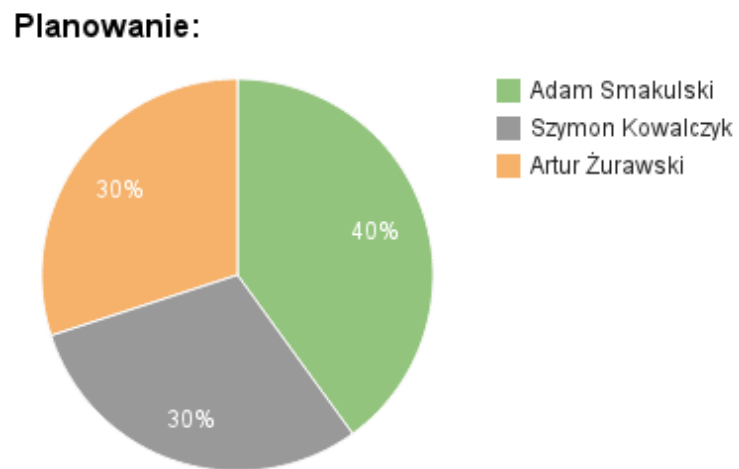
6.3.1 Porównanie założeń z realizacją

Wszystkie założenia dotyczące wydania pierwszego zostały zrealizowane zgodnie z harmonogramem.

6.3.2 Rozkład pracy

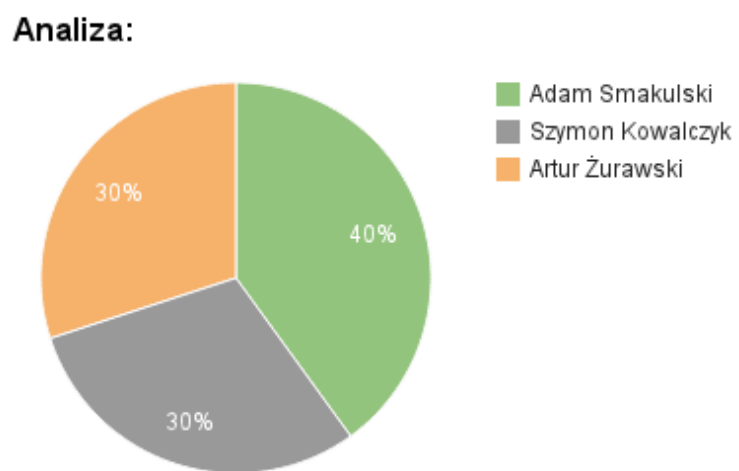
Dane przedstawione na poniższych diagramach kołowych są szacunkowe. W odniesieniu do wydania pierwszego szacowany czas pracy wynosił: 540h; 100 - planowanie; 100 - analiza; 250 - projektowanie; 70 - implementacja; 20 - testowanie.

6.3.2.1 Planowanie



Rysunek 6.2: Planowanie: Rozkład pracy

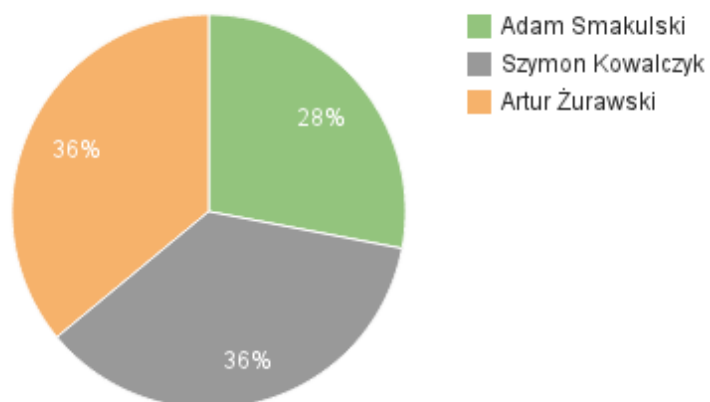
6.3.2.2 Analiza



Rysunek 6.3: Analiza: Rozkład pracy

6.3.2.3 Projektowanie

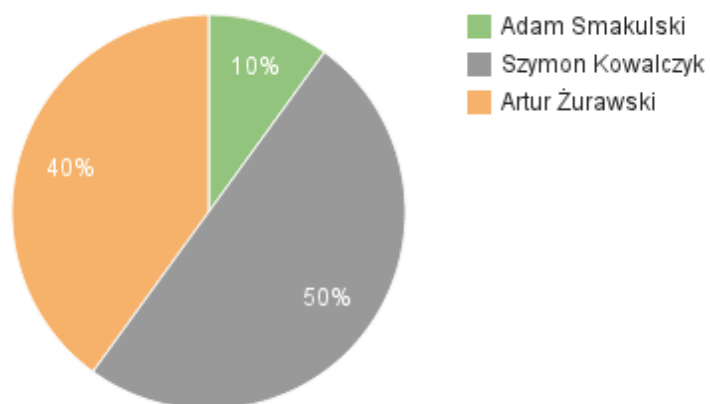
Projektowanie:



Rysunek 6.4: Projektowanie: Rozkład pracy

6.3.2.4 Implementacja

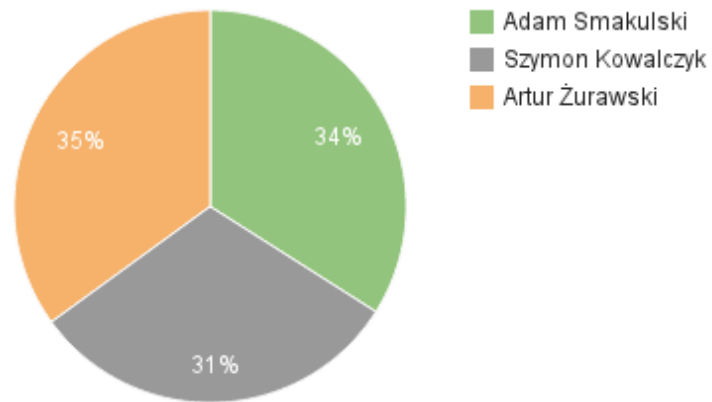
Implementacja:



Rysunek 6.5: Implementacja: Rozkład pracy

6.3.2.5 Testowanie

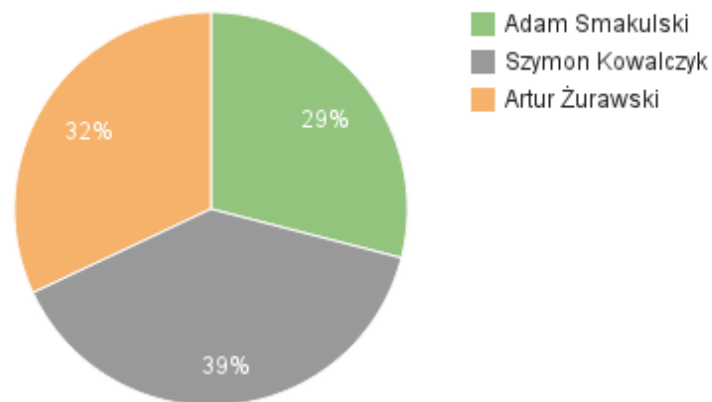
Testowanie:



Rysunek 6.6: Testowanie: Rozkład pracy

6.3.2.6 Rozkład całkowity

Rozkład całkowity:



Rysunek 6.7: Całkowity rozkład pracy

Rozdział 7

Wydanie II

7.1 Implementacja

7.1.1 Moduły

Moduły wykonane w drugim wydaniu.

7.1.1.1 Rejestracja

Do modułu logowania z pierwszego wydania został dodany moduł rejestracji. Jest to niezbędny element do rezerwacji pokoju. Użytkownik jest identyfikowany za pomocą loginu.

7.1.1.2 Graficzny wybór pokoi

Mapa pokoi została stworzona z wykorzystaniem komponentów oferowanych przez framework Vaadin. Dane potrzebne do rozmieszczenia pokoi na mapie pobierane są z bazy danych. W panelu administratora zaimplementowana została funkcjonalność pozwalająca na dodawanie pokoi oraz późniejszą edycję ich parametrów.

7.1.1.3 Moduł osoby sprzątającej

Moduł osoby sprzątającej został stworzony z myślą o zautomatyzowaniu procesu przekazywania danych o tym jaki pokój wymaga posprzątania oraz jaki jest już posprzątany, gotowy do wynajęcia. Osoba sprzątająca posiada swój odrębny widok w aplikacji i ma możliwość dodawania pokoi do swojej listy, zmiany ich stanu oraz usuwania z listy.

7.1.1.4 Moduł administratora

Moduł administratora dostępny jest dla konta z uprawnieniami administratora i pozwala na edycję kont użytkowników oraz konfigurację parametrów nowych lub istniejących pokoi w hotelu. W pierwszym założeniu wszystkie parametry pokoju miały być dostępne z poziomu

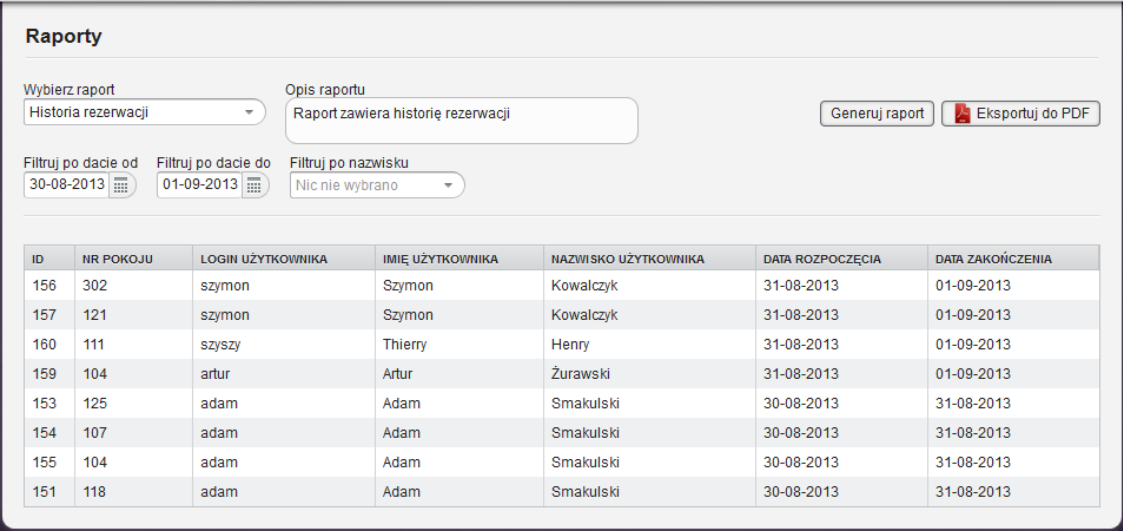
Administradora. W związku z tym iż niektóre parametry pokoi powinny być dostępne dla Managera, zostały rozdzielone pomiędzy konto Managera (cena, opis, stan) i Administratora (edycja numeru, położenia oraz rozmiar).

7.1.1.5 Moduł managera

Moduł Managera został zaimplementowany głównie z myślą o możliwości generowania raportów. Ze względu na późniejsze, nieplanowane dodanie edycji parametrów pokoju do konta Managera, zdecydowaliśmy, że panel tego użytkownika zostanie podzielony na dwa pod-widoki: Lista pokoi i Raporty. Do generowania raportów użyliśmy biblioteki iText.

7.1.1.5.1 Raporty Moduł raportowy zawiera dynamicznie generowany panel z wyborem raportu oraz z różnymi typami filtrów dla raportu. Widok panelu jest w pełni konfigurowalny dla każdego raportu z poziomu bazy danych, dzięki czemu czas na wprowadzenie nowego raportu jest krótki i wynosi około 15 minut dla standardowego raportu z polami do filtrowania. Moduł jest w pełni integrowalny z pozostałymi modułami w aplikacji oraz może być zintegrowany z innymi aplikacjami wykorzystującymi te same lub inne technologie przy niewielkim nakładzie pracy.

Poniżej przedstawiony jest ekran z wygenerowanego raportu “Historia rezerwacji”



The screenshot shows a web interface titled "Raporty". It includes a dropdown menu for "Wybierz raport" (Select report) with "Historia rezerwacji" (Reservation history) selected. A text field for "Opis raportu" (Report description) contains "Raport zawiera historię rezerwacji". There are buttons for "Generuj raport" (Generate report) and "Eksportuj do PDF" (Export to PDF). Below these are three filter sections: "Filtruj po dacie od" (Filter by date from) with "30-08-2013", "Filtruj po dacie do" (Filter by date to) with "01-09-2013", and "Filtruj po nazwisku" (Filter by name) with "Nic nie wybrano". The main part of the interface is a table with 7 columns: ID, NR POKOJU, LOGIN UŻYTKOWNIKA, IMIĘ UŻYTKOWNIKA, NAZWISKO UŻYTKOWNIKA, DATA ROZPOCZĘCIA, and DATA ZAKOŃCZENIA. The table contains 8 rows of reservation data.

ID	NR POKOJU	LOGIN UŻYTKOWNIKA	IMIĘ UŻYTKOWNIKA	NAZWISKO UŻYTKOWNIKA	DATA ROZPOCZĘCIA	DATA ZAKOŃCZENIA
156	302	szymon	Szymon	Kowalczyk	31-08-2013	01-09-2013
157	121	szymon	Szymon	Kowalczyk	31-08-2013	01-09-2013
160	111	szyszy	Thierry	Henry	31-08-2013	01-09-2013
159	104	artur	Artur	Żurawski	31-08-2013	01-09-2013
153	125	adam	Adam	Smakulski	30-08-2013	31-08-2013
154	107	adam	Adam	Smakulski	30-08-2013	31-08-2013
155	104	adam	Adam	Smakulski	30-08-2013	31-08-2013
151	118	adam	Adam	Smakulski	30-08-2013	31-08-2013

Rysunek 7.1: Historia rezerwacji

Wygenerowany raport można przenieść do pliku PDF. Na poniższym ekranie wydruk historii rezerwacji. Do generowania raportów została użyta biblioteka iText.



Nazwa raportu: Historia rezerwacji

Opis raportu: Raport zawiera historię rezerwacji

Report wygenerowany przez: Manager of QR Magic Key, N 01.09.2013, 15:48:42

ID	Nr pokoju	Login użytkownika	Imię użytkownika	Nazwisko użytkownika	Data rozpoczęcia	Data zakończenia
156	302	szymon	Szymon	Kowalczyk	31-08-2013	01-09-2013
157	121	szymon	Szymon	Kowalczyk	31-08-2013	01-09-2013
160	111	szyszy	Thierry	Henry	31-08-2013	01-09-2013
159	104	artur	Artur	Żurawski	31-08-2013	01-09-2013
153	125	adam	Adam	Smakulski	30-08-2013	31-08-2013
154	107	adam	Adam	Smakulski	30-08-2013	31-08-2013
155	104	adam	Adam	Smakulski	30-08-2013	31-08-2013
151	118	adam	Adam	Smakulski	30-08-2013	31-08-2013

Rysunek 7.2: Historia rezerwacji

7.1.2 Napotkane problemy

Podczas implementacji modułów dla wydania drugiego napotkano na kilka problemów. Głównym problemem było nieprzemyślane rozmieszczenie widoków w poszczególnych modułach. Kolejnym problemem były nieudane próby instalacji dedykowanej kamery do czytnika kodów. Przy użyciu dedykowanej kamery nie udało się osiągnąć szybkości przetwarzania obrazu porównywalnej do tej, którą udało się osiągnąć przy użyciu kamery USB. Zaistniałe sytuacje spowodowały opóźnienia względem harmonogramu, ale ostatecznie nie przeszkodziły w pomyślnym zakończeniu projektu.

7.1.3 Przykładowe algorytmy

Poniżej przedstawiono metodę generującą mapę pokoi:

```
1 @Override
2 public void showRooms() {
3     this.mapLayout.removeAllComponents();
```

```

4      this.floorUpButton.setEnabled(this.floorNumber != Numbers.MAX_FLOOR.number);
5      this.floorDownButton.setEnabled(this.floorNumber != Numbers.MIN_FLOOR.number);
6      this.clickableRooms.clear();
7
8      if (this.availableRooms.isEmpty()) {
9          Notification.show(Message.ROOM_LIST_EMPTY, Notification.Type.HUMANIZED_MESSAGE);
10     }
11
12     for (final Room room : this.availableRooms) {
13         if ((room.getNumber() / Numbers.FLOOR_MULTIPLY.number) != this.floorNumber) {
14             continue;
15         }
16         final String[] position = room.getPosition().split(",");
17         final int x = Integer.parseInt(position[Numbers.X.number].trim());
18         final int y = Integer.parseInt(position[Numbers.Y.number].trim());
19         final int width = Integer.parseInt(position[Numbers.WIDTH.number].trim()) * Size.ROOM_CELL;
20         final int height = Integer.parseInt(position[Numbers.HEIGHT.number].trim()) * Size.ROOM_CELL;
21
22         final RoomButton roomButton = new RoomMapButtonImpl(width, height, room.getNumber());
23
24         this.mapLayout.addComponent(roomButton, "left: " + (x * Size.ROOM_CELL) + "; top: " + (y * Size.ROOM_CELL
25     ));
26     roomButton.addClickListener(this);
27     roomButton.setRoom(room);
28     this.clickableRooms.add(roomButton);
29 }
30
31 for (final Room room : this.unavailableRooms) {
32     if ((room.getNumber() / Numbers.FLOOR_MULTIPLY.number) != this.floorNumber) {
33         continue;
34     }
35     final String[] position = room.getPosition().split(",");
36     final int x = Integer.parseInt(position[Numbers.X.number].trim());
37     final int y = Integer.parseInt(position[Numbers.Y.number].trim());
38     final int width = Integer.parseInt(position[Numbers.WIDTH.number].trim()) * Size.ROOM_CELL;
39     final int height = Integer.parseInt(position[Numbers.HEIGHT.number].trim()) * Size.ROOM_CELL;
40
41     final HorizontalLayout unavailableRoom = new HorizontalLayout();
42     final Label roomNumber = new Label(String.valueOf(room.getNumber()));
43     roomNumber.setSizeUndefined();
44     unavailableRoom.setWidth(width + Size.PIXELS);
45     unavailableRoom.setHeight(height + Size.PIXELS);
46     unavailableRoom.addComponent(roomNumber);
47     unavailableRoom.setComponentAlignment(roomNumber, Alignment.MIDDLE_CENTER);
48     this.mapLayout.addComponent(unavailableRoom, "left: " + (x * Size.ROOM_CELL) + "; top: " + (y * Size.
49     ROOM_CELL));
50     unavailableRoom.setStyleName(Style.ROOM_DISABLED);
51 }
52
53 if (null != this.selectedRoom) {
54     for (final RoomButton button : this.clickableRooms) {
55         final String styleName;
56         if (this.selectedRoom == button.getRoom()) {
57             styleName = Style.ROOM_SELECTED;
58         } else {
59             styleName = Style.ROOM;
60         }
61         button.setStyleName(styleName);
62     }
63 }

```

Poniżej przedstawiono zapytanie pobierające raport dotyczący czasu sprzątania pokoi:

```
1 SELECT DATE_FORMAT(END_DATE, '%d-%m-%Y'),
2     FIRST_NAME,
3     LAST_NAME,
4     NUMBER,
5     CONCAT(TIMESTAMPDIFF(HOUR,START_DATE,END_DATE), ' godzin ',
6     MOD( TIMESTAMPDIFF(MINUTE,START_DATE, END_DATE), 60), ' minut ')
7 FROM QR_CLEANING_RECORD
8 JOIN QR_USER ON QR_CLEANING_RECORD.CLEANER_ID=QR_USER.ID
9 JOIN QR_ROOM ON QR_CLEANING_RECORD.ROOM_ID=QR_ROOM.ID
10 WHERE END_DATE >= SUBDATE(CURDATE(), INTERVAL 30 DAY)
11 ORDER BY DATE(END_DATE) DESC
```

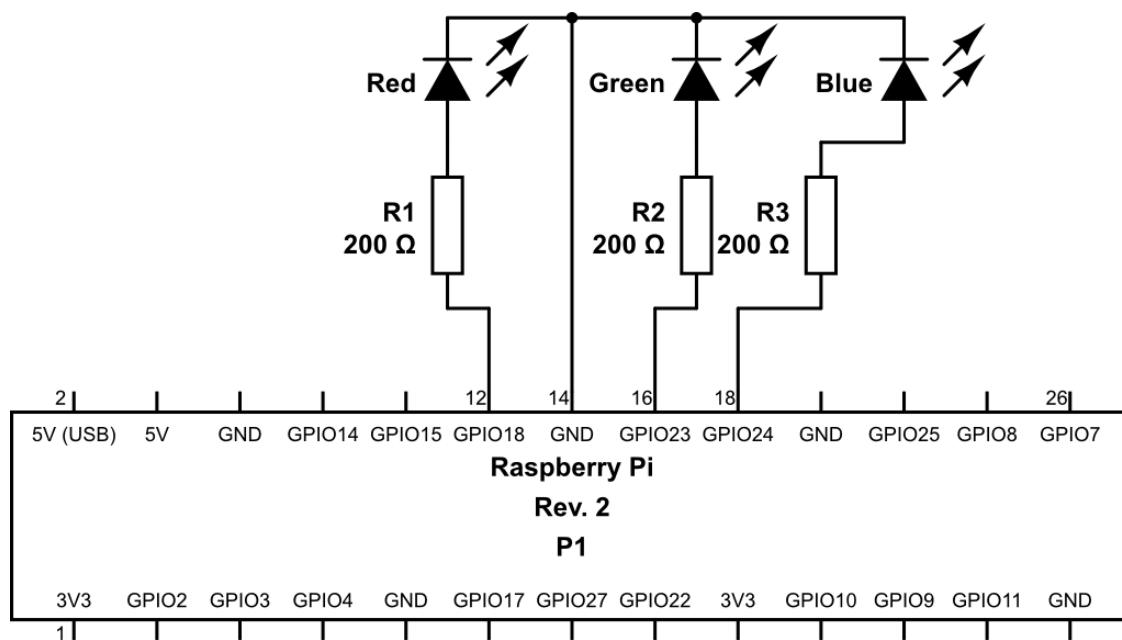
7.1.4 RaspberryPi

7.1.4.1 Urządzenia peryferyjne

Kamera W maju 2013 Raspberry Pi Foundation wyprodukowało dedykowaną kamerę dla komputera Raspberry Pi podłączaną do portu CSI. Kamera została zaimplementowana testowo. Urządzenie niestety nie posiadało możliwości zamontowania w systemie Linux jako urządzenie blokowe. Było to przyczyną niemożliwość natywnej obsługi kamery przez biblioteki Perla rozpoznające kody QR. Jedyną możliwością odczytu kodów QR była analiza strumienia video z kamery. Przetwarzanie strumienia video było znacznie wolniejszą metodą analizy kodów QR w porównaniu do wykorzystanej pierwotnie kamery USB. Spadek responsywności był na tyle duży, iż zrezygnowano z wykorzystania natywnej kamery przy odczycie kodów QR.

Dioda RGB Do komputera RaspberryPi dodano diodę LED wraz z rezystorami 200Ω. Dioda służy do komunikacji komputera z użytkownikiem informując o efektach walidacji kodu QR. Na rysunku 7.3 przedstawiono sposób podłączenia diody do portu GPIO. Dioda została zaprogramowana aby świecić się w trzech kolorach:

- Niebieski, w przypadku gdy kod QR został rozpoznany, ale nie przeszedł testu walidacji - zawierał znaki inne niż cyfry.
- Czerwony, w przypadku gdy kod QR został rozpoznany, przeszedł walidację zawartości, ale zapytanie do bazy nie zwróciło pozytywnych wyników.
- Zielony, w przypadku gdy kod QR został rozpoznany, przeszedł walidację zawartości i zwrócił pozytywną odpowiedź na zadane zapytanie z bazy danych.



Rysunek 7.3: Schemat podłączenia diody LED RGB do portów GPIO. Schemat jak również i sam układ został wykonany przez zespół.

7.2 Testy

7.2.1 Testy walidacyjne/systemowe

Podczas testów systemowych system został sprawdzony pod kątem zgodności z listą wymagań funkcjonalnych.

Nazwa testu	Rejestracja, logowanie i edycja danych
Opis testowanego przypadku	Po wywołaniu formularza rejestracji poprzez wybór opcji „załóż konto” w pierwszej kolejności wypełniano pola danymi, które nie powinny przejść walidacji pól, np. w polu adres email wpisano ciąg znaków bez wymaganego znaku @ rozdzielającego część symbolizującą nazwę konta od domeny i skrótu państwa. Następnie formularz wypełniono poprawnymi danymi i zatwierdzono przyciskiem „załóż konto”. System zwrócił komunikat o poprawnym założeniu konta. Następnie wybrano opcję „zaloguj się” i w odpowiednie miejsca wpisano login i hasło. Po akceptacji wprowadzonych danych zmieniła się górna belka w oknie systemu informująca o poprawnym zalogowaniu i dająca dostęp do panelu konta. Po wybraniu opcji „Edycja profilu” ukazało się okno zawierające dane konta. Po przeprowadzeniu walidacji pól zapisano nowe dane za pomocą przycisku „Zapisz”. System poinformował o zapisanych zmianach. Po wybraniu opcji zmiana hasła ukazało się okno zmiany hasła. W pierwszej kolejności podano złe stare hasło a gdy system zareagował na nieprawidłowość podano poprawne dane i zatwierdzono przyciskiem „Zapisz”.
Oczekiwane wartości	Poprawne przejście walidacji pól i założenie konta w systemie oraz logowanie do systemu przy użyciu podanych danych
Wartości uzyskane	Zgodne z oczekiwanymi
Wynik testu	Pozytywny

Nazwa testu	Rezerwacja pokoju
Opis testowanego przypadku	Po wywołaniu formularza rezerwacji poprzez wybór opcji „Rezerwacja” wybrano losowe daty przyjazdu i wyjazdu oraz zdefiniowano maksymalną liczbę dwóch gości. Po wybraniu opcji „Kontynuuj” ukazała się lista pokoi. Po wybraniu jednego z pokoi kliknięto opcję „Przejdź do płatności”, po czym ukazał się ekran płatności. W pierwszej kolejności wypełniono pola danymi innymi niż oczekiwane. Po ukazaniu się informacji od systemu o błędnie wprowadzonych danych do formularza pola wypełniono prawidłowymi danymi i potwierdzono wypełnienie formularza za pomocą przycisku „Potwierdź”. Ukazało się okno informujące o przesłaniu klucza na adres email przypisany do konta.
Oczekiwane wartości	Zarezerwowanie pokoju
Wartości uzyskane	Zgodne z oczekiwanymi
Wynik testu	Pozytywny

Nazwa testu	Zarządzanie rezerwacjami
Opis testowanego przypadku	Przed przystąpieniem do testu w bazie danych zostało dodanych kilka dodatkowych rezerwacji dla konta testowego. Po zalogowaniu do systemu wybrana została opcja „Moje rezerwacje” z rozwijanego menu na górnej belce okna systemowego. Wybrano listy „Planowane rezerwacje”, „Aktywne rezerwacje” oraz „Archiwalne rezerwacje” a w każdej z list znajdowały się odpowiednie dane wprowadzone przed przystąpieniem do testu. Z listy „Planowane rezerwacje” została wybrana rezerwacja, a następnie wybrano opcję „Przedłuż rezerwację” Ukazała się strona kalendarza pozwalająca na wybór nowej daty wyjazdu. Po kliknięciu „Sprawdź dostępność Twojego pokoju” system wyświetlił informacje o możliwości przedłużenia rezerwacji dla danego pokoju i otworzone zostało okno płatności. Po wypełnieniu i zatwierdzeniu danych został wysłany nowy klucz a w liście „Planowane rezerwacje” wyświetlona została nowa data zakończenia pobytu. Ta sama operacja została przeprowadzona dla rezerwacji z listy „Aktywne rezerwacje”. W następnej części testu została zaznaczona rezerwacja i wybrana opcja „Anuluj rezerwację”. Po potwierdzeniu chęci anulowania rezerwacji zaznaczona uprzednio rezerwacja została usunięta z listy „Planowane rezerwacje”.
Oczekiwane wartości	Przedłużenie i anulowanie rezerwacji dla planowanej i aktywnej rezerwacji
Wartości uzyskane	Zgodne z oczekiwanymi
Wynik testu	Pozytywny

Nazwa testu	Chwilowa blokada pokoju
Opis testowanego przypadku	Otworzone zostały dwie przeglądarki. Na każdej z nich po zalogowaniu na osobne konta została wyświetlona mapa pokoi dla tej samej daty. W pierwszej przeglądarce po wyborze konkretnego pokoju potwierdzono rezerwację i zatrzymano się w kroku płatności. W drugiej przeglądarce wybrano ten sam pokój do rezerwacji i wybrano opcję „Potwierdź”. System poinformował, iż pokój nie jest już dostępny, a na odświeżonym widoku pokój zmienił stan na „Zajęty”. Po 10 minutach odświeżono okno mapy pokoi, a wcześniej zajęty pokój zmienił stan na „Wolny” i rezerwacja ponownie była możliwa.
Oczekiwane wartości	Blokada rezerwacji pokoju przez więcej niż jednego użytkownika
Wartości uzyskane	Zgodne z oczekiwaniami
Wynik testu	Pozytywny

Nazwa testu	Walidacja kodów QR
Opis testowanego przypadku	W pierwszym przypadku został użyty kod wygenerowany podczas rezerwacji w poprzednim teście. Po zbliżeniu kodu QR do kamery komputera Raspberry Pi zapaliła się zielona dioda LED. Po zmianie daty rezerwacji / id / unikatowego numeru hash / numeru pokoju i zbliżeniu użytego wcześniej kodu QR dioda zapaliła się na czerwony kolor. Po wygenerowaniu w zewnętrznym generatorze kodów QR kodu zawierającego losowe cyfry i zbliżeniu do czytnika dioda zapaliła się na kolor czerwony. Po wygenerowaniu w zewnętrznym generatorze kodów QR kodu zawierającego losowe cyfry oraz inne znaki i zbliżeniu do czytnika dioda zapaliła się na kolor niebieski. Po odłączeniu czytnika od internetu i zbliżeniu kodu QR zawierającego poprawne dane dioda zapaliła się naprzemiennie na kolor czerwony, zielony i niebieski.
Oczekiwane wartości	Dioda czytnika prawidłowo reaguje na kody QR zawierające poprawne i niepoprawne dane jak również na sytuację gdy czytnik nie ma możliwości podłączenia się do bazy danych w celu weryfikacji kodu QR
Wartości uzyskane	Zgodne z oczekiwaniami
Wynik testu	Pozytywny

Nazwa testu	Utworzenie nowego konta, edycja i nadanie uprawnień przez Administratora
Opis testowanego przypadku	Po zalogowaniu na konto Administratora i wyboru opcji „Lista użytkowników” z menu Administratora ukazuje się ekran prezentujący listę użytkowników systemu. Po kliknięciu przycisku „Dodaj” wyświetla się ekran z polami opisującymi tworzonego użytkownika wraz z rozwijanym menu umożliwiającym wybór uprawnień. Po wypełnieniu pól i kliknięciu przycisku zapisz na liście prezentującej listę użytkowników pojawia się nowa pozycja z nowo wprowadzonymi danymi. Po wybraniu z listy nowo utworzonego użytkownika i kliknięciu przycisku „Edytuj” pojawia się ekran z możliwością wprowadzenia zmian w parametrach konta, w tym również jego uprawnień. Wpisane zostają nowe wartości a zmiany zostają zatwierdzone przyciskiem „Zapisz”. Po wylogowaniu z konta Administratora i zalogowaniu na konto edytowanego użytkownika, użytkownik posiada widok odpowiadający jego uprawnieniom.
Oczekiwane wartości	Utworzenie nowego konta, edycja i zmiana uprawnień przez Administratora
Wartości uzyskane	Zgodne z oczekiwaniami
Wynik testu	Pozytywny

Nazwa testu	Wyświetlenie stanów pokoi, wydrukowanie raportu i zmiana stanu pokoju przez Managera hotelu
Opis testowanego przypadku	Po zalogowaniu na konto Managera wyświetla się widok listy pokoi wraz z ich stanem i dodatkowymi atrybutami. Widok ten jest również dostępny po kliknięciu „Lista pokoi” z menu Managera. Po kliknięciu „Raporty” z menu managera wyświetla się widok służący do generowania raportów. Po rozwinięciu listy „Wybierz raport” wyświetla się lista wyboru typów raportów. Po wybraniu opcji „Aktualne rezerwacje” i zatwierdzenie przyciskiem „Generuj raport” wyświetla się lista aktualnych rezerwacji. Dostępne są również listy przychodów, historia sprzątnięcia pokoi oraz lista użytkowników. Po kliknięciu na ikonę PDF utworzony zostaje raport w formacie PDF możliwy do zapisania lokalnie na dysku.
Oczekiwane wartości	Generowanie raportów i wyświetlanie ich na ekranie jak również zapisanie w formacie PDF
Wartości uzyskane	Zgodne z oczekiwaniami
Wynik testu	Pozytywny

Nazwa testu	Zmiana statusu pokoju przez managera
Opis testowanego przypadku	Po zalogowanie na konto Managera i wyświetleniu listy pokoi nastąpił wybór pokoju i wybranie opcji „Edytuj”. Na nowym ekranie wyświetla się okno z możliwością wyboru stanu pokoju. Po zmianie z „Gotowy” na „Wyłączony z użytku” zostaje wybrana opcja „zapisz”. Po wylogowaniu z konta Managera i przejścia do widoku „Rezerwacja” pokój ze zmienionym statusem nie jest dostępny do zarezerwowania. Po zalogowaniu na konto Managera i ponownej zmianie statusu na „Gotowy” pokój jest znów dostępny do rezerwacji.
Oczekiwane wartości	Zmiana stanu pokoju z „Gotowy” na „Wyłączony z użytku” blokuje możliwość zarezerwowania pokoju.
Wartości uzyskane	Zgodne z oczekiwaniami
Wynik testu	Pozytywny

Nazwa testu	Zmiana statusu pokoju z „Do posprzątania” na „Posprzątny” poprzez konto Osoby sprzątającej. Testowanie kodów QR z dostępem do kilku pokoi.
Opis testowanego przypadku	Po zalogowaniu na konto Osoby sprzątającej wyświetla się lista pokoi wymagających sprzątnięcia. Po zaznaczeniu odpowiednich pokoi i wyboru opcji „Zapisz” wyświetla się informacja o zapisanych zmianach. Po przejściu do widoku „Moja lista” wyświetla się widok z listą pokoi w trakcie sprzątania. Po wybraniu opcji „Zmień stan” statu spokoju ulega zmianie z „w trakcie sprzątania” „posprzątny”. Opcja zapisz powoduje usunięcie z listy pokoi ze statusem „posprzątny”. Opcja „Wygeneruj klucz” powoduje wysłanie kodu QR na adres email przypisany do konta Osoby sprzątającej. Między godziną 11 a 15 kod QR zbliżony do czytnika z przypisanym numerem będącym jednocześnie na liście „Moja lista” Osoby sprzątającej powoduje zapalenie zielonej diody LED na czytniku kodów. Zbliżenie kodu QR do czytnika z przypisanym numerem pokoju nie będącym jednocześnie na liście „Moja lista” Osoby sprzątającej lub zbliżenie kodu do czytnika między godziną 15 a 11 powoduje zapalenie czerwonej diody LED na czytniku kodów. Zbliżenie kodu QR wygenerowanego dla konkretnej osoby sprzątającej i zbliżenie go do czytnika pokoju znajdującego się na liście innej osoby sprzątającej powoduje zapalenie się czerwonej diody LED na czytniku kodów.
Oczekiwane wartości	Zmiana statusu pokoju z „Do posprzątania” na „Posprzątny” poprzez konto Osoby sprzątającej oraz wygenerowanie kodu dającego dostęp tylko do konkretnych pokoi tylko w danym czasie.
Wartości uzyskane	Zgodne z oczekiwaniami
Wynik testu	Pozytywny

7.3 Raport końcowy

7.3.1 Moduły

Moduły wykonane w drugim wydaniu.

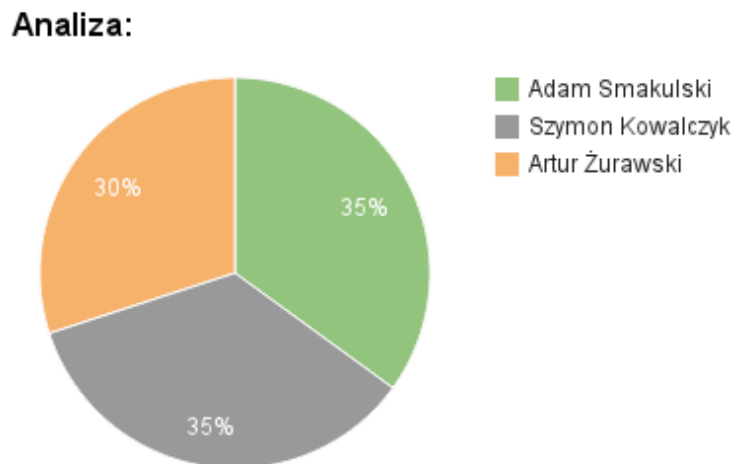
- Rejestracja
- Graficzny wybór pokoi

- Moduł osoby sprzątającej (możliwość manualnego przypisywania stanu pokojom)
- Moduł administratora (edycja i usuwanie kont)
- Moduł managera (generowanie raportów)

7.3.2 Rozkład pracy

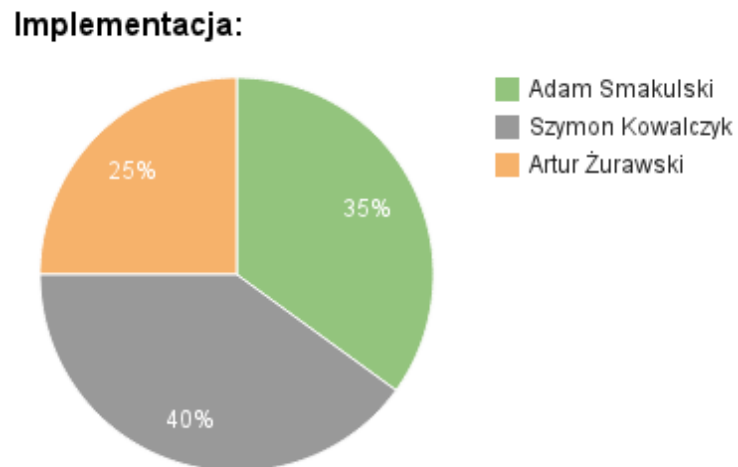
Dane przedstawione na poniższych diagramach kołowych są szacunkowe. W odniesieniu do wydania drugiego szacowany czas pracy wynosił: 1260h; 100 - analiza; 500 - implementacja; 300 - projektowanie; 360 - testowanie.

7.3.2.1 Analiza



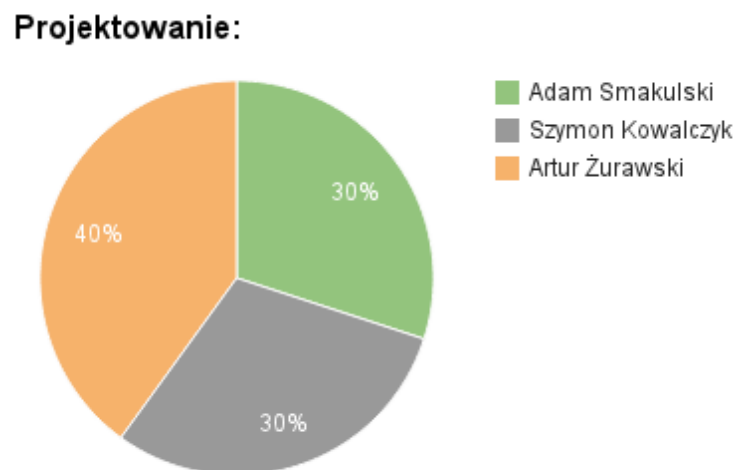
Rysunek 7.4: Analiza: Rozkład pracy

7.3.2.2 Implementacja



Rysunek 7.5: Implementacja: Rozkład pracy

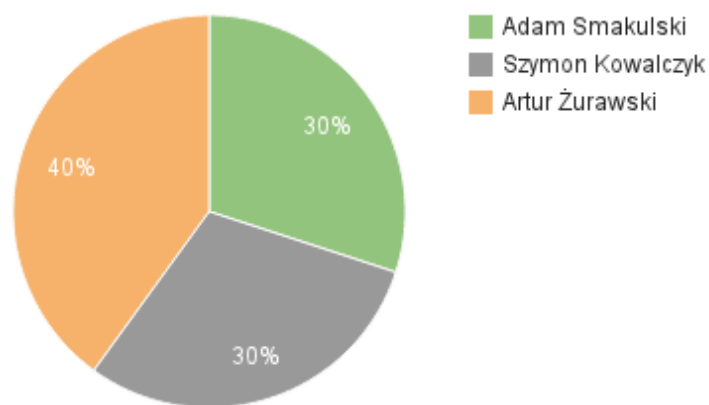
7.3.2.3 Projektowanie



Rysunek 7.6: Projektowanie: Rozkład pracy

7.3.2.4 Testowanie

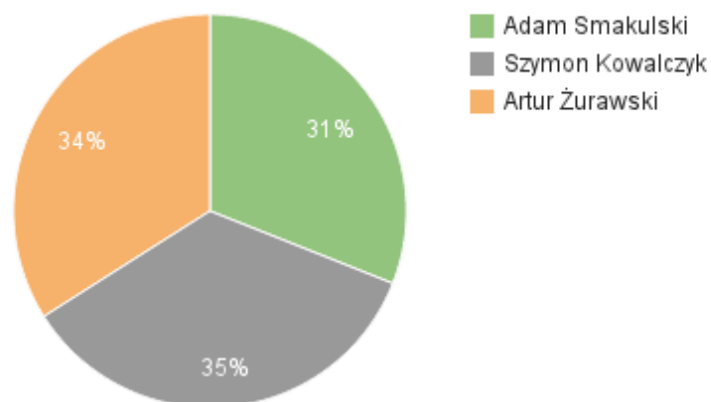
Testowanie:



Rysunek 7.7: Testowanie: Rozkład pracy

7.3.2.5 Rozkład całkowity prac

Rozkład całkowity:



Rysunek 7.8: Całkowity rozkład pracy

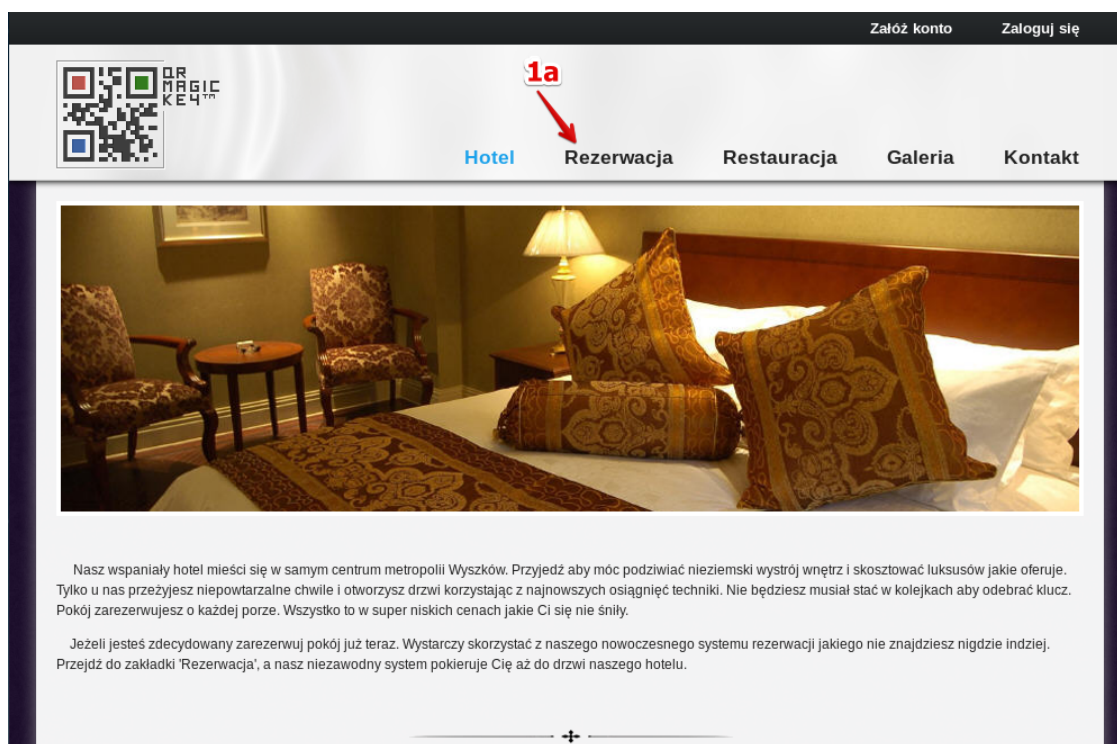
Rozdział 8

Prezentacja systemu

Poniżej przedstawiona została prezentacja systemu. Wybrano standardowy scenariusz rezerwacji pokoju.

Na rysunku 8.1 widoczny jest ekran główny systemu.

1a Użytkownik wybiera zakładkę „Rezerwacja”.



Rysunek 8.1: Ekran widoku głównego

Po wybraniu opcji „Rezerwacja” użytkownik zostaje przeniesiony na ekran wyboru parametrów pokoju przedstawionym na rysunku 8.2.

2a Użytkownik wybiera datę przyjazdu

2b Użytkownik wybiera datę wyjazdu

2c Użytkownik wybiera maksymalną liczbę gości

2d Użytkownik przechodzi do kolejnego kroku klikając na przycisk „Kontynuuj”

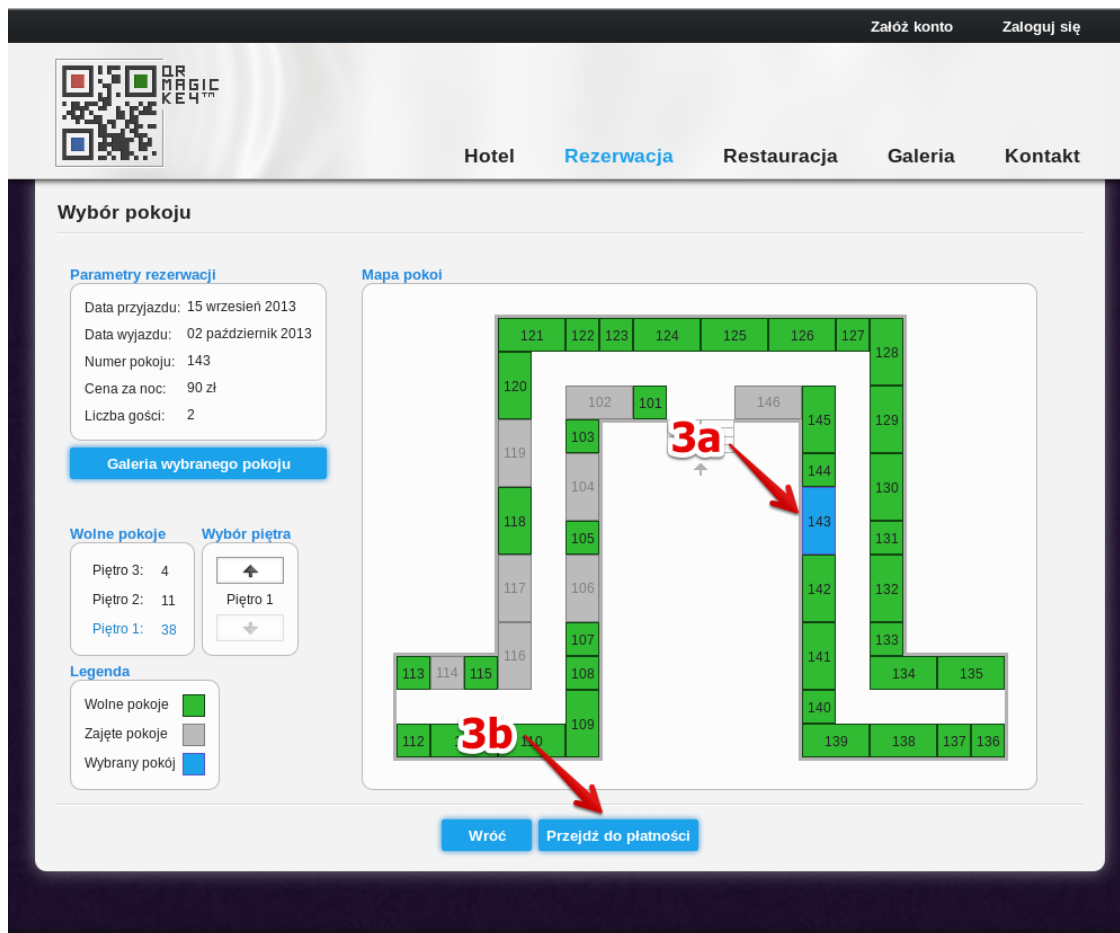
The screenshot shows a web application interface for room reservations. At the top, there is a navigation bar with links: "Załącz konto", "Zaloguj się", "Hotel", "Rezerwacja" (highlighted in blue), "Restauracja", "Galeria", and "Kontakt". Below the navigation bar is a QR code and the text "QR MAGIC KEY". The main content area is titled "Wybór parametrów rezerwacji". It contains two sections: "Wybór daty" and "Dodatkowe parametry". In the "Wybór daty" section, there are two date pickers: "Data przyjazdu:" (set to 15 wrzesień 2013) and "Data wyjazdu:" (set to 02 październik 2013). In the "Dodatkowe parametry" section, there are two rows: "Maksymalna liczba gości:" with a dropdown menu (set to 2) and a checkbox "Włącz" (checked), and "Maksymalna cena:" with a dropdown menu (set to 2) and a checkbox "Włącz" (unchecked). At the bottom of the form is a blue button labeled "Kontynuuj". Red arrows and labels point to specific elements: "2a" points to the arrival date picker, "2b" points to the departure date picker, "2c" points to the "Maksymalna liczba gości:" dropdown, and "2d" points to the "Kontynuuj" button.

Rysunek 8.2: Ekran wyboru parametrów rezerwacji

Użytkownik zostaje przeniesiony na ekran mapy pokoi, przedstawiony na rysunku 8.3, w celu wybrania lokalizacji pokoju.

3a Użytkownik wybiera interesujący go pokój poprzez kliknięcie na niego na mapie

3b Użytkownik klika przycisk „Przejdź do płatności”

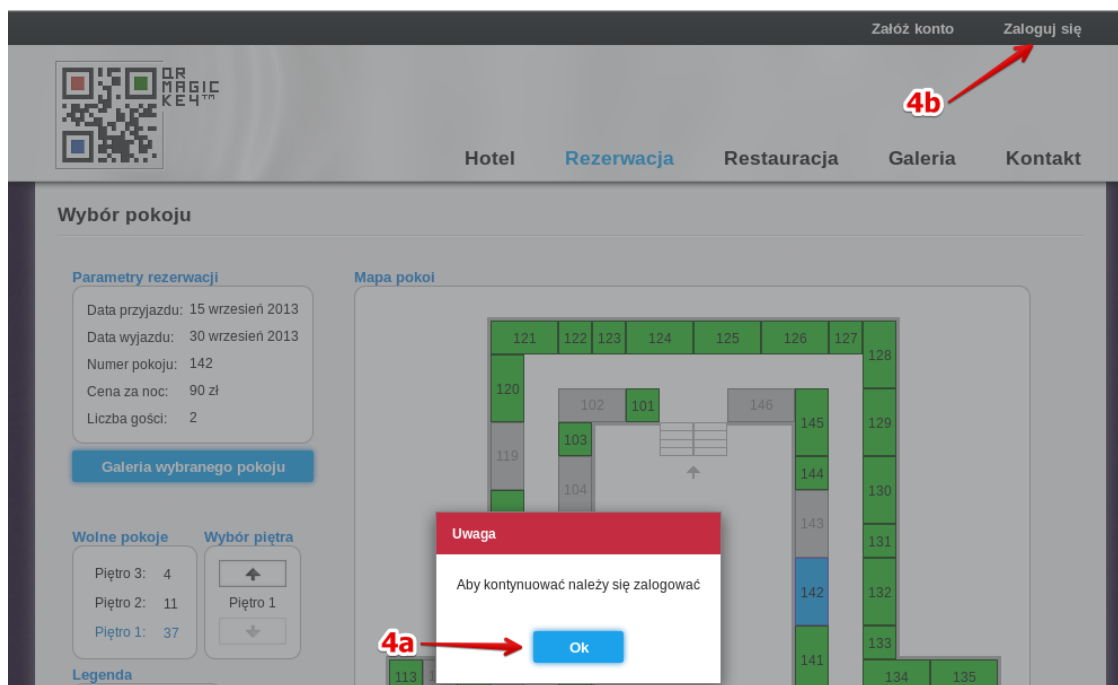


Rysunek 8.3: Ekran wyboru pokoju

W wyniku tego, iż użytkownik nie był zalogowany, pokazuje się okienko z informacją „Aby kontynuować należy się zalogować” z możliwością zamknięcia przyciskiem „OK”.

4a Użytkownik klika na przycisk „OK” w celu zamknięcia okienka z informacją

4b Użytkownik wybiera opcję „Zaloguj się”



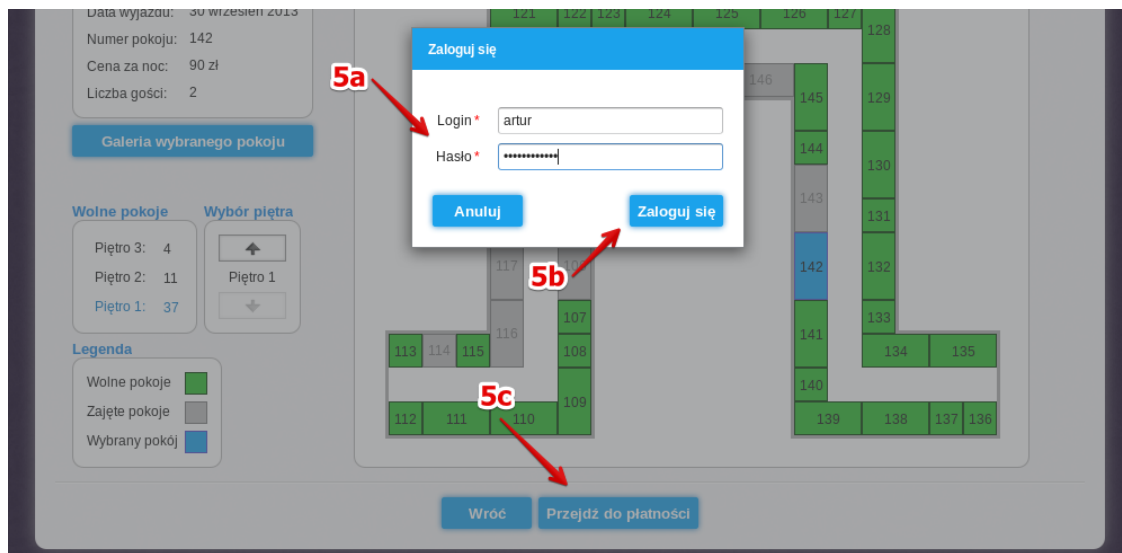
Rysunek 8.4: Ekran informujący o konieczności logowania

Ekran logowania

5a Użytkownik wprowadza dane logowania

5b Użytkownik zatwierdza wprowadzone dane wybierając przycisk „Zaloguj się”

5c Użytkownik zatwierdza wybór pokoju wybierając przycisk „Przejdź do płatności”



Rysunek 8.5: Ekran logowania

Użytkownik zostaje przekierowany na ekran płatności.

6a Użytkownik wybiera jedną z możliwych form płatności

6b Użytkownik wprowadza dane karty płatniczej

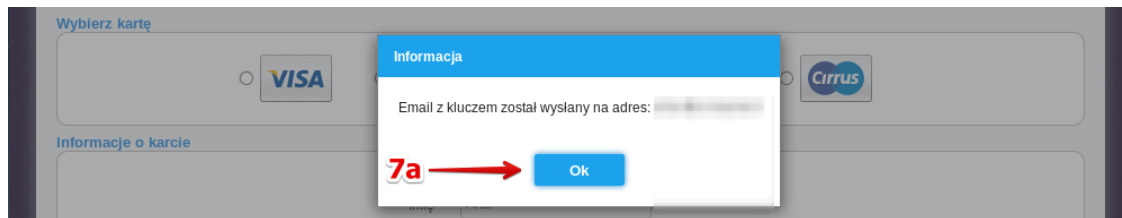
6c Użytkownik potwierdza płatność wybierając przycisk „Potwierdź”

The screenshot shows a web application interface for a payment screen. At the top, there is a QR code labeled 'QR MAGIC KEY™' and a navigation bar with links: Hotel, Rezerwacja, Restauracja, Galeria, and Kontakt. The user is logged in as 'Witaj Artur'. The main section is titled 'Płatność' and contains three sub-sections: 'Podsumowanie', 'Wybierz kartę', and 'Informacje o karcie'. The 'Podsumowanie' section displays the trip details: Data przyjazdu: 15 wrzesień 2013 : 15:00, Data wyjazdu: 02 październik 2013 : 11:00, Numer pokoju: 143, and Koszt pobytu: 1530 zł (90zł. / doba). The 'Wybierz kartę' section shows five payment method options: VISA, VISA Electron (selected), Maestro, AMERICAN EXPRESS, and Cirrus. The 'Informacje o karcie' section contains form fields for: Imię (Artur), Nazwisko (Żurawski12, with a tooltip 'Nazwisko może zawierać tylko litery alfabetu'), Numer karty (6660006660009999999), Data ważności karty (01 / 2017, with a dropdown for 'miesiąc / rok'), and Kod CVV2 / CVC2 / CAV2 (1234, with a note '* (3 lub 4 cyfry z tyłu Twojej karty)'). A red arrow labeled '5a' points to the VISA Electron option. A red arrow labeled '5b' points to the Nazwisko field. A red arrow labeled '5c' points to the 'Potwierdź' button at the bottom of the form. The footer contains a grid of links: Hotel, Vaadin, Perl, Tomcat, JavaScript; Informacje, Restauracja, MySQL, Kontakt, Galeria; Regulamin, Galeria, JavaScript, Rezerwacja, Hotel; QR Code, PJWSTK Gdansk, HTML, Regulamin, Home; and Raspberry Pi, Java, GIMP, Vaadin.

Rysunek 8.6: Ekran płatności

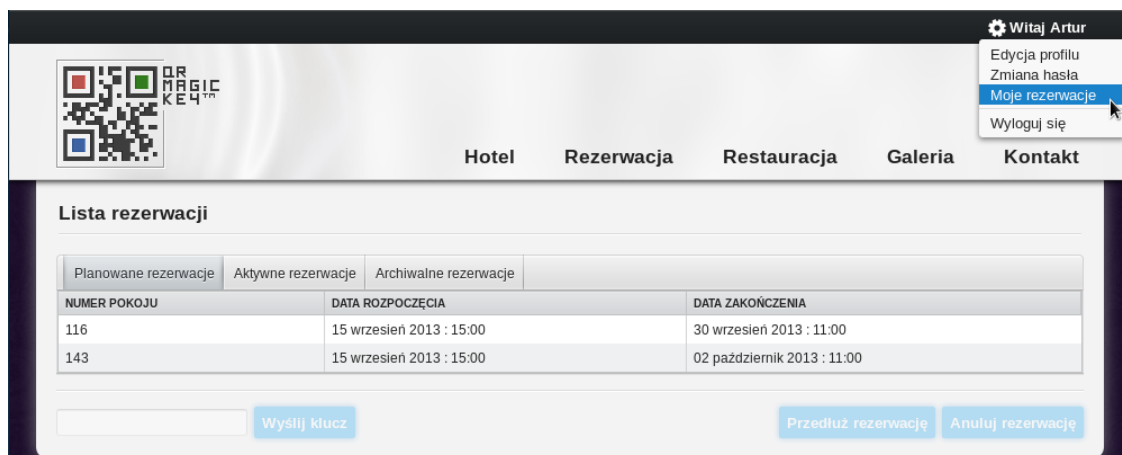
Użytkownik zostaje poinformowany o wysłaniu kodu na adres podany przy rejestracji konta..

7a Użytkownik zamyka okno wybierając przycisk „OK”.



Rysunek 8.7: Ekran potwierdzenia wysłania kodu QR

Użytkownik zostaje przekierowany na poniższy ekran „Lista rezerwacji” na której przedstawiona jest lista jego planowanych rezerwacji.



Rysunek 8.8: Ekran listy rezerwacji

Rozdział 9

Wkład własny - Artur Żurawski

Ten rozdział opisuje najważniejsze elementy mojego wkładu w pracę dyplomową. Wszystkie poniżej opisane elementy są mojego autorstwa.

Hardware: Podczas prac nad projektem QR Magic Hotel moim głównym zadaniem była praca nad czytnikiem kodów QR. Czasochłonnym zajęciem było przeanalizowanie dostępnego na rynku hardware'u i software'u pod kątem przydatności w projekcie. Dostępne na rynku elementy często były niewyposażone w wymagane interfejsy, posiadały niską jakość wykonania lub wysoką cenę, nie adekwatną do oferowanych możliwości. Ostatecznie wybrałem komputer Raspberry Pi który był projektem bardzo obiecującym ze względu na pracę pod kontrolą otwartego i uniwersalnego systemu Linux, mnogość potencjalnych zastosowań i niewielką cenę, co skutkowało dużą popularnością, która w konsekwencji prowadziła do dobrego wsparcia przez pozostałych użytkowników na forach tematycznych.

Perl: Fakt samodzielnej pracy nad czytnikiem kodów posiadał bez wątpienia minus w postaci braku wsparcia przy pracy nad zagadnieniem, ze strony pozostałych członków zespołu. Pozwoliło to jednak na wybór bardziej niszowego, w porównaniu do języka JAVA, języka PERL który nie wzbudzał wielkich emocji ze strony pozostałych członków zespołu, a doskonale wpisywał się w oczekiwania jakie stawiałem względem języka który miał obsługiwać proces odczytu i analizy kodów QR - posiadał bardzo niskie zapotrzebowanie na zasoby sprzętowe, był bardzo responsywny, dużą funkcjonalność programu dało się uzyskać przy pomocy małej ilości kodu i miał dostępne moduły pozwalające na integrację z Raspberry Pi. Z racji kilkuletniego stażu pracy na stanowisku Administratora sieci i systemów Linux / UNIX, nie bez znaczenia był również fakt iż widziałem potencjalne zastosowania dla języka Perl w pracy zawodowej, który jest bez wątpienia o wiele częściej wykorzystywany przez Administratorów w porównaniu do wcześniej wspomnianego języka Java. Przypuszczenia okazały się trafne, ponieważ obecnie często piszę programy w Perlu, które ułatwiają mi pracę przy codziennej analizie dużej ilości logów systemowych, automatyzacji pracy czy wykonywaniu narzędzi wspierających pracę Administratora,

a jest to niewątpliwą konsekwencją udziału przy wspomnianym elemencie pracy dyplomowej.

Moduły: Poza kamerą, ważnym interfejsem Raspberry Pi był port GPIO, który pozwolił na użycie stworzonego przeze mnie modułu diody RGB. Biblioteki Perla pozwoliły na sterowanie sygnałami wysyłanymi na port GPIO, co w konsekwencji prowadziło do zapalania się innego koloru diody w zależności wyniku walidacji kodu QR lub stanu połączenia z bazą danych.

Na poniższym kodzie zaprezentowana jest funkcja odpowiedzialna za inicjalizację obsługi odpowiednich pinów portu GPIO oraz podawanie napięcia na poszczególne anody diody RGB celem jej świecenia w odpowiednim dla zdarzenia kolorze:

```
1 Device::BCM2835::init() || die "Could not init library";
2 Device::BCM2835::gpio_fsel(Device::BCM2835::RPI_GPIO_P1_12, Device::BCM2835::
  BCM2835_GPIO_FSEL_OUTP);
3 Device::BCM2835::gpio_fsel(Device::BCM2835::RPI_GPIO_P1_16, Device::BCM2835::
  BCM2835_GPIO_FSEL_OUTP);
4 Device::BCM2835::gpio_fsel(Device::BCM2835::RPI_GPIO_P1_18, Device::BCM2835::
  BCM2835_GPIO_FSEL_OUTP);
5
6 # [...]
7
8 sub blink {
9   my $count=$_[1];
10
11   my $red_led = sub {
12     Device::BCM2835::gpio_write(Device::BCM2835::RPI_GPIO_P1_12, 1);
13     Device::BCM2835::delay(300);
14     Device::BCM2835::gpio_write(Device::BCM2835::RPI_GPIO_P1_12, 0);
15     Device::BCM2835::delay(300);
16   };
17   my $green_led = sub {
18     Device::BCM2835::gpio_write(Device::BCM2835::RPI_GPIO_P1_16, 1);
19     Device::BCM2835::delay(300);
20     Device::BCM2835::gpio_write(Device::BCM2835::RPI_GPIO_P1_16, 0);
21     Device::BCM2835::delay(300);
22   };
23   my $blue_led = sub {
24     Device::BCM2835::gpio_write(Device::BCM2835::RPI_GPIO_P1_18, 1);
25     Device::BCM2835::delay(300);
26     Device::BCM2835::gpio_write(Device::BCM2835::RPI_GPIO_P1_18, 0);
27     Device::BCM2835::delay(300);
28   };
29   if ( $_[0] eq "r" ) {
30     until ($count eq 0) {
31       if ($debug) {print "Blink RED\n";}
32       $count--;
33       $red_led->();
34     }
35   }
36   elsif ( $_[0] eq "g" ) {
37     until ($count eq 0) {
38       if ($debug) {print "Blink GREEN\n";}
39       $count--;
40       $green_led->();
41     }
42   }
43   elsif ( $_[0] eq "b" ) {
44     until ($count eq 0) {
45       if ($debug) {print "Blink BLUE\n";}
46     }
47   }
48 }
```

```

45         $count--;
46         $blue_led->();
47     };
48 }
49 elseif ( $_[0] eq "rgb" ) {
50     until ( $count eq 0 ) {
51         if ( $debug ) {print "Blink RGB\n"};
52         $count--;
53         $red_led->();
54         $green_led->();
55         $blue_led->();
56     };
57 }
58 };

```

Integracja z bazą danych aplikacji: Aby umożliwić integrację z resztą systemu, czytnik musiał komunikować się z bazą danych porównując odczytane kody QR z wartościami zapisanymi w bazie. Nie byłoby to możliwe bez odpowiednich modułów i odpowiednio skonstruowanego zapytania. Poniżej przedstawiona funkcja db_query odpytuje bazę danych uwzględniając fakt innego zapytania dla kodu QR gościa hotelu i osoby sprzątającej.

```

1 sub db_query {
2     my ($query_room,$query_cleaner,$db_sock, $db_resp);
3     $query_cleaner = "
4         select * from
5         QR_MASTER_KEY
6         INNER JOIN
7         QR_MASTER_KEY_2_ROOM
8         ON QR_MASTER_KEY.ID=
9         QR_MASTER_KEY_2_ROOM.MASTER_KEY_ID
10        WHERE QR_MASTER_KEY.ID='$_[0]'
11        AND QR_MASTER_KEY.HASH='$_[1]'
12        AND now() between QR_MASTER_KEY.START_DATE and QR_MASTER_KEY.
13        END_DATE
14        AND QR_MASTER_KEY_2_ROOM.ROOM_ID='$reader_num'
15        ";
16     $query_room = "
17         select * from QR_RESERVATION
18         where ID='$_[0]'
19         AND HASH='$_[1]'
20         AND ROOM_ID='$reader_num'
21         AND now() between START_DATE and END_DATE
22        ";
23     $db_sock = DBI->connect("DBI:mysql:database=$sql_db;host=$sql_h", $sql_u, $sql_p) || print "
24     Could not connect to database: $DBI::errstr" && return "2";
25     $db_resp = $db_sock->do($query_cleaner);
26     if ( $debug ) {print "-DB- Cleaner DB query returned $db_resp\; $_[0] $_[1] $reader_num \n"};
27     if ( $db_resp ge 1 ) {
28         $db_sock->disconnect();
29         return "0";
30     } else {
31         $db_resp = $db_sock->do($query_room);
32         $db_sock->disconnect();
33         if ( $debug ) {print "-DB- Room DB query returned $db_resp\; $_[0] $_[1] $reader_num \n"};
34         if ( $db_resp ge 1 ) {
35             return "0";
36         } else {
37             return "1";
38         };
39     };
40 }

```

```
36 };  
37 };
```

Raspberry Pi - system operacyjny: Komputer Raspberry Pi nie mógłby funkcjonować bez odpowiednio przygotowanego systemu operacyjnego. W związku z założeniem, iż Raspberry Pi nie będzie podłączone do monitora, podczas startu komputera program do sprawdzania kodów QR musi zostać automatycznie uruchomiony. Osiągnąłem to poprzez napisanie skryptu daemona systemowego uruchamiającego program poprzez sesję programu SCREEN. Sesja programu SCREEN daje możliwość podłączenia się do działającego programu celem bardziej dogłębnej analizy jego zachowania, przy włączonym trybie debug. Poniżej prezentuje skrypt inicjujący integrujący się z systemem operacyjnym, napisanym w języku Bash:

```
1  #!/bin/bash  
2  ### BEGIN INIT INFO  
3  # Provides:          qr_scanner  
4  # Required-Start:    $network  
5  # Required-Stop:     $network  
6  # Default-Start:     2 3 4 5  
7  # Default-Stop:      0 1 6  
8  # Short-Description: Start/stop qr daemon  
9  ### END INIT INFO  
10  
11 #Ustawienia zmiennych  
12  
13 USER=root # Uruchom program jako  
14 QR_SCANNER=/etc/qr_reader/qr_scanner.pl # Lokalizacja programu  
15 SCREEN=/usr/bin/screen # Lokalizacja binarki programu screen  
16 SCREEN_NAME=qr # Nazwa sesji screen  
17 PIDFILE=/var/run/qr_scanner.pid # pidfile  
18  
19 start() {  
20     start-stop-daemon --start --background --oknode \  
21         -- pidfile "$PIDFILE" --make-pidfile \  
22         --chuid $USER \  
23         --exec $SCREEN -- -DmUS $SCREEN_NAME $QR_SCANNER  
24     if [[ $? -ne 0 ]]; then  
25         echo -e "QR Scanner has failed to start \t\t [\e[0;31m FAIL \e[0m] \n"  
26         exit 1  
27     fi  
28     echo -e "QR Scanner started successfully \t [\e[0;32m OK \e[0m] \n"  
29 }  
30  
31 stop() {  
32     start-stop-daemon --stop -- pidfile "$PIDFILE"  
33     if [[ $? -ne 0 ]]; then  
34         echo -e "QR Scanner has failed to stop \t\t [\e[0;31m FAIL \e[0m] \n"  
35         exit 1  
36     fi  
37     echo -e "QR Scanner stopped successfully \t [\e[0;32m OK \e[0m] \n"  
38 }  
39  
40  
41 case "$1" in  
42     start)  
43         start  
44     ;;  
45     stop)
```

```

46         stop
47     ;;
48     restart )
49         stop;
50     sleep 2;
51     start
52     ;;
53     *)
54         echo "Usage: $0 {start|stop|restart}"
55     ;;
56 esac
57
58 exit 0

```

L^AT_EX: Dokumentacja pracy dyplomowej w całości została napisana w języku L^AT_EX. Treść pracy była wspólnie tworzona przez wszystkich członków zespołu, jednak ja byłem odpowiedzialny za przeniesienie treści dokumentacji pracy dyplomowej na język L^AT_EX. Głównym bodźcem do podjęcia takiej decyzji była chęć stworzenia profesjonalnie wyglądającego dokumentu. Bezpłatne edytory tekstowe obligowały do poświęcenia dużej uwagi na formatowanie pracy, jednocześnie nie zapewniając satysfakcjonujących nas wizualnych efektów końcowych. Uważam, że uzyskany efekt wizualny pracy jest w pełni zadowalający. Mam nadzieję, że czytelnik jest ze mną zgodny w tej kwestii. L^AT_EXpo początkowym ustaleniu sposobu formatowania dokumentu pozwalał się potem już tylko skupić na jego treści. Proste funkcje warunkowe pozwoliły także na utworzenie różnych wersji jednego dokumentu w zależności od autora pracy. Dokumenty L^AT_EXprzed kompilacją do plików PDF mają formę „czystego” tekstu. Fakt ten wpłynął na wersjonowanie dokumentu z wykorzystaniem repozytoriów GIT, co pozwoliło mi dzielić się efektami pracy z pozostałymi członkami zespołu. Poniżej prezentuję wykonaną przeze mnie część konfiguracji dokumentu:

```

1
2 \documentclass[12pt,oneside,hidelinks]{memoir}
3 \usepackage{fullpage}
4
5 %\newif\ifartur
6 %\newif\ifszymon
7 %\newif\ifadam
8
9 % Wyświetlanie odpowiedniego tekstu w zależności od autora pracy. Odkomentować tylko jedno:
10 %\szymontrue
11 %\arturtrue
12 %\adamtrue
13
14 \usepackage[indentfirst]{%wcięcie dla nowego paragrafu}
15 \setlength{\headheight}{14pt}
16 \usepackage{polski}
17 \usepackage[polish]{babel}
18 \usepackage[utf8]{inputenc}
19 \usepackage[T1]{fontenc}
20 \usepackage[variablett]{lmodern} %font do kodu
21 \usepackage{graphicx} %Obrazki
22
23 %% Nie dzieli wyrazów
24 \tolerance=1

```

```

25 \emergencystretch=\maxdimen
26 \hyphenpenalty=10000
27 \hbadness=10000
28
29 \usepackage{float} %Oplywanie obazkow
30 \usepackage{hyperref} %Klikalny TOC w PDF
31
32 \usepackage{multirow} %Wlasciwosci dokumentu PDF
33 \hypersetup{
34     pdftitle={Praca Inzynierska QR Magic Hotel},
35     pdfauthor={Kowalczyk, Smakulski, Zurawski},
36     pdfsubject={Dokumentacja},
37     pdfkeywords={QR Magic Hotel},
38     bookmarksnumbered=true,
39     bookmarksopen=true,
40     bookmarksopenlevel=2,
41     % colorlinks=true,
42     pdfstartview=Fit,
43     pdfpagemode=UseOutlines,
44 }
45
46 % [...]
47
48 \lstdefinestyle {BashInputStyle}{ %styl dla fragmentow kodu lini polecen
49     language=bash,
50     basicstyle=\scriptsize\ttfamily,
51     numbers=left,
52     numberstyle=\tiny,
53     numbersep=3pt,
54     frame=tb,
55     columns=fullflexible,
56 }
57
58 \usepackage{wrapfig} % otaczanie obrazkow
59
60 \usepackage{font=footnotesize,labelfont=bf}{caption} %male podpisy pod tabelami i obrazami
61
62 % Dzielenie tabel
63 \newcolumntype{L}[1]{>{\raggedright\let\newline\\\arraybackslash\hspace{0pt}}m{#1}}
64 \newcolumntype{C}[1]{>{\centering\let\newline\\\arraybackslash\hspace{0pt}}m{#1}}
65 \newcolumntype{R}[1]{>{\raggedleft\let\newline\\\arraybackslash\hspace{0pt}}m{#1}}
66
67 \newcommand{\scellp}[2][c]{\begin{tabularx}{.475\textwidth}{#1}{@{}X@{}}#2\end{tabularx}} %Laczony
68     multicolumn z podzialem linii
69 \newcommand{\scell}[2][c]{\begin{tabularx}{.97\textwidth}{#1}{@{}X@{}}#2\end{tabularx}} %Laczony
70     multicolumn na szerokosc strony
71
72 \setsecnumdepth{subparagraph} %Jak gleboko numerowac paragrafy
73 \setcounter{tocdepth}{5} %jak gleboko pisac praragrafy na TOC
74
75 \usepackage{paralist} %Male odstepy przy wypunktowaniu
76
77 % [...]

```

Testy funkcjonalne: Odpowiedzialny byłem również, za przeprowadzenie testów funkcjonalnych aplikacji. Wyniki mojej pracy w tym zakresie można zobaczyć na stronie 88.

Pozostałe: Poza wyżej wymienionymi, odpowiedzialny byłem również za przygotowanie środowiska serwera wirtualnego, wraz z konfiguracją VPN umożliwiającą pracę wszystkich członków zespołu na „jednym ekranie”. Dało to możliwość wspólnego

analizowania efektów pracy. Ponadto byłem odpowiedzialny za przygotowanie dostępnej dla wszystkich członków zespołu, odpowiednio zabezpieczonej bazy danych, z początku na dedykowanym serwerze wirtualnym, a z czasem na przeznaczonym do tego hostingu. Dało to możliwość wspólnej pracy na jednej bazie danych ułatwiając analizę wspólnej pracy niezależnie przez nas tworzonych elementów systemu.

Rozdział 10

Wkład własny - Szymon Kowalczyk

W trakcie prac nad projektem moim głównym zadaniem było stworzenie większości modułów aplikacji internetowej. Przed przystąpieniem do implementacji poszczególnych funkcjonalności zawartych w specyfikacji wymagań, byłem odpowiedzialny za stworzenie szkieletu aplikacji, który pozwolił w późniejszym etapie w prosty sposób dodawać kolejne widoki oraz umożliwił nawigację między nimi. W tym celu wykorzystany został wzorzec projektowy MVP oraz framework Vaadin, z którym dane mi było wstępnie zapoznać się w trakcie zajęć prowadzonych przez dr. Jakuba Neumanna w semestrze VII. Chęć lepszego poznania tego frameworku była głównym powodem jego wyboru.

Od początku pisania pracy wspólnie z kolegami z zespołu brałem czynny udział w fazach analizy i projektowania, a także w powstawaniu dokumentacji projektu.

Wśród moich obowiązków znalazło się także stworzenie niektórych grafik wykorzystanych do poprawienia strony wizualnej aplikacji internetowej, dzięki czemu zdobyłem pewne doświadczenie w pracy z programem do tworzenia i obróbki grafiki jakim jest GIMP.

Choć została mi przydzielona rola głównego programisty nie oznaczało to, że jestem w tej dziedzinie ekspertem. Od początku byłem nastawiony na zdobywanie nowej wiedzy oraz rozwój w kierunku jakim jest programowanie.

Poniższa lista opisuje główne moduły aplikacji internetowej wykonane przeze mnie:

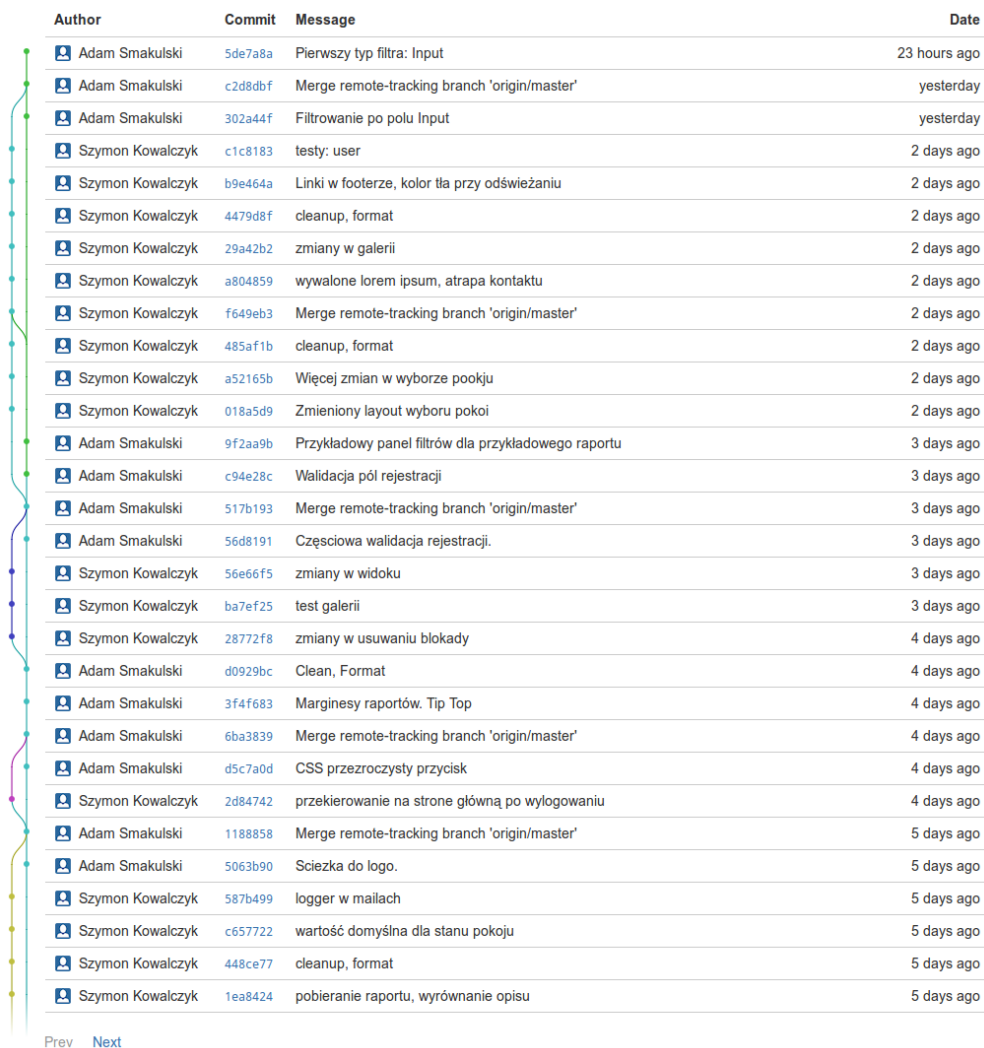
- Moduł rezerwacji pozwalający na wyszukanie oraz zaprezentowanie na mapie pokoi, zgodnie z parametrami wybranymi przez użytkownika. Celem było stworzenie funkcjonalności pozwalającej użytkownikowi zdecydować nie tylko kiedy, ale także w której części hotelu chciałby przebywać.
- Moduł osoby sprzątajacej, który maksymalnie uprościł czynności jakie dana osoba musi wykonać, aby dostać się do odpowiednich pokoi. Moduł ten pozwolił także na zapisywanie informacji niezbędnych dla managera, w celu ustalenia kto i kiedy odpowiedzialny był za posprząatanie danego pokoju.
- Moduł zarządzania rezerwacjami przez gości hotelowego pozwalający na modyfikację rezerwacji już po ich opłaceniu. Dzięki niemu użytkownik ma możliwość anulowania lub

przedłużenia rezerwacji, a także podzielenia się kluczem do pokoju z innymi korzystając z funkcji ponownego wysłania kodu na wybrany adres e-mail.

- Panel gościa hotelowego udostępniający możliwości takie jak edycja profilu oraz zmiana hasła przez użytkownika.
- Panel administratora dzięki, któremu osoba o odpowiednich uprawnieniach będzie w stanie zarządzać kontami użytkowników poprzez zmianę uprawnień oraz edycję niektórych danych. Dodatkowo administrator może dodawać do systemu nowe pokoje określając jednocześnie ich pozycję na mapie.

Dodatkowe materiały obrazujące mój udział w powstawaniu systemu:

1. Na początek typowy log z serwisu bitbucket.org przeznaczonego dla używanego przez nas systemu kontroli wersji GIT. Jasno widać pełną współpracę oraz częstotliwość zmian wprowadzanych zarówno przeze mnie jak i Adama Smakulskiego pełniącego rolę drugiego programisty.



Author	Commit	Message	Date
Adam Smakulski	5de7a8a	Pierwszy typ filtra: Input	23 hours ago
Adam Smakulski	c2d8dbf	Merge remote-tracking branch 'origin/master'	yesterday
Adam Smakulski	302a44f	Filtrowanie po polu Input	yesterday
Szymon Kowalczyk	c1c8183	testy: user	2 days ago
Szymon Kowalczyk	b9e464a	Linki w footerze, kolor tła przy odświeżaniu	2 days ago
Szymon Kowalczyk	4479d8f	cleanup, format	2 days ago
Szymon Kowalczyk	29a42b2	zmiany w galerii	2 days ago
Szymon Kowalczyk	a804859	wywalone lorem ipsum, atrapa kontaktu	2 days ago
Szymon Kowalczyk	f649eb3	Merge remote-tracking branch 'origin/master'	2 days ago
Szymon Kowalczyk	485af1b	cleanup, format	2 days ago
Szymon Kowalczyk	a52165b	Więcej zmian w wyborze pokoi	2 days ago
Szymon Kowalczyk	018a5d9	Zmieniony layout wyboru pokoi	2 days ago
Adam Smakulski	9f2aa9b	Przykładowy panel filtrów dla przykładowego raportu	3 days ago
Adam Smakulski	c94e28c	Walidacja pól rejestracji	3 days ago
Adam Smakulski	517b193	Merge remote-tracking branch 'origin/master'	3 days ago
Adam Smakulski	56d8191	Częściowa walidacja rejestracji.	3 days ago
Szymon Kowalczyk	56e66f5	zmiany w widoku	3 days ago
Szymon Kowalczyk	ba7ef25	test galerii	3 days ago
Szymon Kowalczyk	28772f8	zmiany w usuwaniu blokady	4 days ago
Adam Smakulski	d0929bc	Clean, Format	4 days ago
Adam Smakulski	3f4f683	Marginesy raportów. Tip Top	4 days ago
Adam Smakulski	6ba3839	Merge remote-tracking branch 'origin/master'	4 days ago
Adam Smakulski	d5c7a0d	CSS przezroczysty przycisk	4 days ago
Szymon Kowalczyk	2d84742	przekierowanie na strone główną po wylogowaniu	4 days ago
Adam Smakulski	1188858	Merge remote-tracking branch 'origin/master'	5 days ago
Adam Smakulski	5063b90	Ścieżka do logo.	5 days ago
Szymon Kowalczyk	587b499	logger w mailach	5 days ago
Szymon Kowalczyk	c657722	wartość domyślna dla stanu pokoju	5 days ago
Szymon Kowalczyk	448ce77	cleanup, format	5 days ago
Szymon Kowalczyk	1ea8424	pobieranie raportu, wyrównanie opisu	5 days ago

Prev Next

Rysunek 10.1: Zrzut ekranu z serwisu bitbucket.org

2. 2. Poniżej znajduje się statyczna metoda z klasy Confirmation. Jest to przykład pokazujący, że nawet porządny framework może posiadać pewne braki. Vaadin w chwili powstawania aplikacji nie dostarczał żadnego komponentu odpowiedzialnego za wyświetlanie okna z powiadomieniem, dającego użytkownikowi możliwość wyboru tak/nie. Mimo, iż w trakcie pisania pracy zdecydowaliśmy się na zmianę wersji na nowszą, nie rozwiązało to problemu braku tego komponentu. Ponieważ jest on często wykorzystywany w całej aplikacji postanowiłem sam napisać klasę, która da nam do dyspozycji wspomniany element bez konieczności powtarzania tego samego kodu za każdym razem gdy zajdzie potrzeba skorzystania z okna modalnego. Kod powstał w taki sposób aby łatwo dało się rozpoznać oraz zareagować na podjętą przez użytkownika decyzję.

```

1 public static void showDouble(final ConfirmationType type, final String question, final Confirmation.Callback
2     callback) {
3     Confirmation.show(type, question, callback);
4
5     Confirmation.yesButton = new NativeButton(Caption.YES);
6     Confirmation.yesButton.setStyleName(Style.QRMK_BUTTON);
7     Confirmation.yesButton.addClickListener(new ClickListener() {
8         private static final long serialVersionUID = -6371729101936110690L;
9
10        @Override
11        public void buttonClick(final ClickEvent event) {
12            UI.getCurrent().removeWindow(Confirmation.window);
13            Confirmation.callback.onDialogResult(true);
14        }
15    });
16
17    Confirmation.noButton = new NativeButton(Caption.NO);
18    Confirmation.noButton.setStyleName(Style.QRMK_BUTTON);
19    Confirmation.noButton.addClickListener(new ClickListener() {
20
21        private static final long serialVersionUID = -6371729101936110690L;
22
23        @Override
24        public void buttonClick(final ClickEvent event) {
25            UI.getCurrent().removeWindow(Confirmation.window);
26            Confirmation.callback.onDialogResult(false);
27        }
28    });
29
30    final HorizontalLayout buttons = new HorizontalLayout();
31    buttons.setSizeUndefined();
32    buttons.setSpacing(true);
33    buttons.addComponents(Confirmation.noButton, Confirmation.yesButton);
34
35    Confirmation.layout.addComponents(Confirmation.label, Labels.br(), buttons);
36    Confirmation.layout.setComponentAlignment(buttons, Alignment.MIDDLE_CENTER);
37
38 }

```

3. Kolejny wycinek kodu będący fragmentem prezentera reagującego na wybór parametrów rezerwacji przez użytkownika. Wyraźnie pokazuje rolę prezentera w modelu MVP będącego pośrednikiem między logiką aplikacji, a widokiem. Dodatkowo ukazana jest walidacja dat, której implementacja poprzedzona została stworzeniem przeze mnie odpowiedniego schematu blokowego.

```

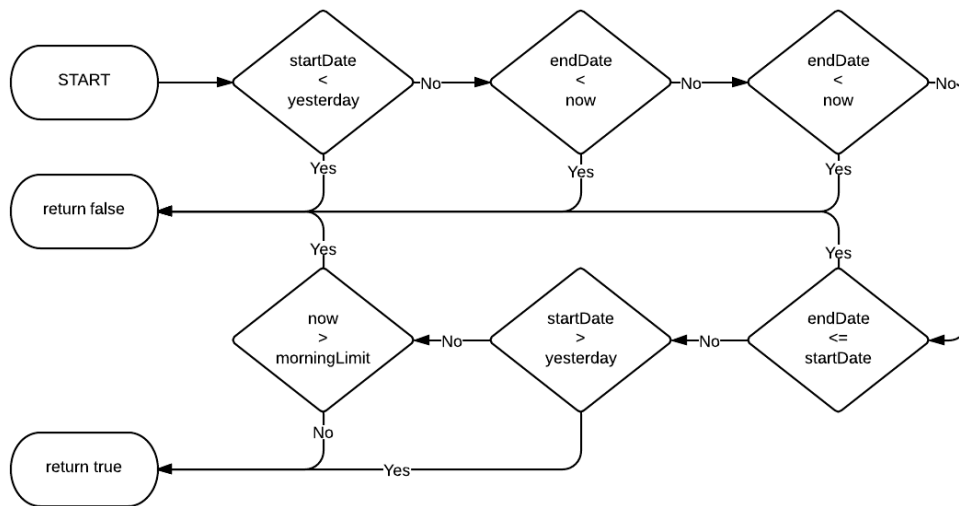
1 @Override
2 public void datesSelected(final DateTime startDate, final DateTime endDate, final Integer capacity, final Integer
3     price) {
4     if (!this.validateDates(startDate, endDate)) {
5         Confirmation.showSingle(ConfirmationType.WARNING, Message.INVALID_EXTEND_DATE);
6         return;
7     }
8
9     this.reservation.setStartDate(Time.getWithStartHour(startDate));
10    this.reservation.setEndDate(Time.getWithEndHour(endDate));
11    this.reservation.setPrice(price);
12    this.reservation.setCapacity(capacity);
13    this.bookingView.showRoomSelectionView();

```

```

14 }
15
16 private boolean validateDates(final DateTime startDate, final DateTime endDate) {
17
18     final DateTime now = DateTime.now();
19     final DateTime yesterday = DateTime.now().minusDays(1);
20
21     if (Time.noon(startDate).isBefore(Time.noon(yesterday)) ||
22         Time.noon(endDate).isBefore(Time.noon(now)) ||
23         !Time.noon(endDate).isAfter(Time.noon(startDate)) ||
24         (!Time.noon(startDate).isAfter(Time.noon(yesterday)) && now.getHourOfDay() > Time.LATE_LIMIT)) {
25         return false;
26     }
27     return true;
28 }

```



Rysunek 10.2: Schemat blokowy walidacji dat

- Innym przykładem mojego wkładu w powstały projekt jest obrazek przedstawiający końcowy etap tworzenia logo systemu w programie GIMP. Jako główny element zastosowany został kod QR z niestandardowymi kolorami oraz tłem. Warto zauważyć, że nawet zmodyfikowany w ten sposób kod nadaje się do odczytu przez aplikacje do tego przeznaczone.



Podsumowując mogę śmiało powiedzieć, że praca nad projektem wiele mnie nauczyła. Przekonałem się, że o sukcesie decyduje nie tylko wiedza, ale także umiejętność pracy w grupie. Przez cały okres trwania projektu starałem się zapoznawać z zadaniami wykonywanymi przez pozostałych członków zespołu, aby nabywać wiedzę na temat mniej znanych mi aspektów informatyki, a także dzielić się własną.

Rozdział 11

Wkład własny- Adam Smakulski

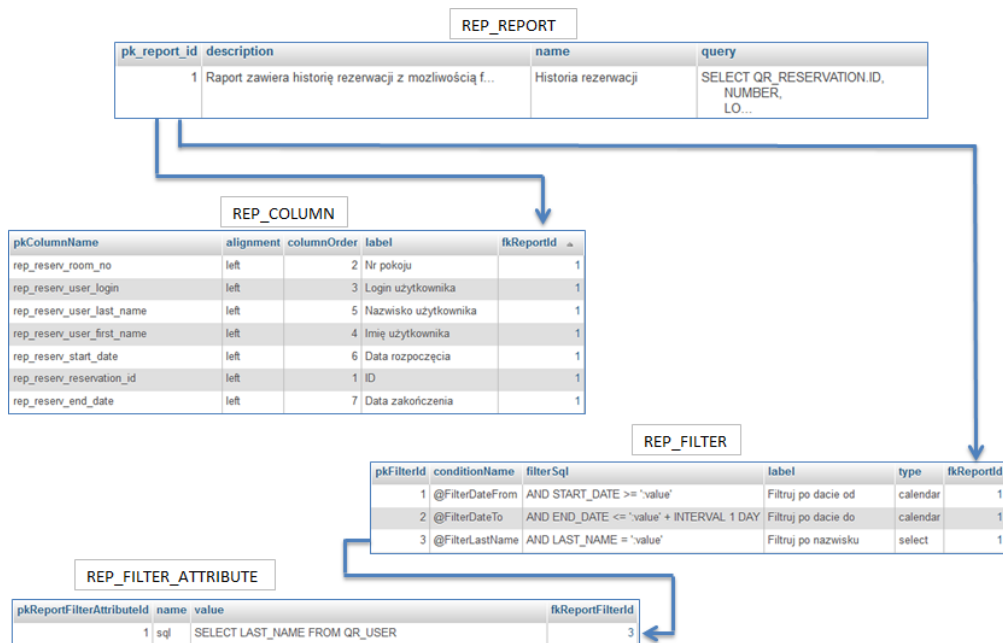
W pierwszym semestrze byłem w głównej mierze zaangażowany w dokumentowanie naszego projektu. Brałem czynny udział w fazie planowania, analizy i projektowania.

Jestem autorem lub współautorem większości diagramów i opisów. Nie przeszkodziło mi to w zapoznawaniu się z pracą kolegów i pogłębianiu wiedzy na tematy mniej mi znane.

W drugim semestrze większy nacisk położyłem na programowanie. Zaprojektowałem i zaimplementowałem moduł raportowy wraz z zapisem raportów do pliku PDF. Temat modułu raportowego zdecydowałem się podjąć w sposób nietuzinkowy. Stworzyłem interfejs bazodanowy, dzięki, któremu, rozbudowa modułu raportowego jest prostsza.

Wszystkie nazwy raportów, opisy raportów, kolumny, filtry oraz atrybuty filtrów są przechowywane w bazie danych w odpowiednich dla siebie tabelach i odpowiadają za sposób generowania ekranu z wynikami poszczególnych raportów. Dzięki takiemu rozwiązaniu definiowanie nowego raportu nie wymaga ingerowania w kod Javy.

Na poniższym schemacie przedstawiam zasadę działania mechanizmu raportowego dla raportu „Historia rezerwacji”:



Rysunek 11.1: Schemat działania na bazie danych

Poniżej opisuję tabele zawarte w bazie danych:

REP_REPORT – tabela zawierająca listę raportów dostępnych w module

REP_COLUMN – tabela zawierająca dane o kolumnach występujących w danym raporcie

REP_FILTER – tabela zawierająca dostępne filtry dla danego raportu w postaci listy wyboru, kalendarza lub pola tekstowego

REP_FILTER_ATTRIBUTE – tabela zawierająca dodatkowe atrybuty nazwa – wartość dla filtra dostępnego w raporcie

Aby przybliżyć działanie modułu raportowego przedstawiam pseudokod Javy:

1. Użytkownik wybiera raport z listy na ekranie
2. W panelu generowane są filtry dla raportu oraz opis raportu
3. W momencie kliknięcia „Generuj”, mechanizm generuje zapytanie raportowe

a) Pobranie podstawowego zapytania raportowego

```
1 SELECT COL1, ..., COLn FROM TABELA WHERE @Parametr1
```

b) Wyszukiwany jest w raporcie filtr, który przypisany jest do parametru o nazwie Parametr1.

c) Dla odnalezionego filtra Parametr1 zamieniany jest na fragment zapytania SQL zawierający warunek dla danego pola w raporcie.

```
1 SELECT COL1, ..., COLn FROM TABELA WHERE COL1 LIKE "%tekst%"
```

4. Utworzone zapytanie raportowe zostaje wywołane i wyniki zostają przekazane na ekran aplikacji.

Przedstawione poniżej elementy panelu raportowego, zaznaczone na czerwono, generowane są dynamicznie na podstawie danych w tabelach raportowych:

Rysunek 11.2: Elementy panelu raportowego

Poza panelem raportowym oprogramowałem walidację danych wprowadzanych przez użytkownika w aplikacji internetowej. Poniżej przedstawiam wycinek kodu odpowiedzialnego za walidację emaila w formularzu rejestracji.

```
1 public void validate(final Object value) throws InvalidValueException {  
2     final String email = (String) value;  
3     if (!email.matches("[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,4}$")) {  
4         throw new InvalidValueException(String.format(Message.INVALID_MAIL_SYNTAX));  
5     }  
6 }
```

Poniżej przedstawiam klasę ValidatorUtil.java, w której są metody pomocnicze wywoływane na potrzeby walidacji pól zabraniających wprowadzenie kolejno liter i znaków:

```
1 public class ValidatorUtil {
2
3     public static boolean containsDigit(final String text) {
4         for (int i = 0; i < text.length(); i++) {
5             if (Character.isDigit(text.charAt(i))) {
6                 return true;
7             }
8         }
9         return false;
10    }
11
12    public static boolean containsLetter(final String text) {
13        for (int i = 0; i < text.length(); i++) {
14            if (Character.isLetter(text.charAt(i))) {
15                return true;
16            }
17        }
18        return false;
19    }
20 }
```

Podsumowując, praca nad dokumentowaniem projektu pozwoliła mi na poszerzenie wiedzy z zakresu Inżynierii Oprogramowania, którą będę mógł wykorzystać w niedawno podjętej pracy zawodowej. Mogłem również wykorzystać i poszerzyć zdobytą podczas studiów znajomość programowania oraz baz danych.

Praca nad projektem nauczyła mnie również pracy w zespole. Pozwoliła mi wypracować sposoby rozwiązywania zadań i problemów w większym, grupowym projekcie.

Spis rysunków

2.1	Rich Picture - Opis problemu i rozwiązanie	4
3.1	Model Procesu Wytwórczego	8
4.1	Diagram przypadków użycia - Aktorzy	22
4.2	Diagram przypadków użycia - Niezalogowany Gość	23
4.3	Diagram przypadków użycia - Zalogowany Gość	24
4.4	Diagram przypadków użycia - Zalogowany Administrator	25
4.5	Diagram przypadków użycia - Zalogowany Manager	26
4.6	Diagram przypadków użycia - Zalogowana Osoba sprzątająca	27
4.7	Diagram sekwencji przebiegu rezerwacji	28
4.8	Diagram Klas	31
4.9	Diagram stanów - Pokój	33
4.10	Diagram stanów - Rezerwacja	34
5.1	Architektura aplikacji	35
5.2	Wzorzec MVC	36
5.3	Wzorzec MVP	37
5.4	Diagram pakietów	38
5.5	Diagram klas pakietu it.pjwstk.qrmk.model	40
5.6	Diagram klas pakietu it.pjwstk.qrmk.model c.d.	41
5.7	Diagram klas pakietu it.pjwstk.qrmk.view	42
5.8	Diagram klas pakietu it.pjwstk.qrmk.view c.d.	43
5.9	Diagram klas pakietu it.pjwstk.qrmk.presenter	44
5.10	Diagram klas pakietu it.pjwstk.qrmk.service	45
5.11	Diagram klas pakietu it.pjwstk.qrmk.util	46
5.12	Diagram sekwencji wyboru parametrów rezerwacji	52
5.13	Diagram sekwencji wyboru pokoju	53
5.14	Diagram sekwencji płatności	54
5.15	Schemat architektury Vaadin	56
5.16	Diagram ERD	58
5.17	Diagram nawigacji dla Administratora	63
5.18	Diagram nawigacji dla Gościa hotelowego	63
5.19	Diagram nawigacji dla Managera	64

5.20	Diagram nawigacji dla Osoby sprzątającej	64
5.21	Szkielet dla ekranu niezalogowanego użytkownika - ekran główny	65
5.22	Szkielet dla ekranu gościa - wybór pokoju	66
5.23	Szkielet dla ekranu managera - raporty	66
5.24	Szkielet dla ekranu managera i administratora - lista pokoi	67
5.25	Szkielet dla ekranu managera - edycja pokoju	67
5.26	Szkielet dla ekranu administratora - edycja pokoju	68
5.27	Szkielet dla ekranu dla zalogowanego i niezalogowanego użytkownika - parametry rezerwacji	68
5.28	Zrzut ekranu prezentujący ekran wyboru pokoju w widoku Gościa hotelowego	69
5.29	Zrzut ekranu prezentujący ekran generowania raportów w widoku Managera	70
5.30	Zrzut ekranu prezentujący ekran listy pokoi w widoku Administratora	71
6.1	Platforma komputerowa Raspberry Pi	76
6.2	Planowanie: Rozkład pracy	80
6.3	Analiza: Rozkład pracy	80
6.4	Projektowanie: Rozkład pracy	81
6.5	Implementacja: Rozkład pracy	81
6.6	Testowanie: Rozkład pracy	82
6.7	Całkowity rozkład pracy	82
7.1	Historia rezerwacji	84
7.2	Historia rezerwacji	85
7.3	Schemat podłączenia diody LED RGB do portów GPIO. Schemat jak również i sam układ został wykonany przez zespół.	88
7.4	Analiza: Rozkład pracy	96
7.5	Implementacja: Rozkład pracy	97
7.6	Projektowanie: Rozkład pracy	97
7.7	Testowanie: Rozkład pracy	98
7.8	Całkowity rozkład pracy	98
8.1	Ekran widoku głównego	99
8.2	Ekran wyboru parametrów rezerwacji	100
8.3	Ekran wyboru pokoju	101
8.4	Ekran informujący o konieczności logowania	102
8.5	Ekran logowania	103
8.6	Ekran płatności	104
8.7	Ekran potwierdzenia wysłania kodu QR	105
8.8	Ekran listy rezerwacji	105
10.1	Zrzut ekranu z serwisu bitbucket.org	115
10.2	Schemat blokowy walidacji dat	117
10.3	Zrzut ekranu z programu GIMP	118

11.1 Schemat działania na bazie danych	120
11.2 Elementy panelu raportowego	121

Dodatek A

Lista plików załączonych na płycie CD

```
./attachments
├─ wstepny_plan_projektu.pdf
├─ dokument_zalozen_wstepnych.pdf
└─ specyfikacja_wymagan_systemowych.pdf
```

Bibliografia

- [1] Włodzimierz Dąbrowski, Kazimierz Subieta, *Podstawy inżynierii oprogramowania*. Wydawnictwo PJWSTK, 2005.
- [2] Michał Śmiałek, *Zrozumieć UML 2.0 - Metody modelowania obiektowego*. Wydawnictwo Helion, 2005.
- [3] Craig Larman, *UML i wzorce projektowe*. Wydawnictwo Helion, 2011.
- [4] Eric Freeman, Kathy Sierra, Bert Bates, *Head First Design Patterns. Edycja polska (Rusz głową!)*. Wydawnictwo Helion, 2005.
- [5] Stanisław Wrycza, Bartosz Marcinkowski, Krzysztof Wyrzykowski, *Język UML 2.0 w modelowaniu systemów informatycznych*. Wydawnictwo Helion, 2005.
- [6] Book of Vaadin *publikacja grupowa z oficjalnej strony vaadin.com*.
- [7] Kathy Sierra, Bert Bates, *Rusz głową! Java*. Wydawnictwo Helion, Wydanie II, 2011.
- [8] Cat S. Horstmann, Gary Cornell, *Java Podstawy Wydanie VIII*. Wydawnictwo Helion, 2008.
- [9] Randal L. Schwartz, Tom Christiansen, *Perl - Wprowadzenie*. Wydawnictwo O'Reilly, 2000.
- [10] <https://code.google.com/p/zxing/>
- [11] <http://www.raspberrypi.org/>
- [12] <http://www.gimp.org/>