PROJEKT GR.4 WYPOŻYCZALNIA ROWERÓW

Przedmiot: Metodyki Wytwarzania Oprogramowania

Projekt: Projekt i implementacja systemu informacyjnego z wykorzystaniem wybranej metodyki wytwarzania oprogramowania

1. Informacje ogólne

- Tytuł projektu: Aplikacja webowa z wypożyczalnią rowerów
- Autorzy:

Emil	Łyszczak
Michał	Floriańczyk
Kamil	Cios
Piotr	Gumkowski
Jakub	Zarzycki
Igor	Odolak

- Kierunek / Rok: Informatyka rok III
- Semestr: 5
- Prowadzący: dr Marcin Kacprowicz AEH

2. Opis projektu

2.1. Cel projektu

Celem projektu jest zaprojektowanie i zaimplementowanie systemu wypożyczalni rowerów, który umożliwia użytkownikom wybór danego modelu roweru i wypożyczenie go w konkretnym terminie. System zostanie opracowany zgodnie z metodyką Scrum.

2.2. Zakres projektu

Baza danych – przechowuje informacje o rowerach, użytkownikach oraz historii wypożyczeń; zapewnia trwałość danych.

Panel administracyjny - umożliwia administratorowi zarządzanie flotą rowerów i podgląd historii wypożyczeń.

Usługa webowa - realizuje logikę biznesową systemu, obsługuje żądania użytkowników (np. wypożyczenie, zwrot, dodanie roweru) i komunikuje się z bazą danych.

Moduł uwierzytelniania - odpowiada za rejestrację i logowanie użytkowników oraz kontrolę dostępu.

3. Wymagania systemowe

3.1. Wymagania funkcjonalne

Funkcje użytkownika:

F1. Rejestracja użytkownika. Umożliwia tworzenie konta. Priorytet: Średni

F2. Przegląd listy dostępnych rowerów. Umożliwia podjęcie decyzji, jaki model użytkownik chce wypożyczyć. Priorytet: Wysoki

F3. Wypożyczanie roweru: Umożliwia wypożyczenie wybranego modelu roweru na dany termin i rezerwację miejsca odbioru. Priorytet: Wysoki

F4. Formularz z wypełnieniem danych klienta: Niezbędny do wypożyczenia roweru. Priorytet: Wysoki

F5. Możliwość zwrotu i reklamacji wadliwego roweru. Użytkownik może wysłać zgłoszenie o wadach, niezgodnościach wypożyczonego roweru przed terminem wypożyczenia. Priorytet: Niski

F6. Mapka z dostępnymi modelami: Użytkownik może przeglądać na mapie, w których punktach są dostępne modele do wypożyczenia. Priorytet: Niski

F7. Pobranie historii wypożyczeń użytkownika: Użytkownik może sprawdzić, w których miejscach i w jakich dniach wypożyczał modele, śledząc swoją aktywność. Priorytet: Niski

F8. Powiadomienia: Aplikacja wysyła powiadomienia użytkownika na ich e-mail oraz SMS, kiedy dokonają rezerwacji. Priorytet: Średni

F9. Oceny i opinie: Użytkownicy mogą dokonywać ocen danego modelu roweru, aby pomóc w doborze innym użytkownikom. Priorytet: Niski

F10. Programy lojalnościowe: Użytkownicy mogą korzystać z programów lojalnościowych za wypożyczanie rowerów, dzięki czemu zdobywają zniżki i bonusy zachęcające do ponownego korzystania z aplikacji. Priorytet: niski

F11. Szybkie zgłoszenia: możliwość zgłoszenia przez chat nagłych problemów z rowerem, jak np. przebita dętka, instrukcje korzystania. Priorytet: Niski

F12. Płatności online: Możliwość zapłacenia za wypożyczenie za pośrednictwem PayU/Blik/Karta/Google Pay/Apple PAY. Priorytet: Wysoki

Funkcje administracyjne:
F13. Zarządzanie flotą rowerów przez administratora: Dodawanie i usuwanie modeli rowerów. Priorytet: Wysoki

F14. Podgląd historii wypożyczeń rowerów: Możliwość analizy i pobrania raportu o wszystkich wypożyczeniach dokonanych przez użytkowników. Priorytet: Średni

F15: Pobranie historii wypożyczenia danego użytkownika. Analiza, czy użytkownicy są powracający, jak często wypożyczają. Priorytet: Niski

F16. Weryfikacja zgłoszeń o zwroty i reklamacje. Administrator weryfikuje i odpowiada na zgłoszenia z panelu klienta, czy przyjmuje reklamacje i zwrot pieniędzy czy odrzuca. Priorytet: Niski

F17. Zarządzanie kontami użytkowników: Administrator może blokować konta, które naruszają regulaminy. Priorytet: Średni

F18. Zarządzanie punktami wypożyczeń: Administrator może modyfikować istniejące lokalizacje oraz dodawać nowe. Priorytet: Średni

F19. Raporty statystyczne: Administrator może wygenerować na podstawie historii wypożyczeń analizy statystyczne dotyczące popularności danego modelu, liczby aktywnych użytkowników, czasu wypożyczenia itd Priorytet: Niski

F20. Kody rabatowe: Administrator może generować użytkownikom specjalne kody zniżkowe na wypożyczenie. Priorytet: Niski

3.2. Wymagania niefunkcjonalne

N1. Responsywność: aplikacja jest dobrze wyświetlana zarówno na komputerze jak i na telefonie. Kategoria: Wydajność
N2. Bezpieczeństwo: aplikacja musi przechowywać dane użytkowników i historię wypożyczeń w sposób bezpieczny. Kategoria: Bezpieczeństwo
N3. Niezawodność: system działa bez przerwy, reagując na problemy i pomyłki użytkowników. Kategoria: Wydajność

N4. Obsługa wielu klientów w jednym czasie: System reaguje na przeciążenia, pozwala na zakładanie dużej ilości kont i wypożyczanie rowerów w czasie rzeczywistym przez wielu użytkowników na raz. Kategoria: Efektywność

N5. Intuicyjność. Aplikacja webowa i panel administracyjny są zaprojektowane w sposób przystępny i intuicyjny dla użytkownika, zmniejszając ilość pomyłek. Kategoria: Efektywność

N6. Integralność danych. Dane w bazie (np. dostępność rowerów) muszą być zawsze spójne, nawet przy równoczesnych operacjach wielu użytkowników. Kategoria: Bezpieczeństwo

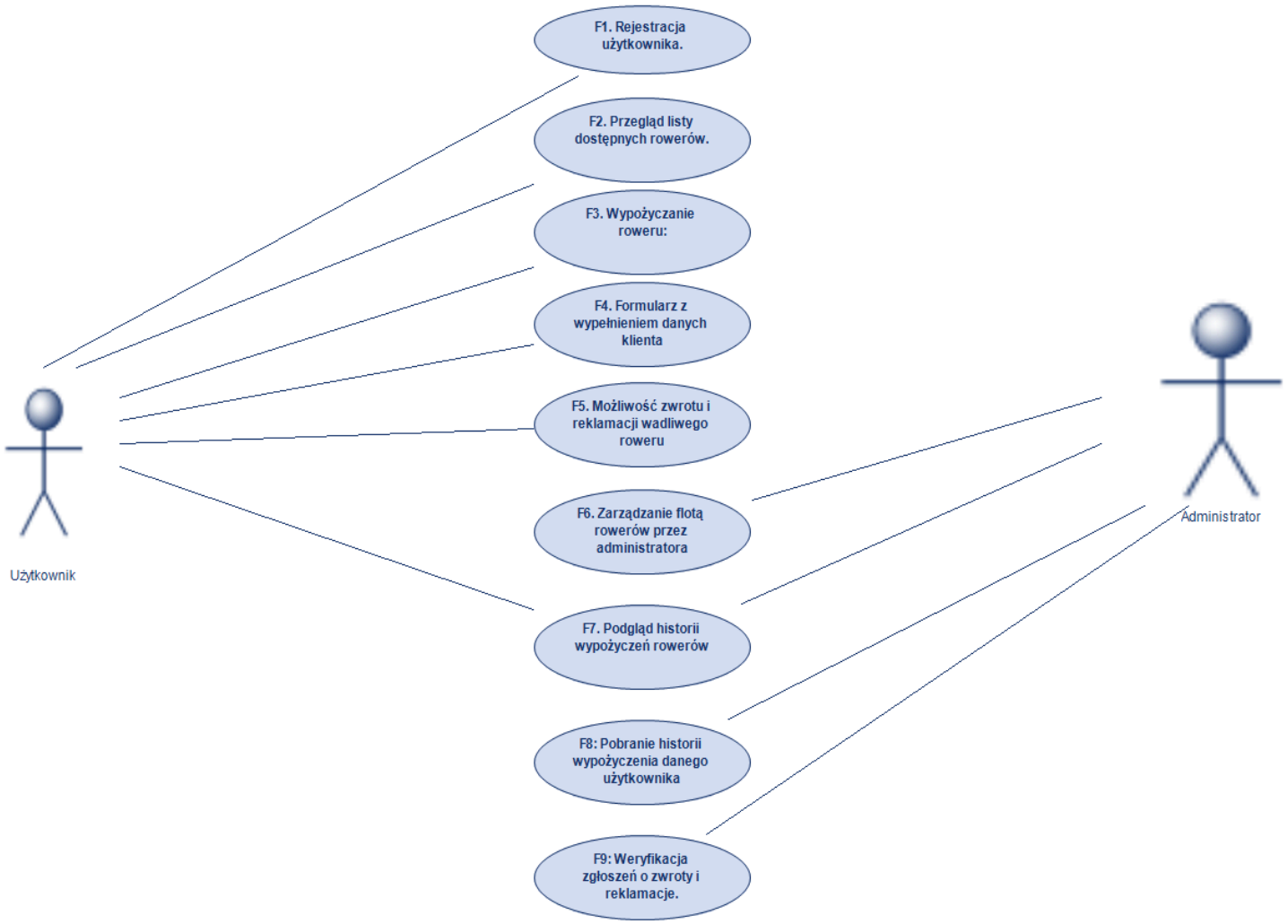
N7. Dostępność WCAG 2.1 AA - Zgodność ze standardem dostępności, aby aplikacja mogła być czytelna dla większości użytkowników, w tym tych z niepełnosprawnościami. Kategoria: Użyteczność

N8. Audytowalność i śledzenie zmian - Możliwość sprawdzania wprowadzonych zmian przez

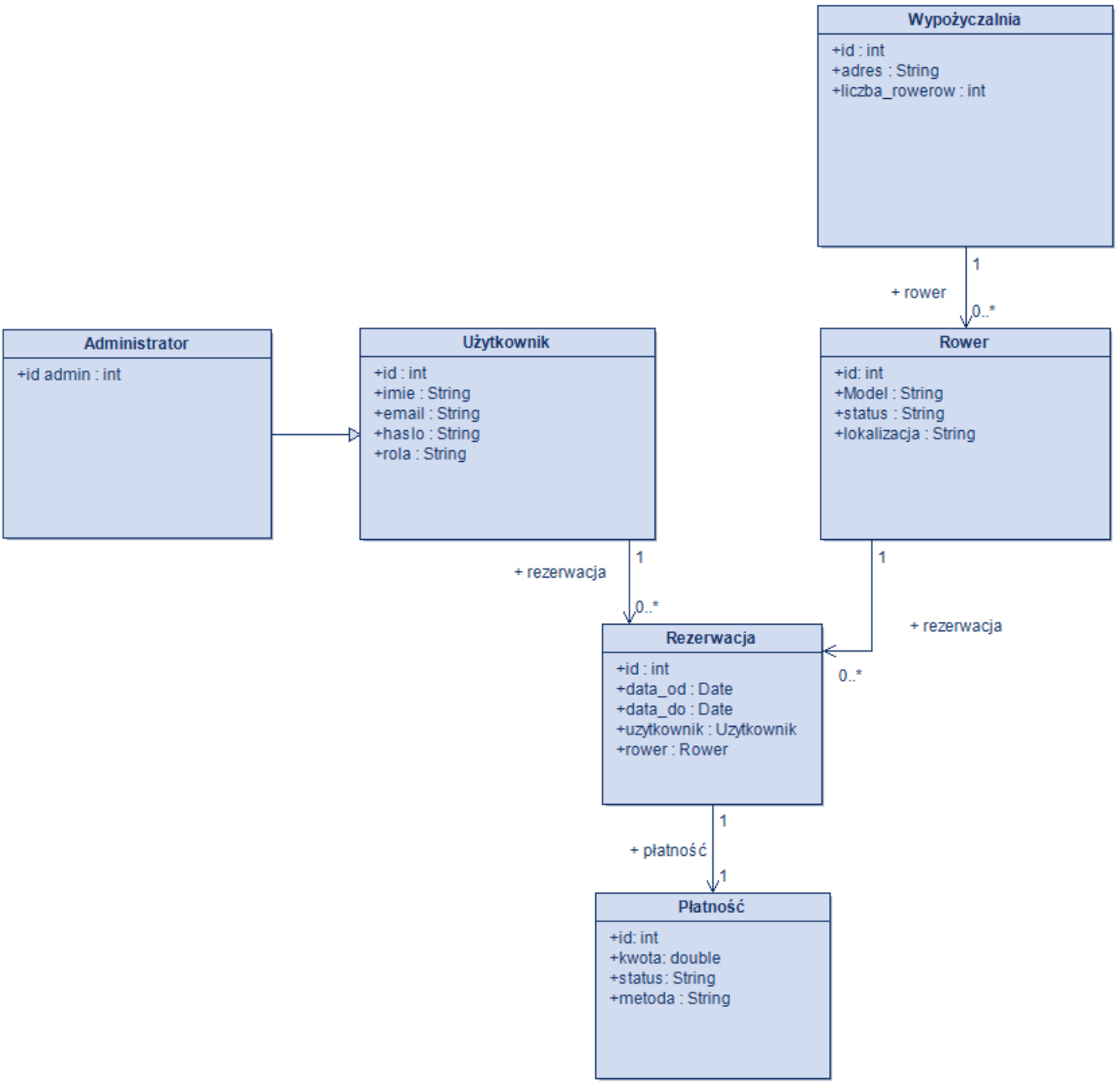
administratorów dotyczących poszczególnych możliwości użytkowników czy wypożyczeń. - Kategoria: Bezpieczeństwo

4. Diagramy UML

4.1. Diagram przypadków użycia



4.2. Diagram klas



5. Implementacja metodyki

W metodyce Scrum proces wytwarzania został podzielony na kilka sprintów. Każdy sprint obejmował planowanie, implementację, testowanie i przegląd wyników.

Zdecydowaliśmy się na wybór metodyki Waterfall do realizacji tego projektu. Model kaskadowy pozwala na przechodzenie po kolei przez odgórnie zaplanowane etapy, rozpoczynając kolejny dopiero po zakończeniu poprzedniego. Ta metodyka dobrze sprawdziła się w naszym przypadku, ponieważ projekt miał stabilne i jasne wymagania.

Kolejność realizacji etapów:
1. Analiza wymagań – określiliśmy zakres projektu oraz skompletowaliśmy listę wymagań funkcjonalnych i niefunkcjonalnych dla systemu wypożyczalni.

2. Projektowanie systemu – wybraliśmy technologię: język Java z frameworkiem Spring Boot, Thymeleaf do widoków oraz SQLite do bazy danych. Zaprojektowaliśmy strukturę aplikacji w oparciu o diagramy UML (klas i przypadków użycia).

3. Implementacja – zaimplementowaliśmy kolejno moduły aplikacji: warstwę bazy danych, logikę biznesową, panel administratora z obsługą zdjęć oraz interfejs użytkownika.

4. Testowanie – po zakończeniu prac programistycznych przeprowadziliśmy testy manualne kluczowych procesów, takich jak wypożyczanie czy logowanie.

5. Wdrożenie – system został uruchomiony lokalnie jako kompletna aplikacja webowa z automatyczną inicjalizacją bazy danych.

6. Konserwacja – na końcowym etapie poprawiliśmy wykryte błędy konfiguracyjne i usprawniliśmy działanie panelu edycji danych.

6. Wzorce projektowe i architektoniczne

Projekt został zrealizowany w oparciu o architekturę MVC (Model-View-Controller), co zapewniło wyraźny podział na logikę biznesową, model danych oraz warstwę prezentacji (widoki Thymeleaf). Zastosowano również architekturę warstwową, oddzielającą dostęp do danych (Repozytorium) od obsługi żądań (Controller).

Wzorce projektowe wykorzystane w aplikacji to:

- Repozytorium: Abstrakcja warstwy danych, umożliwiająca operacje na bazie SQLite bez użycia surowego SQL
- Wstrzykiwanie zależności: Mechanizm frameworka Spring używający @Autowired, który automatycznie zarządza tworzeniem i łączeniem komponentów aplikacji.
- Singleton: Domyślny wzorzec dla komponentów Springa, który zapewnia istnienie tylko jednej instancji kontrolera i serwisu w pamięci, co optymalizuje wydajność.

7. Testowanie

Kod Testu

ID testu	Opis	Wynik
T1	Test jednostkowy poprawności logiki wypożyczenia.	Zaliczone
T2	Test próby wejścia do panelu administratora bez logowania	Zaliczone
T3	Test poprawnego logowania do panelu administratora z hasłem admin123	Zaliczone
T4	Test dodawania nowego obiektu do bazy danych	Zaliczone
T5	Test poprawności strony głównej	Zaliczone

8. Architektura systemu

System został zaprojektowany w architekturze klient-serwer jako aplikacja webowa oparta na frameworku Spring Boot. Wewnętrzna struktura projektu realizuje wzorzec MVC, co zapewnia oddzielenie logiki biznesowej od warstwy wizualnej Thymeleaf oraz warstwy danych. Główne moduły systemu to moduł klienta (przeglądanie i rezerwacja), zabezpieczony panel administratora (zarządzanie flotą i uwierzytelnianie) oraz warstwa trwała odpowiedzialna za integrację z bazą SQLite. Komunikacja w systemie odbywa się za pośrednictwem protokołu HTTP, wykorzystując założenia stylu architektonicznego REST (obsługa żądań GET do pobierania zasobów i POST do przesyłania formularzy).

9. Szczegóły implementacji

Technologie i narzędzia:

- **Java**: język programowania Logiki biznesowej aplikacji.
- **Spring Boot 3**: Framework zarządzający konfiguracją, serwerem i zależnościami.
- **Spring Data JPA (Hibernate)**: Narzędzie do mapowania obiektów Java na tabele w bazie danych.
- **Thymeleaf**: Silnik szablonów do generowania dynamicznego HTML po stronie serwera.
- **SQLite**: Bezserwerowa baza danych przechowująca informacje w pliku.
- **Maven**: Narzędzie do automatyzacji budowania projektu i zarządzania bibliotekami.
- **JUnit 5 & Mockito**: Biblioteki do testów jednostkowych i integracyjnych.

Przykłady implementacji:
- Klasa **Rower.java**: Definiuje strukturę danych roweru i mapuje ją bezpośrednio na kolumny w tabeli bazy danych.

- Klasa **AdminController.java**: Kontroler obsługujący żądania HTTP. Łączy warstwę prezentacji z bazą danych, weryfikuje sesję użytkownika i przekazuje dane do widoku.

- Plik **admin.html**: Warstwa widoku. Odpowiada za dynamiczne wyświetlenie tabeli z rowerami pobranymi z kontrolera.

10. Wnioski i dalszy rozwój

Projekt realizuje podstawowe cele: umożliwia kompleksowe zarządzanie flotą rowerów (wraz z obsługą zdjęć), uwierzytelnianie administratora oraz składanie rezerwacji przez klientów. System działa stabilnie w oparciu o bazę SQLite, a poprawność kluczowych funkcjonalności została zweryfikowana testami automatycznymi.

Możliwe dalsze usprawnienia obejmują wprowadzenie modułu kont dla klientów (rejestracja i historia wypożyczeń), wdrożenie API REST do integracji z aplikacjami mobilnymi oraz rozbudowę walidacji formularzy. System można również rozwinąć o powiadomienia e-mail, integrację z bramką płatności online oraz mapę z lokalizacją rowerów.

11. Bibliografia

https://start.spring.io
https://mvnrepository.com

https://docs.oracle.com/javase/8/docs/api/overview-summary.html

https://www.thymeleaf.org/documentation.html

https://docs.junit.org/current/user-guide/

https://site.mockito.org

https://docs.spring.io/spring-data/jpa/reference/4.1-SNAPSHOT/jpa/getting-started.html

https://chatgpt.com/

https://gemini.google.com/