Specyfikacja Projektowa Systemu (SDS)

System Harmonogramu Szkoły Jazdy

Projekt: **Scheduler**

Data: 3 lutego 2026 r.

Wersja: 1.0.0 (Final version)

1. Architektura Systemu

1.1 Model Architektoniczny

Aplikacja została zaprojektowana w oparciu o klasyczną **Architekturę Warstwową (Layered Architecture)**, co zapewnia separację odpowiedzialności oraz ułatwia testowanie poszczególnych komponentów:

- **Warstwa Prezentacji (Web Layer):** Wykorzystuje Spring MVC i silnik szablonów Thymeleaf do renderowania interfejsu użytkownika po stronie serwera (SSR).
- **Warstwa Logiki Biznesowej (Service Layer):** Zawiera serwisy (`BookingService`, `CalendarService`), które implementują rygorystyczne reguły biznesowe (BR) zdefiniowane w URS.
- **Warstwa Dostępu do Danych (Persistence Layer):** Opiera się na Spring Data JPA, umożliwiając abstrakcję nad relacyjną bazą danych PostgreSQL.
- **Warstwa Domenowa (Domain Layer):** Definiuje encje JPA (`User`, `Lesson`) stanowiące fundament modelu danych.

1.2 Stos Technologiczny (Tech Stack)

Komponent	Technologia	Uwagi
Język Programowania	Java 21	Wykorzystanie najnowszego wydania LTS dla optymalnej wydajności.
Framework Główny	Spring Boot 4.0.2	Podstawa ekosystemu aplikacji.
Zarządzanie Zależnościami	Maven	Standardowe narzędzie do budowania projektu.
Baza Danych	PostgreSQL	Relacyjna baza danych zapewniająca spójność danych.
Uwierzytelnianie	Clerk (OIDC/OAuth2)	Zewnętrzny dostawca tożsamości zapewniający bezpieczeństwo.
Frontend	Thymeleaf + Bootstrap 5	Dynamiczne renderowanie widoków z responsywnym stylem CSS.

2. Model Danych i Persystencja

2.1 Struktura Tabel

System wykorzystuje dwie kluczowe encje powiązane relacją jeden-do-wielu:

A. Użytkownik (`app_user`)

Przechowuje dane profilowe oraz stan salda godzinowego. Nazwa tabeli została zmieniona na `app_user`, aby uniknąć konfliktu ze słowem zastrzeżonym `USER` w SQL.

- `id` (BigInt, PK): Automatycznie generowany klucz główny.
- `clerk_id` (String, Unique): Mapowanie na identyfikator zewnętrzny dostawcy Clerk.
- `remaining_hours` (Int): Licznik pozostałych godzin szkoleniowych (domyślnie 30).
- `role` (String/Enum): Rola użytkownika (`STUDENT`, `ADMIN`).

B. Lekcja (`lesson`)

Reprezentuje konkretny slot czasowy w harmonogramie.

- `id` (BigInt, PK): Klucz główny.
- `start_time` (Timestamp): Data i godzina rozpoczęcia lekcji (dokładność do godziny).
- `user_id` (FK): Relacja `@ManyToOne` do kursanta rezerwującego termin (nullable dla blokad).
- `status` (String/Enum): Stan slotu (`AVAILABLE`, `BOOKED`, `BLOCKED`).

2.2 Repozytoria

Zastosowano mechanizmy Spring Data JPA do obsługi zapytań zakresowych, kluczowych dla generowania widoku kalendarza (np. `findAllByStartTimeBetween`).

3. Bezpieczeństwo i Integracja z Clerk

3.1 Mechanizm Uwierzytelniania

Aplikacja integruje się z platformą **Clerk** za pomocą protokołów **OAuth2** oraz **OpenID Connect (OIDC)**.

1. Użytkownik loguje się przez panel Clerk.
2. Po pomyślnym uwierzytelnieniu, system otrzymuje token ID oraz Access Token.
3. `CustomOidcUserService` przechwytuje moment logowania, weryfikuje istnienie użytkownika w lokalnej bazie danych i w razie potrzeby inicjalizuje nowy profil (Just-In-Time Provisioning).

3.2 Autoryzacja i Role

Zabezpieczenia są konfigurowane w klasie `SecurityConfig` za pomocą `SecurityFilterChain`:

- **Publiczne:** Strona główna, zasoby statyczne (CSS/JS).
- **Role-Based Access Control (RBAC):** Ścieżki zaczynające się od `/admin/**` wymagają posiadania roli `ROLE_ADMIN`. Wszystkie inne żądania wymagają statusu `authenticated`.

---

4. Logika Biznesowa (Implementacja)

4.1 Walidacja Rezerwacji (`BookingService`)

Proces rezerwacji lekcji (`bookSlot`) podlega kaskadowej weryfikacji warunków:

1. **Check Date:** Blokowanie prób rezerwacji wstecznej.
2. **Check Availability:** Weryfikacja, czy dany slot nie został już zajęty przez innego kursanta lub zablokowany przez administratora (atomowość transakcji).
3. **Check Balance:** Weryfikacja, czy pole `remaining_hours` kursanta jest większe od zera.
4. **Daily Limit Rule (BR-02):** Zapytanie do bazy danych zliczające lekcje kursanta w danym dniu kalendarzowym (max 1).

4.2 Transakcyjność

Wszystkie operacje modyfikujące (rezerwacja, anulowanie) są oznaczone adnotacją `@Transactional`. Gwarantuje to, że dekrementacja salda godzin użytkownika nastąpi tylko wtedy, gdy rezerwacja lekcji zostanie pomyślnie zapisana w bazie danych.

5. Interfejs i Routing

5.1 Główne Endpointy

- `GET /`: Strona powitalna z informacją o systemie.
- `GET /schedule`: Główny widok kalendarza dla kursanta (wymaga logowania).
- `POST /book`: Endpoint procesujący żądania rezerwacji.
- `GET /admin/dashboard`: Lista kursantów i zarządzanie godzinami.
- `GET /admin/schedule`: Rozszerzony widok kalendarza dla instruktora.

5.2 Strategia Renderowania

Wykorzystanie Thymeleaf pozwala na dynamiczne aplikowanie klas CSS (np. `slot-available`, `slot-booked`) bezpośrednio podczas renderowania strony na serwerze, co eliminuje opóźnienia widoczne w aplikacjach typu SPA (Single Page Application) przy słabszych łączach.

---

6. Utrzymanie i Rozwój

6.1 Środowisko Testowe

System wspiera testy integracyjne z wykorzystaniem bazy danych H2 (w profilu `test`) oraz `@SpringBootTest`, co pozwala na automatyczną weryfikację reguł biznesowych bez konieczności manualnego klikania w interfejsie.

6.2 Monitoring

Logowanie zdarzeń (logging) zostało skonfigurowane na poziomie `DEBUG` dla modułów bezpieczeństwa, co ułatwia diagnostykę problemów z tokenami JWT i komunikacją z serwerami Clerk.