### 

### UNIWERSYTET RZESZOWSKI

**Wydział Nauk Ścisłych i Technicznych**

**Rafał Burbeło**

**Nr albumu: 131412**

**Kierunek: Informatyka I rok**

**Studia I stopnia stacjonarne**

**System elektronicznego głosowania – projekt zaliczeniowy**

**JavaFX, Hibernate z bazą H2, CRUD**

**Dokumentacja techniczna**

**Semestr 2, rok akademicki 2024/2025**

**Grupa 1**

**Prowadzący:**

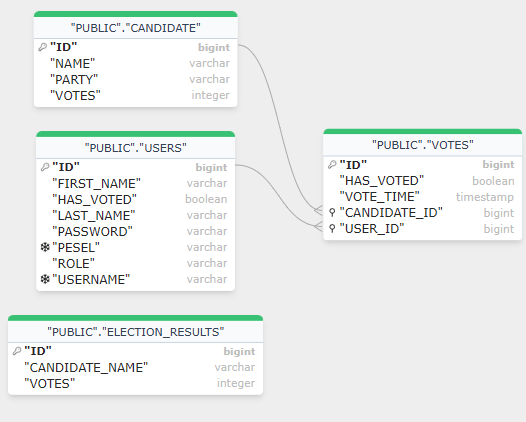
**dr inż. Wojciech Kozioł**

**Rzeszów, 03.06.2025 r.**

**1. Plan skryptu i założenia projektu**

**Cel projektu**

Projekt "Wybory" jest aplikacją desktopową stworzoną w Javie z użyciem JavaFX oraz Hibernate. Umożliwia przeprowadzanie głosowania elektronicznego, zarządzanie użytkownikami i kandydatami oraz podgląd wyników głosowania.

**Schemat ERD bazy danych H2:**  


**Baza danych H2** stanowi rdzeń przechowywania danych w systemie, działając jako **embedded database** (baza osadzona), co oznacza, że jest uruchamiana w procesie aplikacji bez konieczności instalacji zewnętrznego serwera. Kluczowe cechy:

1. **Struktura schematu**  
   Baza oparta jest o model relacyjny z tabelami:
   * USERS (dane użytkowników, role, PESEL)
   * CANDIDATE (kandydaci, partie)
   * VOTES (rejestr głosów)
   * ELECTION\_RESULTS (agregacja wyników)
2. **Automatyczna inicjalizacja**  
   Przy starcie aplikacji Hibernate automatycznie:
   * Generuje schemat na podstawie adnotacji JPA w modelach
   * Wypełnia dane startowe z plików SQL (import.sql, import\_dynamic.sql)
3. **Mechanizmy bezpieczeństwa**
   * Hasła przechowywane jako **hashe BCrypt** (nie w formie plaintext)
   * Unikalne walidacje PESEL i loginów
4. **Zarządzanie transakcjami**  
   Operacje CRUD realizowane przez Hibernate z pełnym wsparciem ACID, w tym:
   * Automatyczne rollbacki przy błędach
   * Optymistyczna kontrola wersji
5. **Backup dynamiczny**  
   System automatycznie eksportuje nowych użytkowników do pliku import\_dynamic.sql, umożliwiając odtworzenie stanu po restarcie aplikacji.

**Dlaczego H2?**

* Idealna do celów deweloperskich/testowych (brak zależności od zewnętrznych serwerów)
* Pełna kompatybilność z JPA/Hibernate
* Wysoka wydajność operacji w pamięci
* Możliwość migracji na produkcyjne bazy (PostgreSQL/MySQL) poprzez zmianę konfiguracji Hibernate.

Baza stanowi integralną część architektury, zapewniając spójność danych przy jednoczesnej lekkości rozwiązania.

**Główne założenia:**

Projekt zaliczeniowy z przedmiotu **Programowanie obiektowe**.

* Aplikacja desktopowa w języku Java.
* Połączenie z relacyjną bazą danych, zaprojektowaną wcześniej.
* Baza danych składa się z co najmniej trzech tabel połączonych relacjami: User (Użytkownicy), Candidate (Kandydaci), Vote (Głosy — tabela pośrednia).
* Udostępnienie funkcjonalności CRUD (Create, Read, Update, Delete) dla danych w bazie.
* Graficzny interfejs użytkownika (GUI) zrealizowany przy użyciu JavaFX.
* Komunikacja z bazą danych za pomocą Hibernate (ORM).
* Estetyczny, intuicyjny interfejs użytkownika.
* Bezpieczne logowanie z wykorzystaniem hashowania haseł (bcrypt).
* Dokumentacja techniczna zawierająca: opis projektu, opis bazy danych i ERD, zrzuty ekranu GUI, opis funkcjonalności oraz fragmenty kodu źródłowego.

**2. Implementacja**

**Technologie:**

* **Język programowania:** Java 17
* **GUI:** JavaFX (FXML)
* **ORM:** Hibernate (JPA)
* **Baza danych:** H2 (pamięciowa lub plikowa)
* **Logowanie:** Logback
* **Hashowanie haseł:** bcrypt (via HashGenerator, cost=10, prefix=2a)

**Struktura projektu**

Wybory/

├── .idea/

├── .mvm/

├── database/

├── src/

│ ├── main/

│ │ ├── java/

│ │ │ └── com/

│ │ │ └── election/

│ │ │ ├── controller/

│ │ │ │ ├── AdminController.java

│ │ │ │ ├── LoginController.java

│ │ │ │ └── UserController.java

│ │ │ ├── dao/

│ │ │ │ ├── CandidateDAO.java

│ │ │ │ └── UserDAO.java

│ │ │ ├── exception/

│ │ │ │ ├── DatabaseException.java

│ │ │ │ ├── ServiceException.java

│ │ │ │ ├── ValidationException.java

│ │ │ │ ├── ViewLoadingException.java

│ │ │ │ └── VotingException.java

│ │ │ ├── model/

│ │ │ │ ├── Candidate.java

│ │ │ │ ├── CandidateResult.java

│ │ │ │ ├── ElectionResult.java

│ │ │ │ ├── User.java

│ │ │ │ └── Vote.java

│ │ │ ├── service/

│ │ │ │ ├── AuthService.java

│ │ │ │ ├── ElectionService.java

│ │ │ │ ├── ExportServicePDF.java

│ │ │ │ └── VotingService.java

│ │ │ └── util/

│ │ │ ├── DataInitializer.java

│ │ │ ├── HibernateUtil.java

│ │ │ ├── HashGenerator.java

│ │ │ ├── Main.java

│ │ │ └── module-info.java

│ │ └── resources/

│ │ └── com/

│ │ └── election/

│ │ └── view/

│ │ ├── admin\_view.fxml

│ │ ├── login.fxml

│ │ ├── style.css

│ │ └── user\_view.fxml

│ └──

│ ├── app\_icon.png

│ ├── hibernate.cfg.xml

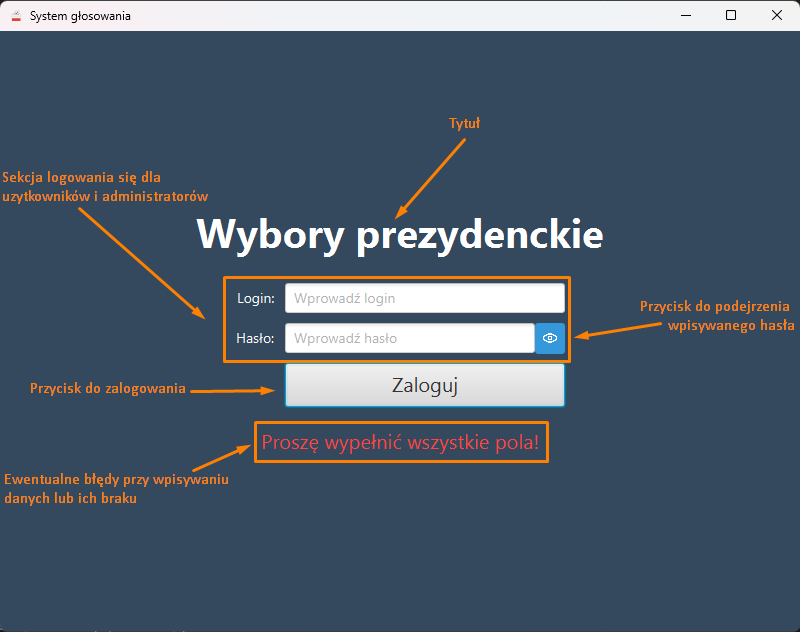
│ ├── import.sql

│ ├── import\_dynamic.sql

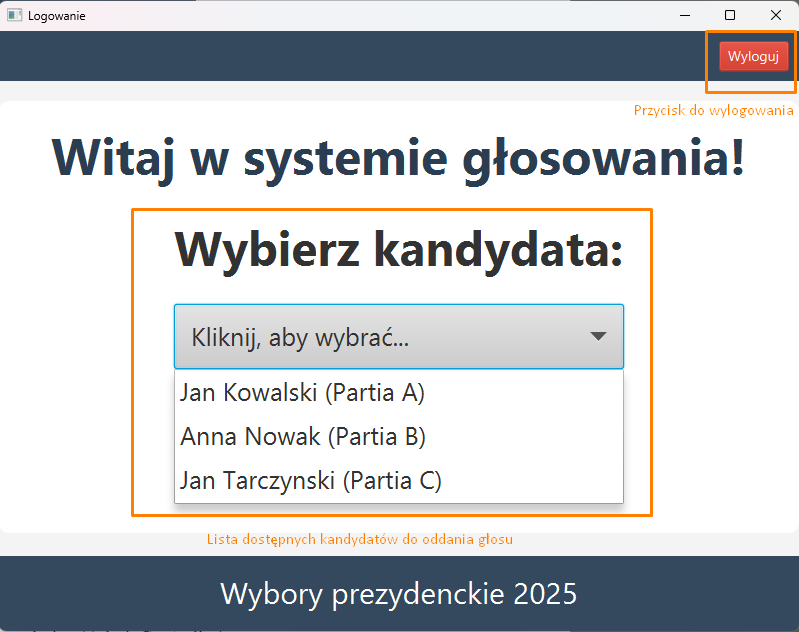
│ └── logback.xml

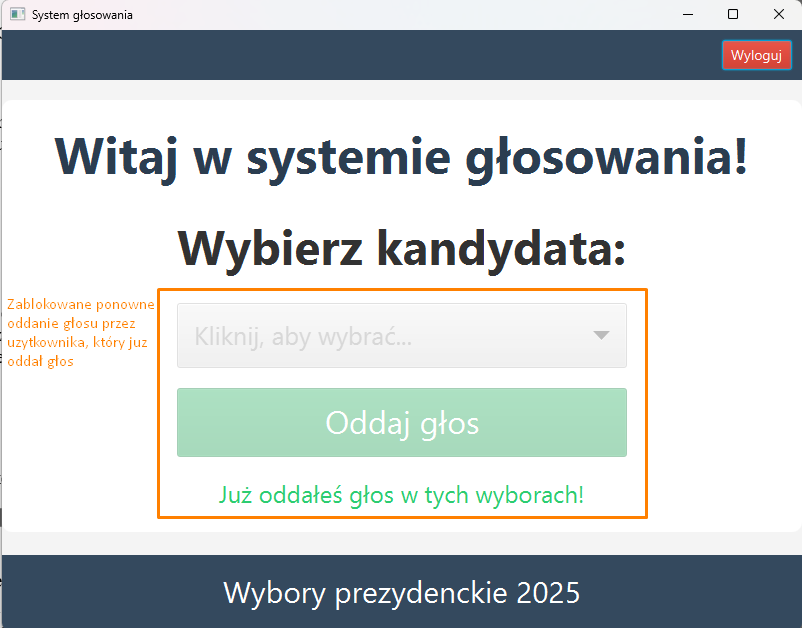
**Kluczowe funkcjonalności:**

* Logowanie użytkownika (LoginController):

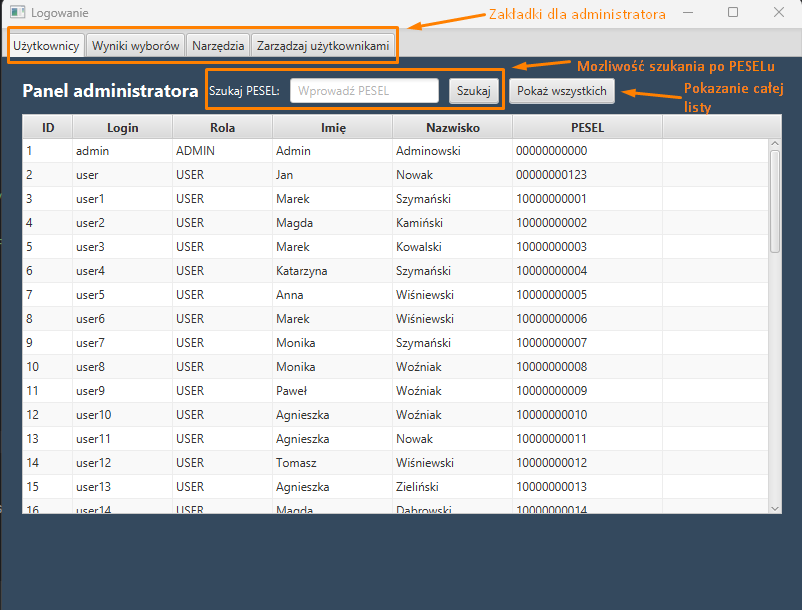


* Głosowanie przez użytkownika (UserController):

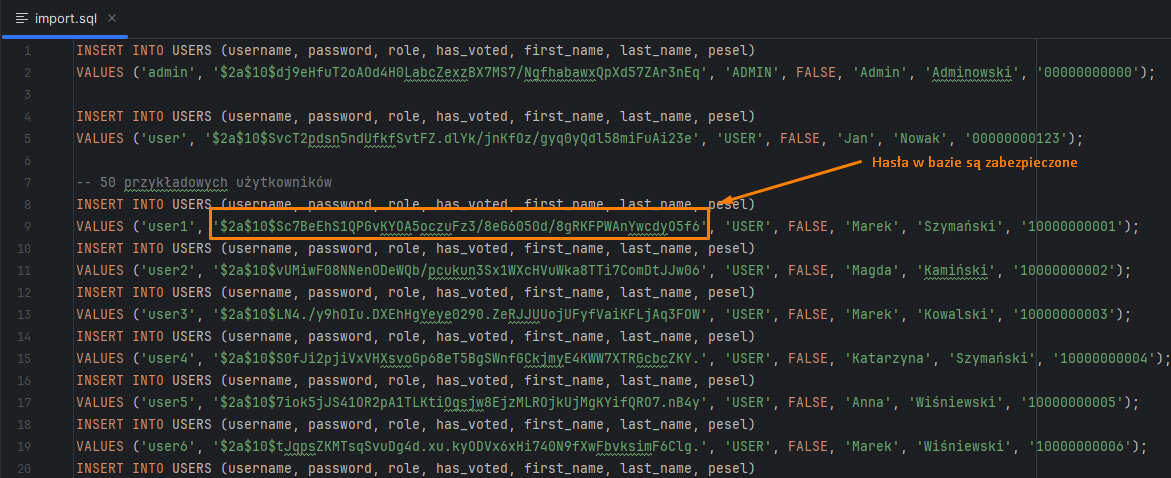


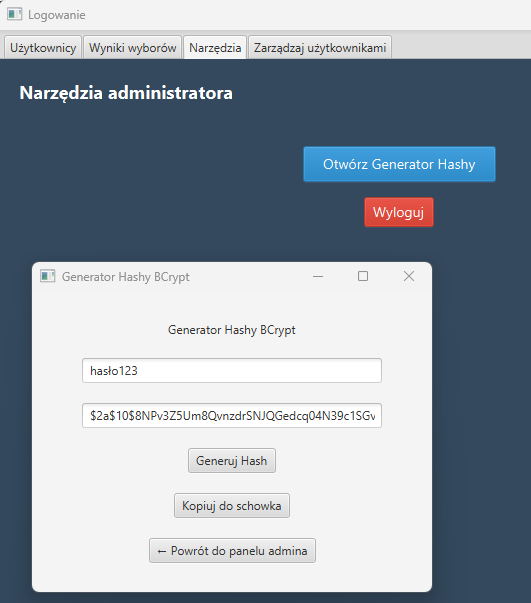


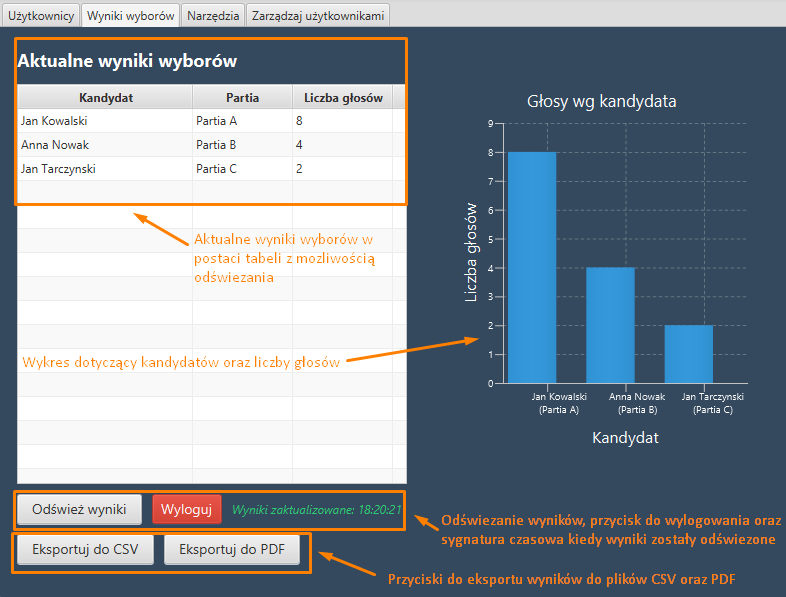
* Zarządzanie kontami i kandydatami przez admina (AdminController):



* Dynamiczne ładowanie danych z pliku SQL:



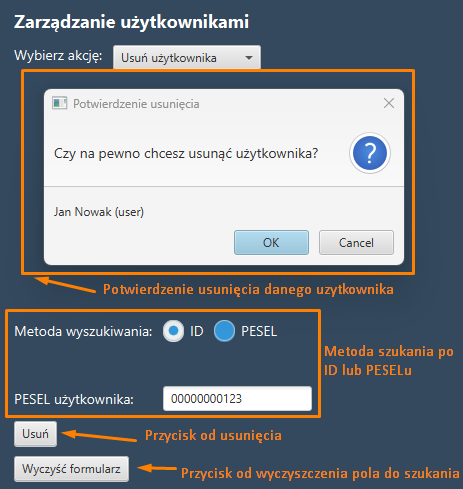
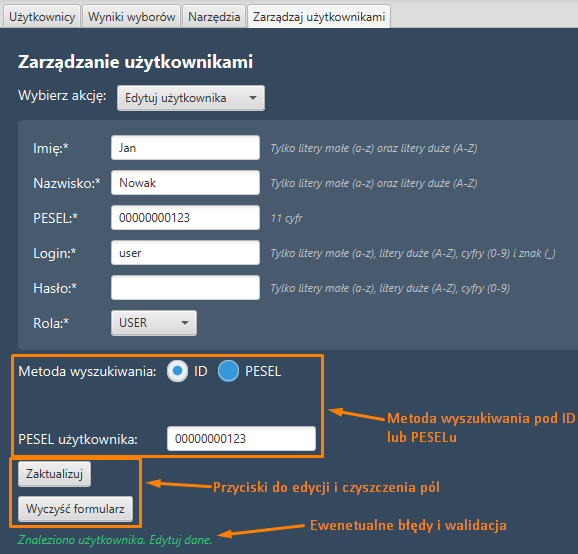
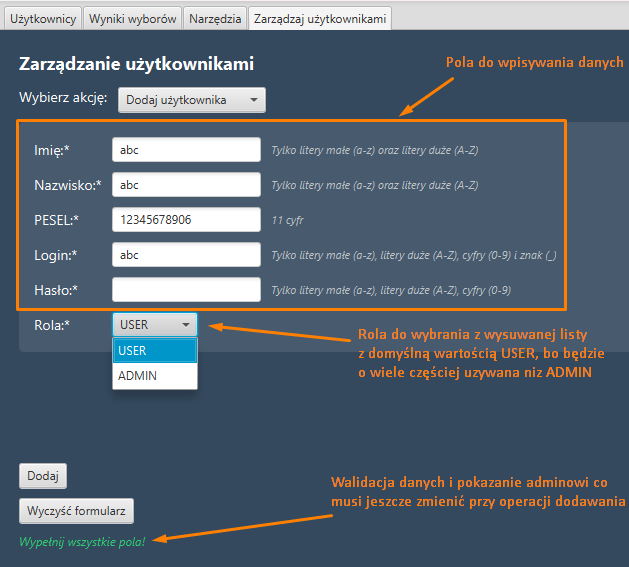
* Możliwość wygenerowania zahashowanego hasła:  
  
* Sekcja wyniki wyborów:



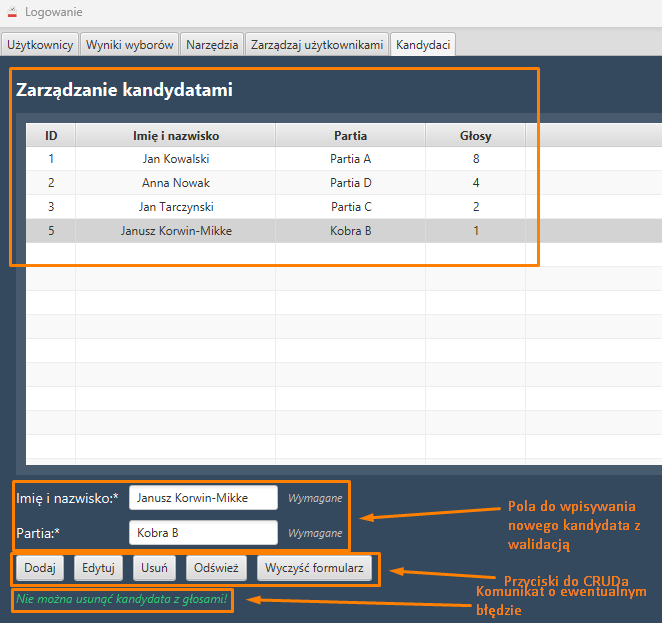
* CRUD – ,,Create Read Update Delete” czyli zarządzanie użytkownikami w bazie danych za pomocą administratora:

Administrator ma możliwość wyboru operacji CRUD poprzez rozwijaną listę:





* Kolejny CRUD tym razem dla Kandydatów:



**3. Testowanie**

**Rodzaje testów:**

* **Testy jednostkowe**: testowano funkcje hashujące oraz DAO
* **Testy integracyjne**: testowano połączenie Hibernate z H2
* **Testy manualne GUI**: testowanie interfejsu JavaFX

**Przykładowe przypadki testowe:**

|  |  |  |
| --- | --- | --- |
| **Test** | **Opis** | **Wynik** |
| Logowanie poprawne | Logowanie dla użytkownika admin z poprawnym hasłem | OK |
| Głosowanie podwójne | Użytkownik próbuje głosować drugi raz | Operacja niedozwolona – system uniemożliwia ponowne głosowanie. |
| Dodanie użytkownika | Administrator dodaje użytkownika z unikalnym PESEL | OK |
| Usunięcie użytkownika | Administrator usuwa istniejącego użytkownika | OK |
| Niepoprawny PESEL | PESEL niezgodny z formatem | Błąd walidacji |

**4. Dokumentacja**

**1. Klasy główne aplikacji**

1. HashGenerator.java
   * Odpowiedzialność: Generuje hashe BCrypt z podanych haseł
   * Funkcje:
     + Interfejs GUI z polami na hasło i wynik hashowania
     + Przyciski generacji i kopiowania do schowka
     + Integracja z biblioteką jbcrypt
2. Main.java
   * Odpowiedzialność: Punkt startowy aplikacji
   * Funkcje:
     + Inicjalizacja Hibernate i bazy danych
     + Ładowanie ekranu logowania (login.fxml)
     + Zarządzanie cyklem życia aplikacji

**2. Kontrolery (JavaFX)**

1. AdminController.java
   * Odpowiedzialność: Panel administratora
   * Funkcje:
     + Zarządzanie użytkownikami (CRUD)
     + Wyświetlanie wyników wyborów (tabela + wykres)
     + Eksport do CSV/PDF
     + Generator hashów BCrypt
2. LoginController.java
   * Odpowiedzialność: Logowanie użytkowników
   * Funkcje:
     + Walidacja danych logowania
     + Przekierowanie do panelu admina/użytkownika
     + Wyświetlanie błędów uwierzytelniania
3. UserController.java
   * Odpowiedzialność: Panel głosowania
   * Funkcje:
     + Wybór kandydata z listy
     + Zatwierdzanie głosu
     + Blokada ponownego głosowania

**3. Warstwa dostępu do danych (DAO)**

1. CandidateDAO.java
   * Odpowiedzialność: Operacje na kandydatach
   * Funkcje:
     + Pobieranie listy kandydatów
     + Dodawanie/usuwanie/aktualizacja
2. UserDAO.java
   * Odpowiedzialność: Operacje na użytkownikach
   * Funkcje:
     + Logowanie i weryfikacja
     + Zarządzanie kontami
     + Automatyczny backup do pliku SQL

**4. Modele danych**

1. Candidate.java
   * Odpowiedzialność: Reprezentacja kandydata
   * Pola: ID, nazwa, partia, liczba głosów
2. CandidateResult.java
   * Odpowiedzialność: Wyniki wyborów
   * Pola: ID kandydata, nazwa, głosy
3. User.java
   * Odpowiedzialność: Użytkownik systemu
   * Pola: Login, hasło, rola, PESEL, status głosu
4. Vote.java
   * Odpowiedzialność: Rekord głosu
   * Pola: Użytkownik, kandydat, czas głosowania

**5. Serwisy biznesowe**

1. AuthService.java
   * Odpowiedzialność: Uwierzytelnianie
   * Funkcje: Weryfikacja hasła z użyciem BCrypt
2. ElectionService.java
   * Odpowiedzialność: Logika wyborów
   * Funkcje:
     + Pobieranie wyników
     + Rejestracja głosów
3. ExportServicePDF.java
   * Odpowiedzialność: Generowanie raportów PDF
   * Technologie: Biblioteka LibrePDF
4. VotingService.java
   * Odpowiedzialność: Proces głosowania
   * Funkcje:
     + Sprawdzanie uprawnień
     + Aktualizacja stanu głosowania

**6. Narzędzia i pomocnicze**

1. DataInitializer.java
   * Odpowiedzialność: Inicjalizacja bazy danych
   * Funkcje: Automatyczne wypełnianie danymi z plików SQL
2. HibernateUtil.java
   * Odpowiedzialność: Konfiguracja Hibernate
   * Funkcje: Zarządzanie sesjami i połączeniami z DB

**7. Wyjątki**

* DatabaseException: Błędy operacji bazodanowych
* ServiceException: Błędy logiki biznesowej
* ValidationException: Nieprawidłowe dane wejściowe
* VotingException: Błędy procesu głosowania

**8. Zasoby interfejsu**

1. Pliki FXML:
   * admin\_view.fxml: Layout panelu admina
   * login.fxml: Ekran logowania
   * user\_view.fxml: Panel głosowania
2. style.css
   * Stylizacja komponentów GUI (kolory, czcionki, rozmieszczenie)

System składa się z:

* Frontend: JavaFX (FXML + CSS)
* Backend: Hibernate + JDBC
* Bezpieczeństwo: BCrypt do hashowania haseł
* Funkcjonalności:
  + Głosowanie z walidacją
  + Raportowanie wyników (PDF/CSV)
  + Zarządzanie użytkownikami
  + Backup danych do SQL

Architektura oparta na wzorcu MVC z wyraźnym podziałem na warstwy prezentacji, logiki biznesowej i dostępu do danych.

**5. Poprawa błędów**

W trakcie testów zidentyfikowano i poprawiono:

* Błędne ładowanie użytkowników z pliku bez kolumn first\_name, last\_name, pesel
* Błędy walidacji dla duplikatów PESEL
* Brak obsługi pustych haseł i loginów
* Niewyświetlanie aktualnych danych po dodaniu/edycji/użytkownika (rozwiązano przez loadInitialData())

## 6. Eksport wyników

System umożliwia eksportowanie wyników głosowania do pliku CSV oraz PDF, co pozwala administratorowi na zapisanie i analizę wyników poza aplikacją.

### Eksport do CSV

Eksport wyników do formatu CSV umożliwia łatwe otwarcie pliku np. w Microsoft Excel. Dane są eksportowane w formacie:

* **Nagłówki:** Kandydat; Liczba głosów
* **Separator:** średnik (;) – zgodny z polskimi ustawieniami Excela
* **Kodowanie:** UTF-8 z BOM – zapewnia poprawne wyświetlanie znaków diakrytycznych (polskich znaków)

Dane eksportowane są na podstawie zawartości tabeli wyników w interfejsie administratora.

**Funkcja eksportu wyników głosowania do pliku PDF przy użyciu biblioteki LibrePDF**

Dostępna jest także opcja zapisu wyników do pliku PDF. Wyniki prezentowane są w formie tabelarycznej, a dokument zawiera nagłówek oraz listę kandydatów wraz z uzyskanymi głosami. Format PDF umożliwia łatwe udostępnianie oraz drukowanie raportów wyborczych.

**7. Podsumowanie**

Projekt spełnia założone wymagania. System działa stabilnie, dane są bezpiecznie przechowywane, a funkcjonalności są zgodne z zakładaną specyfikacją.

A wszystko dostępne jest na **GitHub**’ie pod linkiem: <https://github.com/Rafimaniaak/wybory>

Repozytorium zawierające pełen kod źródłowy projektu, konfigurację oraz pliki SQL.

**8. Możliwości rozbudowy**

 Wprowadzenie autoryzacji dwuskładnikowej (2FA)

 Wersja webowa z użyciem Spring Boot + React

 Podłączenie do zewnętrznej bazy PostgreSQL

 Wersja mobilna – aplikacja dla Androida z JavaFX Mobile