

UNIwersYTET RZESZOWSKI
WYDZIAŁ NAUK ŚCIŚLYCH I TECHNICZNYCH
INSTYTUT INFORMATYKI



Yevhen Marchak
134945

Informatyka

Aplikacja do zarządzania wizytami lekarskimi

Praca projektowa

Praca wykonana pod kierunkiem
mgr inż. Ewa Żesławska

Rzeszów 2025

Spis treści

Streszczenie.....	7
Abstract.....	8
1. Opis założeń projektu	9
1.1. Cel projektu	9
1.2. Zakres projektu.....	9
1.3. Uzasadnienie wyboru tematu	9
1.4. Technologie	9
1.5. Struktura pracy	9
2. Analiza i projekt systemu.....	10
2.1. Opis funkcjonalny systemu	10
2.2. Diagram koncepcyjny bazy danych.....	11
2.3. Diagram klas (dziedziczenie)	11
2.4. Opis struktur danych.....	12
3. Harmonogram realizacji projektu	13
3.1. Etapy realizacji	13
3.2. Wizualizacja harmonogramu	13
4. Prezentacja warstwy użytkowej projektu.....	14
4.1. Okno powitalne.....	14
4.2. Panel logowania.....	14
4.3. Panel sekretariatu.....	15
4.4. Zarządzanie lekarzami.....	15
4.5. Panel pacjenta	16
4.6. Formularz danych pacjenta.....	16
4.7. Formularz rezerwacji wizyty	17
4.8. Tabela wizyt.....	17
4.9. Tworzenie danych logowania pacjenta	18
5. Implementacja.....	19
5.1. Struktura projektu.....	19
5.2. Technologie	19
5.3. Kluczowe klasy.....	19
5.4. Przykładowe fragmenty kodu	20
5.4.1. Klasa Doctor	20
5.4.2. Metoda logowania użytkownika	20
5.5. Obsługa wyjątków i walidacja.....	21
5.5.1. Niepoprawne dane logowania	21
5.5.2. Istniejący login podczas rejestracji	22

5.5.3. Konflikt terminów wizyt	22
6. Testowanie i podsumowanie	24
6.1. Testowanie systemu	24
6.1.1. Testy jednostkowe	24
6.1.2. Testy integracyjne	24
6.1.3. Testy użytkowe	24
6.2. Podsumowanie	24
Bibliografia	26
Spis rysunków	27
Spis listingów	28

Streszczenie

Celem niniejszego projektu było zaprojektowanie i zaimplementowanie aplikacji desktopowej do kompleksowego zarządzania wizytami lekarskimi. System umożliwia rejestrację i logowanie pacjentów oraz sekretariatu, przeglądanie i edycję danych pacjentów, lekarzy oraz historii wizyt.

Zastosowano język Java 17, bibliotekę Swing do stworzenia graficznego interfejsu użytkownika oraz bazę danych PostgreSQL z obsługą poprzez JDBC. Projekt uwzględnia warstwową architekturę aplikacji, walidację danych oraz obsługę wyjątków.

Efektom pracy jest funkcjonalna i intuicyjna aplikacja wspomagająca zarządzanie procesem umawiania wizyt w placówce medycznej.

Abstract

The aim of this project was to design and implement a desktop application for comprehensive management of medical appointments. The system allows for registration and login of patients and secretaries, as well as browsing and editing data of patients, doctors, and visit history.

The project uses Java 17, the Swing library for GUI creation, and PostgreSQL with JDBC for database handling. It follows a layered architecture and includes data validation and exception handling.

The result is a functional and user-friendly application that supports the scheduling of appointments in a medical facility.

1. Opis założeń projektu

1.1. Cel projektu

Celem niniejszego projektu było stworzenie aplikacji desktopowej umożliwiającej kompleksowe zarządzanie wizytami lekarskimi. Aplikacja pozwala na przegląd, dodawanie, edycję oraz usuwanie informacji o pacjentach, lekarzach i wizytach.

1.2. Zakres projektu

Projekt obejmuje implementację:

- interfejsu graficznego w technologii **Swing**,
- komunikacji z relacyjną bazą danych PostgreSQL przez **JDBC**,
- systemu autoryzacji użytkowników (sekretariat, pacjent),
- podstawowych operacji **CRUD** na danych,
- walidacji, filtrowania, sortowania i obsługi wyjątków.

1.3. Uzasadnienie wyboru tematu

Temat został wybrany ze względu na jego praktyczne zastosowanie oraz popularność w instytucjach medycznych. Projekt umożliwił studentowi rozwój umiejętności z zakresu projektowania aplikacji wielowarstwowych, obsługi baz danych oraz projektowania GUI.

1.4. Technologie

W projekcie zastosowano:

- język programowania **Java 17**,
- bibliotekę **Swing** do interfejsu graficznego,
- **PostgreSQL** jako system zarządzania bazą danych,
- narzędzie **LaTeX** do przygotowania dokumentacji.

1.5. Struktura pracy

Praca składa się z sześciu głównych rozdziałów:

- Opis założeń projektu,
- Opis struktury projektu,
- Harmonogram realizacji projektu,
- Prezentacja warstwy użytkowej projektu,
- Implementacja projektu,
- Testowanie i podsumowanie,

2. Analiza i projekt systemu

2.1. Opis funkcjonalny systemu

System zarządzania wizytami lekarskimi zapewnia funkcjonalność dla dwóch głównych typów użytkowników: **sekretariatu** oraz **pacjenta**. Każdy z użytkowników ma dostęp do dedykowanego panelu, zawierającego odpowiednie opcje i możliwości:

- **Sekretariat:**

- przeglądanie, dodawanie, edytowanie i usuwanie danych pacjentów,
- zarządzanie wizytami: aktualizacja statusu, dodawanie notatek lekarskich,
- przeglądanie oraz zarządzanie lekarzami.

- **Pacjent:**

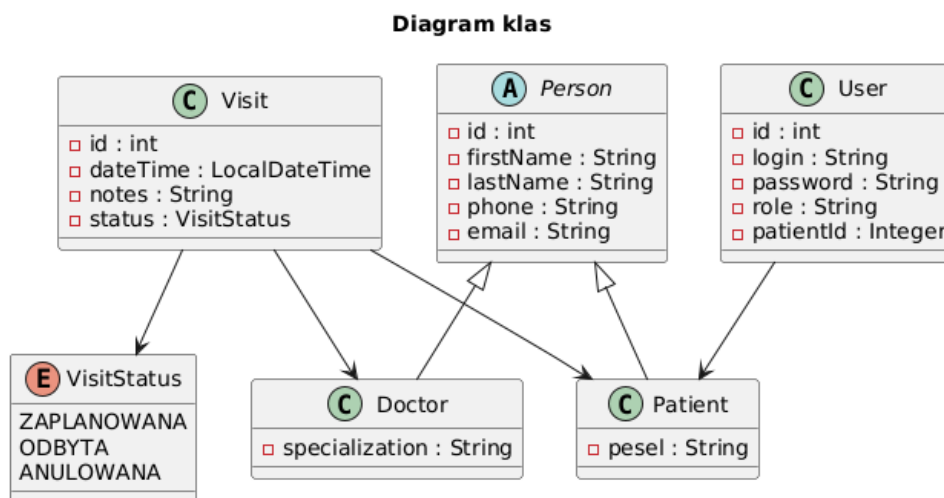
- przeglądanie danych osobowych,
- przeglądanie historii wizyt wraz z notatkami od lekarza,
- rezerwacja nowej wizyty, wybierając lekarza i termin.

2.2. Diagram koncepcyjny bazy danych



Rys. 2.1. Diagram koncepcyjny bazy danych

2.3. Diagram klas (dziedziczenie)



Rys. 2.2. Diagram klas przedstawiający hierarchię dziedziczenia

2.4. Opis struktur danych

System opiera się na czterech głównych encjach:

- **Pacjent** – dane identyfikacyjne (imię, nazwisko, PESEL, email, telefon),
- **Lekarz** – imię, nazwisko, email, telefon, specjalizacja,
- **Wizyta** – powiązanie lekarza z pacjentem, termin, status, notatka,
- **Użytkownik** – login, hasło, rola (sekretnarz lub pacjent), powiązanie z pacjentem (jeśli dotyczy).

3. Harmonogram realizacji projektu

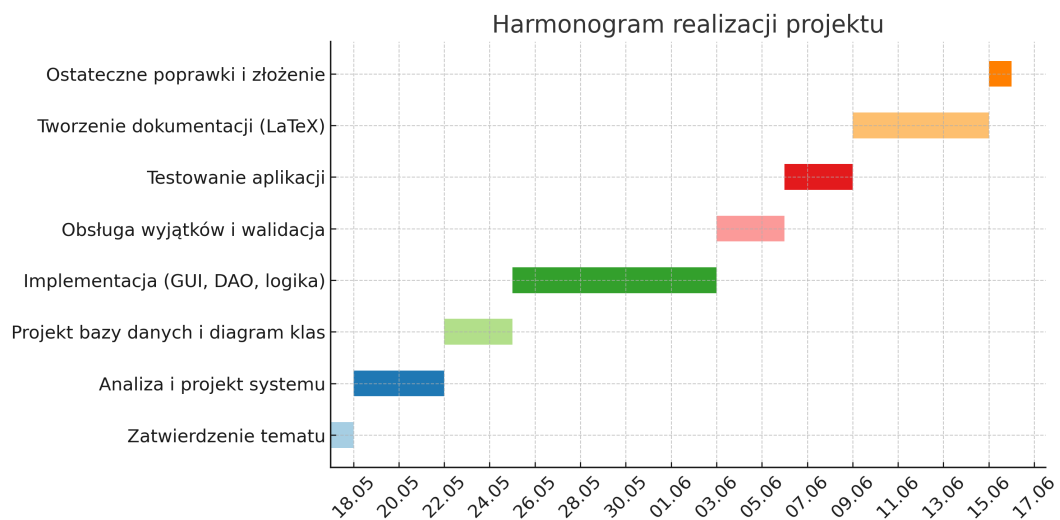
3.1. Etapy realizacji

Prace nad projektem zostały podzielone na następujące etapy:

- Zatwierdzenie tematu – 17.05.2025,
- Analiza i projekt systemu – 18.05–21.05.2025,
- Projekt bazy danych i diagram klas – 22.05–24.05.2025,
- Implementacja systemu (GUI, DAO, logika) – 25.05–02.06.2025,
- Obsługa wyjątków i walidacja – 03.06–05.06.2025,
- Testowanie aplikacji – 06.06–08.06.2025,
- Tworzenie dokumentacji – 09.06–14.06.2025,
- Ostateczne poprawki i złożenie – 15.06.2025.

3.2. Wizualizacja harmonogramu

Poniżej przedstawiono diagram Gantta obrazujący kolejne etapy realizacji projektu:



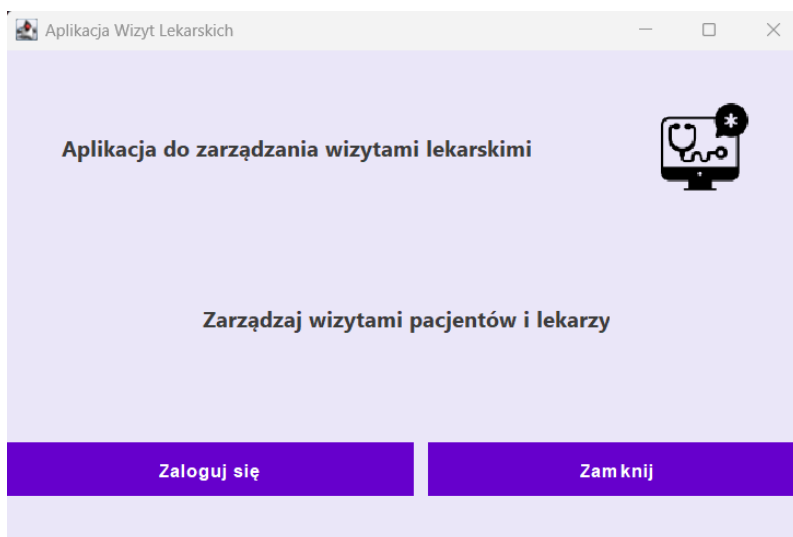
Rys. 3.1. Diagram Gantta – harmonogram realizacji projektu

4. Prezentacja warstwy użytkowej projektu

Graficzny interfejs użytkownika (GUI) został opracowany z myślą o prostocie obsługi, czytelności i intuicyjnej nawigacji. System udostępnia dwa główne panele: dla użytkownika typu **pacjent** oraz dla użytkownika typu **sekreтариат**. Poniżej zaprezentowano poszczególne widoki oraz ich funkcjonalność.

4.1. Okno powitalne

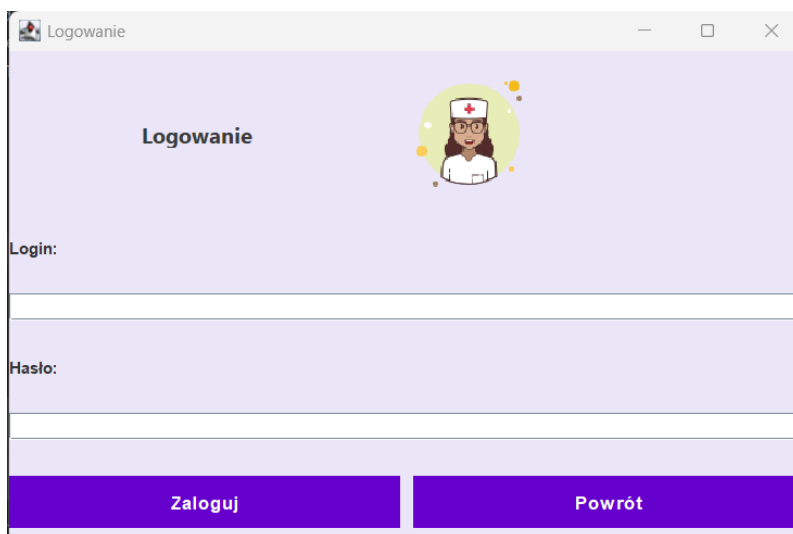
To pierwsze okno aplikacji, z którego użytkownik może przejść do logowania lub zakończyć program.



Rys. 4.1. Okno powitalne aplikacji

4.2. Panel logowania

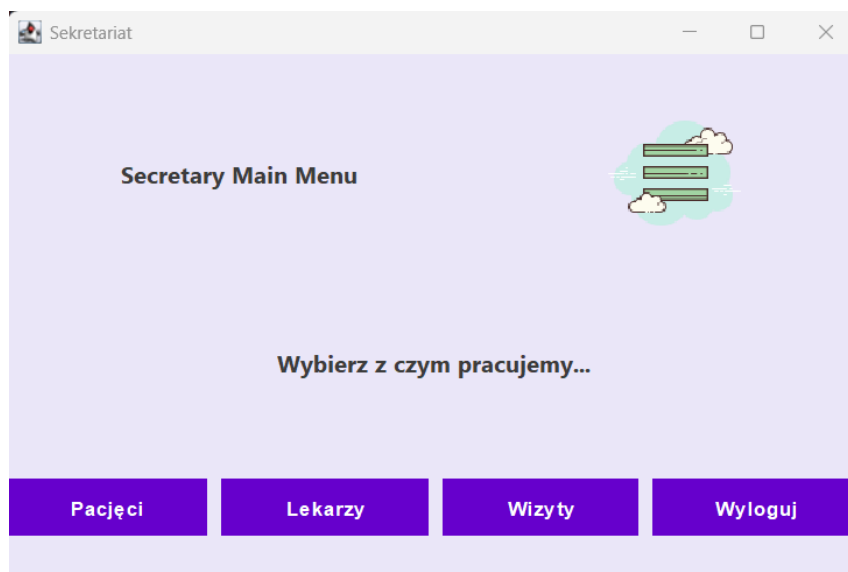
Panel ten zawiera pola do wpisania loginu i hasła oraz przycisk do zalogowania. W przypadku błędnych danych użytkownik otrzymuje stosowny komunikat.



Rys. 4.2. Ekran logowania

4.3. Panel sekretariatu

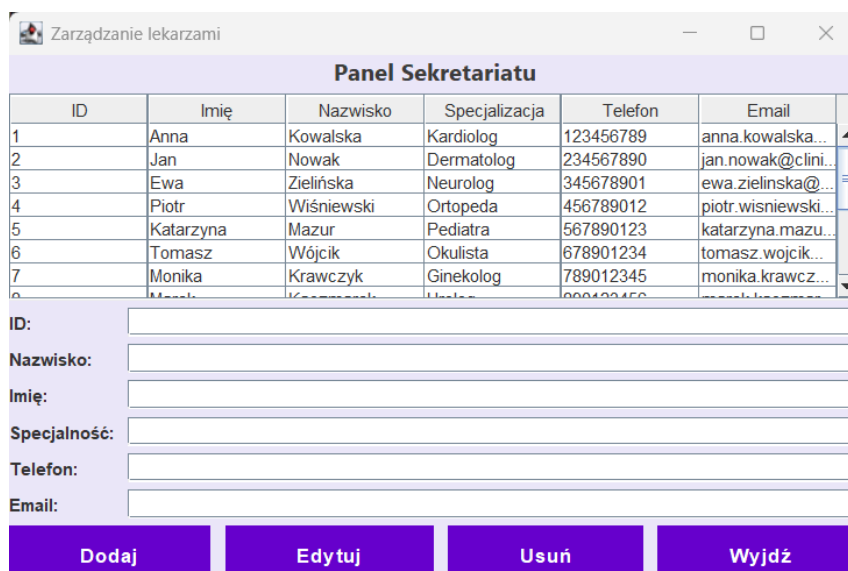
Po zalogowaniu się jako sekretarz użytkownik ma dostęp do głównego panelu zarządzania. Sekretariat może przeglądać, dodawać, edytować oraz usuwać dane pacjentów, lekarzy i wizyt.



Rys. 4.3. Główny panel sekretariatu

4.4. Zarządzanie lekarzami

Sekretariat posiada osobne okno do zarządzania lekarzami. W tym widoku możliwe jest przeglądanie listy wszystkich lekarzy oraz wykonywanie operacji CRUD (Dodaj, Edytuj, Usuń).



Rys. 4.4. Zarządzanie lekarzami – widok sekretariatu

4.5. Panel pacjenta

Po zalogowaniu się jako pacjent użytkownik ma dostęp do panelu z opcjami: „Moje dane”, „Moje wizyty”, „Zarezerwuj wizytę” oraz „Wyloguj”.



Rys. 4.5. Panel główny pacjenta

4.6. Formularz danych pacjenta

Panel „Moje dane” umożliwia pacjentowi podgląd danych osobowych, takich jak imię, nazwisko, PESEL, email oraz numer telefonu.

The screenshot shows a web application window titled "Moje dane". The main area has a light purple background. At the top, it says "Moje Dane" next to a green icon of a folder with a checkmark. Below this, there are five input fields with labels: "Imię:" (Rysz), "Nazwisko:" (Adrian), "PESEL:" (57481337627), "Email:" (ewabobik@o2.pl), and "Telefon:" (792 050 186). At the bottom, there is a blue button labeled "Powrót".

Rys. 4.6. Widok danych pacjenta

4.7. Formularz rezerwacji wizyty

W tym oknie pacjent może zarezerwować wizytę, wybierając lekarza z listy oraz wpisując dokładną datę i godzinę w formacie YYYY-MM-DD HH:mm.

Rys. 4.7. Formularz rezerwacji wizyty

4.8. Tabela wizyt

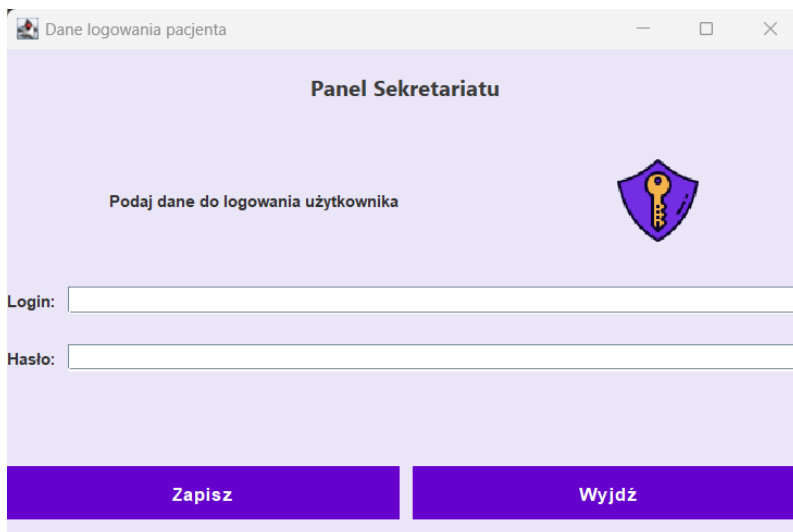
Widok tabelaryczny wizyt umożliwia przeglądanie istniejących wpisów, filtrowanie oraz odświeżanie zawartości.

ID	Data	Pacjent	Lekarz	Status	Notatka
2	2025-06-09T19:...	Adrian Rysz	Jan Nowak	ANULOWANA	Organ oddać wi...
4	2025-07-05T04:...	Monika Haremza	Katarzyna Mazur	ANULOWANA	Oprzeć towarzy...
5	2025-07-10T03:...	Eryk Pitek	Agnieszka Piotr...	ZAPLANOWANA	
6	2025-05-14T21:...	Agnieszka Koter...	Paweł Grabowski	ANULOWANA	Przeciw korzyst...
7	2025-07-02T04:...	Eryk Pitek	Piotr Wiśniewski	ODBYTA	Wątpliwość bez...
8	2025-06-01T21:...	Jerzy Balawend...	Rafał Szymański	ODBYTA	Działanie koron...
9	2025-07-02T12:...	Agnieszka Koter...	Natalia Dąbrow...	ODBYTA	Gra korzeń kost...
10	2025-07-07T08:...	Nataniel Igras	Monika Krawczyk	ANULOWANA	Francuski zaufa...
11	2025-05-15T11:...	Tadeusz Kamyk	Anna Kowalska	ZAPLANOWANA	
12	2025-05-20T21:...	Andrzej Bukowi...	Justyna Górską	ODBYTA	Chłopiec zwierz...
13	2025-06-11T05:...	Kazimierz Zadora	Paweł Grabowski	ZAPLANOWANA	
14	2025-05-23T01:...	Artur Kreczmer	Rafał Szymański	ANULOWANA	Wspólnota móż...
15	2025-05-25T13:...	Alex Indyk	Agnieszka Piotr...	ZAPLANOWANA	

Rys. 4.8. Tabela wizyt z funkcją odświeżania

4.9. Tworzenie danych logowania pacjenta

Podczas dodawania nowego pacjenta możliwe jest także automatyczne tworzenie konta użytkownika z przypisaną rolą.



The screenshot shows a web application window titled "Dane logowania pacjenta". The main content area has a light purple background and is titled "Panel Sekretariatu". Below the title, there is a prompt "Podaj dane do logowania użytkownika" and a shield icon with a key. There are two input fields: "Login:" and "Hasło:". At the bottom, there are two buttons: "Zapisz" (Save) and "Wyjdź" (Exit).

Rys. 4.9. Formularz do tworzenia konta pacjenta

5. Implementacja

W niniejszym rozdziale przedstawiono szczegóły implementacji systemu zarządzania wizytami lekarskimi, obejmujące strukturę projektu, zastosowane technologie oraz przykładowe fragmenty kodu źródłowego.

5.1. Struktura projektu

Projekt został zrealizowany zgodnie z podejściem warstwowym, obejmując następujące główne warstwy:

- **Warstwa danych** – odpowiedzialna za komunikację z bazą danych PostgreSQL z wykorzystaniem JDBC,
- **Warstwa logiki aplikacji** – zarządzająca procesami biznesowymi, walidacją danych oraz obsługą wyjątków,
- **Warstwa prezentacji** – implementująca graficzny interfejs użytkownika (GUI) w technologii Swing.

5.2. Technologie

W projekcie wykorzystano następujące technologie:

- **Język Java 17** – jako podstawowy język programowania,
- **Swing** – do stworzenia graficznego interfejsu użytkownika,
- **PostgreSQL** – jako system zarządzania bazą danych,
- **JDBC** – technologia do komunikacji z bazą danych,
- **LaTeX** – narzędzie do przygotowania dokumentacji.

5.3. Kluczowe klasy

Poniżej przedstawiono kluczowe klasy oraz ich role w systemie:

- **Doctor, Patient, Person** – reprezentujące podstawowe jednostki w systemie,
- **Visit** – zarządzająca danymi dotyczącymi wizyt,
- **User** – obsługująca użytkowników systemu, ich logowanie oraz autoryzację,
- **DoctorDAO, PatientDAO, VisitDAO** – klasy zapewniające dostęp do danych z bazy.

5.4. Przykładowe fragmenty kodu

5.4.1. Klasa Doctor

Listing 5.1. Klasa reprezentująca lekarza — rozszerzenie klasy Person

```
1 public class Doctor extends Person {
2     private String specialization;
3
4     public Doctor(int id, String firstName, String lastName, String phone, String
5     email, String specialization) {
6         super(id, firstName, lastName, phone, email);
7         this.specialization = specialization;
8     }
9
10    @Override
11    public String getInfo() {
12        return "Lekarz: " + firstName + " " + lastName + " - " + specialization;
13    }
14
15    public String getSpecialization() {
16        return specialization;
17    }
18
19    public void setSpecialization(String specialization) {
20        this.specialization = specialization;
21    }
22
23    @Override
24    public String toString() {
25        return firstName + " " + lastName + " - " + specialization;
26    }
27 }
```

5.4.2. Metoda logowania użytkownika

Listing 5.2. Metoda logowania użytkownika z wykorzystaniem JDBC

```
1 public User login(String login, String password) {
2     String sql = "SELECT * FROM users WHERE login = ? AND password = ?";
3
4     try (Connection conn = DatabaseConnection.getConnection();
5         PreparedStatement stmt = conn.prepareStatement(sql)) {
6
7         stmt.setString(1, login);
8         stmt.setString(2, password);
9
10        try (ResultSet rs = stmt.executeQuery()) {
11            if (rs.next()) {
12                return new User(
13                    rs.getInt("id"),
14                    rs.getString("login"),
15                    rs.getString("password"),
16                    rs.getString("role"),
```

```

17         rs.getObject("patient_id") != null ? rs.getInt("patient_id")
18         : null
19     );
19     }
20 }
21
22 } catch (SQLException e) {
23     e.printStackTrace();
24 }
25 return null; //login lub hasło nie prawidłowe
26 }

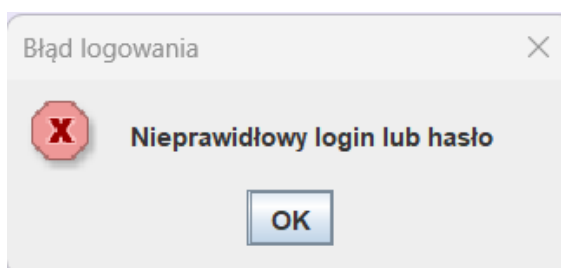
```

5.5. Obsługa wyjątków i walidacja

Aplikacja posiada zaimplementowaną obsługę wyjątków, mającą na celu zapewnienie stabilności oraz poprawności działania systemu. Poniżej przedstawiono przykładowe wyjątki wraz z fragmentami kodu oraz komunikatami wyświetlanymi użytkownikowi.

5.5.1. Niepoprawne dane logowania

W przypadku próby logowania przy użyciu błędnego loginu lub hasła wyświetlany jest odpowiedni komunikat błędu (Rys. 5.1).



Rys. 5.1. Komunikat błędu - niepoprawne dane logowania

Przykładowy fragment kodu:

Listing 5.3. Obsługa błędnego logowania — komunikat dla użytkownika

```

1  if (user != null) {
2
3      dispose();
4
5      if (user.isSecretary()) {
6          new SecretaryMainMenu().setVisible(true);
7      } else if (user.isPatient()) {
8          PatientDAO patientDAO = new PatientDAO();
9          Patient patient = patientDAO.getPatientById(user.getPatientId());
10         if (patient != null) {
11             new PatientMainMenu(patient).setVisible(true);
12         } else {
13             JOptionPane.showMessageDialog(null, "Błąd: nie znaleziono
14             danych pacjenta!", "Błąd", JOptionPane.ERROR_MESSAGE);
15         }
16     } else {

```

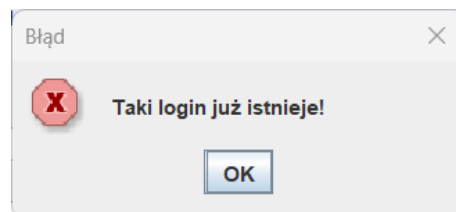
```

17         JOptionPane.showMessageDialog(null, "Nieprawidłowy login lub hasło",
18         "Błąd logowania", JOptionPane.ERROR_MESSAGE);
    }

```

5.5.2. Istniejący login podczas rejestracji

Przy próbie utworzenia nowego użytkownika z loginem, który już istnieje w bazie danych, aplikacja wyświetli stosowny komunikat (Rys. 5.2).



Rys. 5.2. Komunikat błędu - istniejący login

Przykładowy fragment kodu:

Listing 5.4. Walidacja unikalności loginu podczas rejestracji użytkownika

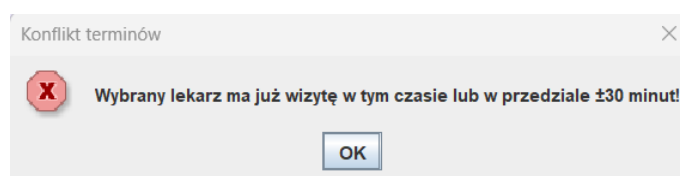
```

1  public boolean loginExists(String login) {
2      String sql = "SELECT 1 FROM users WHERE login = ?";
3      try (Connection conn = DatabaseConnection.getConnection();
4           PreparedStatement stmt = conn.prepareStatement(sql)) {
5          stmt.setString(1, login);
6          ResultSet rs = stmt.executeQuery();
7          return rs.next();
8      } catch (SQLException e) {
9          e.printStackTrace();
10     }
11     return false;
12 }
13 /-----\
14
15 if (dao.loginExists(login)) {
16     JOptionPane.showMessageDialog(null, "Taki login już istnieje!", "Błąd",
17     JOptionPane.ERROR_MESSAGE);
18     return;
19 }

```

5.5.3. Konflikt terminów wizyt

Próba rezerwacji wizyty u lekarza w terminie, który koliduje z inną wizytą (z marginesem ± 30 minut), powoduje wyświetlenie komunikatu o konflikcie (Rys. 5.3).



Rys. 5.3. Komunikat błędu - konflikt terminów wizyt

Przykładowy fragment kodu:

Listing 5.5. Walidacja konfliktów wizyt ± 30 minut u danego lekarza

```
1 public boolean hasDoctorConflict(int doctorId, LocalDateTime dateTime) {
2     String sql = "SELECT COUNT(*) FROM visits WHERE doctor_id = ? AND ABS(EXTRACT(EPOCH
3         FROM (visit_date - ?))) < 1800";
4
5     try (Connection conn = DatabaseConnection.getConnection();
6         PreparedStatement stmt = conn.prepareStatement(sql)) {
7
8         stmt.setInt(1, doctorId);
9         stmt.setTimestamp(2, Timestamp.valueOf(dateTime));
10
11         ResultSet rs = stmt.executeQuery();
12         if (rs.next()) {
13             return rs.getInt(1) > 0;
14         }
15     } catch (SQLException e) {
16         e.printStackTrace();
17     }
18     return false;
19 }
```

Powyższe przykłady stanowią tylko wybrane przypadki obsługi wyjątków w aplikacji. W projekcie zaimplementowano także inne wyjątki, takie jak błędy połączenia z bazą danych, walidacja danych wejściowych i inne komunikaty błędów użytkownika.

6. Testowanie i podsumowanie

6.1. Testowanie systemu

Testowanie aplikacji odbyło się na kilku poziomach: jednostkowym, integracyjnym oraz użytkowym.

6.1.1. Testy jednostkowe

Testy jednostkowe skupiały się na poprawnym działaniu poszczególnych metod klas modelu oraz DAO. Szczególną uwagę zwrócono na poprawność:

- walidacji danych wejściowych,
- działania metod CRUD,
- obsługi wyjątków bazodanowych i walidacyjnych.

Wszystkie testy jednostkowe zostały przeprowadzone ręcznie poprzez sprawdzanie poszczególnych funkcjonalności w środowisku IDE (IntelliJ IDEA).

6.1.2. Testy integracyjne

Testy integracyjne dotyczyły komunikacji między warstwą aplikacji (DAO) a bazą danych PostgreSQL. Testowano poprawność zapytań SQL oraz operacji CRUD realizowanych z poziomu aplikacji. Wszystkie operacje (dodawanie, edycja, usuwanie, filtrowanie) przebiegły pomyślnie, a wyniki testów potwierdziły poprawne funkcjonowanie aplikacji.

6.1.3. Testy użytkowe

Testy użytkowe zostały przeprowadzone z punktu widzenia końcowego użytkownika aplikacji. Testy obejmowały:

- logowanie i autoryzację użytkowników,
- rezerwację wizyt z walidacją konfliktów terminów,
- zarządzanie danymi pacjentów, lekarzy i wizyt przez sekretariat,
- obsługę sytuacji wyjątkowych (np. błędny login lub zajęty termin wizyty).

Aplikacja wypadła pozytywnie, co potwierdziło intuicyjność oraz poprawność działania wszystkich zaimplementowanych funkcjonalności.

6.2. Podsumowanie

Głównym celem projektu było stworzenie funkcjonalnej aplikacji umożliwiającej zarządzanie wizytami lekarskimi. Projekt został zrealizowany zgodnie z wymaganiami, z wykorzystaniem technologii Java oraz Swing, z relacyjną bazą danych PostgreSQL. Zastosowano podejście obiektowe, które zapewniło czytelność oraz łatwość utrzymania kodu.

W przyszłości projekt może zostać rozbudowany o dodatkowe funkcjonalności, takie jak integracja z kalendarzem internetowym czy możliwość eksportu danych do popularnych formatów (np. CSV lub XLS).

Kod źródłowy projektu jest dostępny publicznie w repozytorium:

`github.com/YevhenMarchak/Aplikacja_do_zarzadzania_wizytami_lekarskimi`

Bibliografia

- [1] Cay S. Horstmann. *Core Java Volume I – Fundamentals*. Pearson Education, 12th edition, 2022.
- [2] Regina O. Obe and Leo S. Hsu. *PostgreSQL: Up and Running*. O'Reilly Media, 3rd edition, 2017.
- [3] Matthew Robinson and Pavel Vorobiev. *Swing*. Apress, 2nd edition, 2003.
- [4] Herbert Schildt. *Java: The Complete Reference*. McGraw-Hill Education, 12th edition, 2021.

Spis rysunków

2.1	Diagram koncepcyjny bazy danych	11
2.2	Diagram klas przedstawiający hierarchię dziedziczenia	11
3.1	Diagram Gantta – harmonogram realizacji projektu	13
4.1	Okno powitalne aplikacji	14
4.2	Ekran logowania	14
4.3	Główny panel sekretariatu	15
4.4	Zarządzanie lekarzami – widok sekretariatu	15
4.5	Panel główny pacjenta	16
4.6	Widok danych pacjenta	16
4.7	Formularz rezerwacji wizyty	17
4.8	Tabela wizyt z funkcją odświeżania	17
4.9	Formularz do tworzenia konta pacjenta	18
5.1	Komunikat błędu - niepoprawne dane logowania	21
5.2	Komunikat błędu - istniejący login	22
5.3	Komunikat błędu - konflikt terminów wizyt	22

Spis listingów

5.1	Klasa reprezentująca lekarza — rozszerzenie klasy Person	20
5.2	Metoda logowania użytkownika z wykorzystaniem JDBC	20
5.3	Obsługa błędnego logowania — komunikat dla użytkownika	21
5.4	Walidacja unikalności loginu podczas rejestracji użytkownika	22
5.5	Walidacja konfliktów wizyt ± 30 minut u danego lekarza	23

Załącznik nr 2 do Zarządzenia nr 228/2021 Rektora Uniwersytetu Rzeszowskiego z dnia 1 grudnia 2021 roku w sprawie ustalenia procedury antyplagiatowej w Uniwersytecie Rzeszowskim

OŚWIADCZENIE STUDENTA O SAMODZIELNOŚCI PRACY

..... Yevhen Marchak

Imię (imiona) i nazwisko studenta

Wydział Nauk Ścisłych i Technicznych

..... Informatyka

Nazwa kierunku

..... 134945

Numer albumu

1. Oświadczam, że moja praca projektowa pt.: Aplikacja do zarządzania wizytami lekarskimi

- 1) została przygotowana przeze mnie samodzielnie,
- 2) nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2021 r., poz. 1062) oraz dóbr osobistych chronionych prawem cywilnym,
- 3) nie zawiera danych i informacji, które uzyskałem/am w sposób niedozwolony,
- 4) nie była podstawą otrzymania oceny z innego przedmiotu na uczelni wyższej ani mnie, ani innej osobie.

2. Jednocześnie wyrażam zgodę na udostępnienie mojej pracy projektowej do celów naukowo-badawczych z poszanowaniem przepisów ustawy o prawie autorskim i prawach pokrewnych.

Rzeszów 15.06.2025

(miejscowość, data)

Yevhen Marchak

(czytelny podpis studenta)