Appointments made easy

Spis treści

Wprowadzenie

Celem aplikacji projektowej jest umożliwienie lekarzom sprawnego zarządzania pacjentami, których mają pod opieką, oraz wizytami, które planują. Aplikacja umożliwia dodawanie, edytowanie oraz usuwanie pacjentów oraz wizyt z bazy danych.

Strona techniczna

Projekt został wykonany w języku programowania Java w środowiskach IntelliJ IDEA i Visual Studio Code. Do stworzenia interfejsu graficznego wykorzystana została biblioteka JavaFX oraz program SceneBuilder do jego zaprojektowania. W aktualnej wersji aplikacja zawiera 21 klas oraz 11 plików FXML odpowiadających oknom interfejsu. Stworzone klasy to:

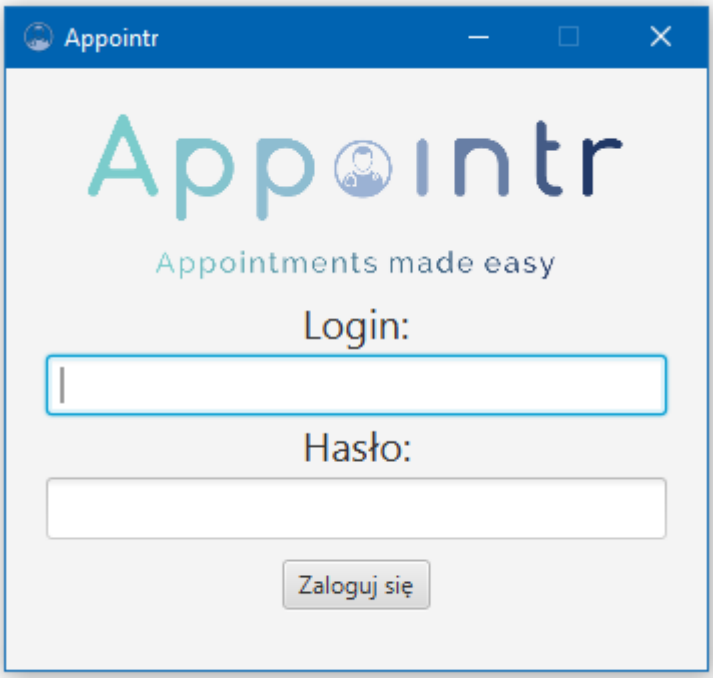
- App – główna klasa odpowiadająca za uruchamianie aplikacji i (obecnie nieużywany) interfejs konsolowy
- Appointment – klasa obsługująca obiekty wizyty

- Person – klasa będąca bazą dla klas odnoszących się do ludzi
- Patient – klasa dziedzicząca po Person, obsługuje obiekty pacjentów
- Doctor - klasa dziedzicząca po Person, obsługuje obiekty lekarzy
- Klasy […]Controller – klasy obsługujące dane okna interfejsu graficznego
- Populate – klasa służąca do tworzenia testowej grupy ludzi do używania w programie
- AppointmentTableRow – klasa służąca do wypełniania tabeli wizyt w interfejsie graficznym
- PatientTableRow – klasa służąca do wypełniania tabeli pacjentów w interfejsie graficznym
- DoctorTableRow – klasa służąca do wypełniania tabeli lekarzy w interfejsie graficznym (funkcjonalność tylko dla administratora)

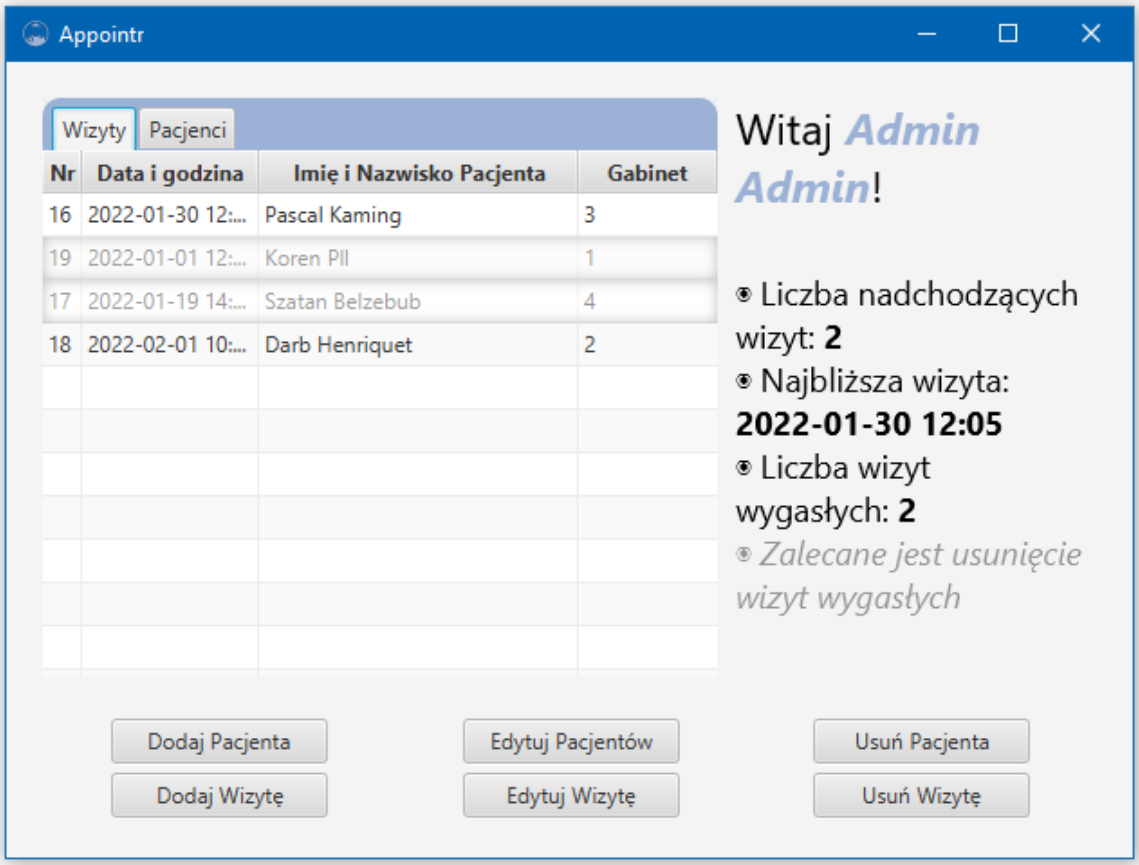
Dane potrzebne do funkcjonowania programu są przechowywane w bazie danych na serwerze Galera Instytutu Informatyki Politechniki Warszawskiej. Baza danych zawiera 12 tabeli, których atrybuty i relacje są zawarte w pliku ModelER.pdf w folderze sql tego repozytorium. W bazie danych znajdują się również funkcje calculate_age, której zadaniem jest wyliczenie wieku lekarza lub pacjenta, calculate_payment, która wylicza zapłatę dla lekarza na bazie wizyt oraz procedury drop_unassigned_patients, która pozbywa się z bazy danych pacjentów, którzy nie są przypisani do żadnej wizyty i delete_late appointments, która usuwa wizyty, których data poprzedza datę obecną.
W bazie danych znajdują się też wyzwalacze tg_add_id_app, który nadaje id wizyty kolejną liczbę całkowitą oraz wyzwalacz BI_GENERATE_PESEL, który generuje PESEL. Baza danych współpracuje z programem w Javie, głównie za pomocą działania klasy DBContext, która zawiera funkcje korzystające z połączenia i wysyła zapytania SQL.

Instrukcja obsługi

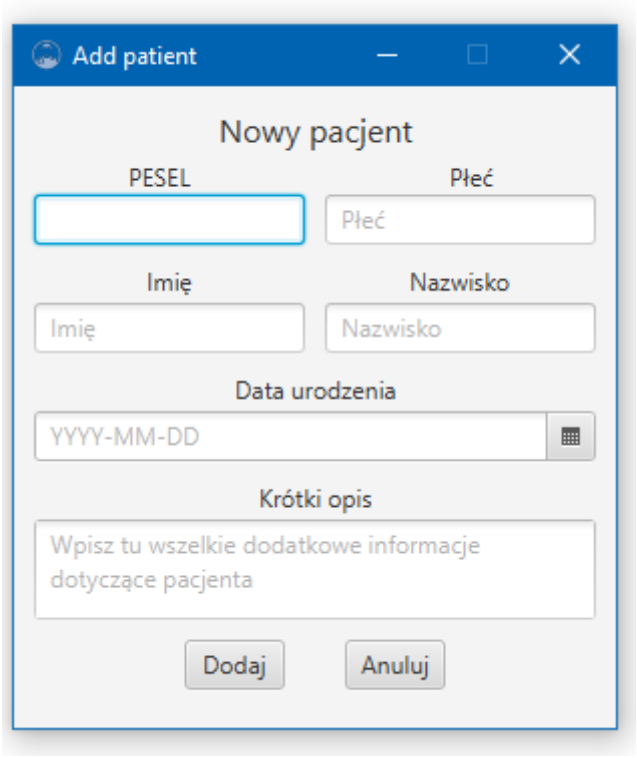
Aby rozpocząć korzystanie z aplikacji należy zalogować się używając loginu i hasła przypisanego do danego lekarza. Login i hasło należy wprowadzić do odpowiadających im pól, a następnie nacisnąć przycisk „Zaloguj się”.

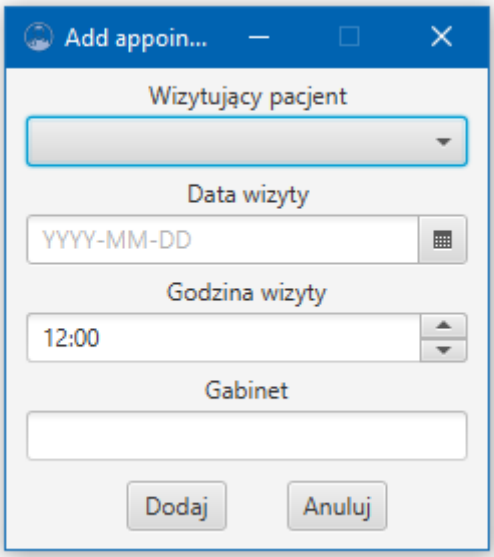


Po zalogowaniu się do systemu, dyspozycyjny staje się panel główny zawierający główne funkcjonalności oprogramowania. Tabela wyświetla wizyty, które są przypisane do zalogowanego doktora. Te z wizyt, które są w kolorze jasno szarym uznawane są za wygasłe, gdyż ich termin zapisany w bazie danych się przedawnił. W zakładce „Pacjenci” znajdują się kluczowe informacje dotyczące pacjentów wpisanych do bazy danych. Klikanie na nagłówki kolumn pozwala na sortowanie wierszy po zawartości danych kolumn. Przyciski umieszczone na dole okna pozwalają na dodawanie, edytowanie i usuwanie poszczególnych pacjentów oraz wizyt.

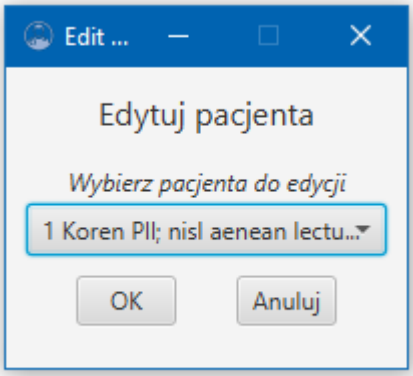


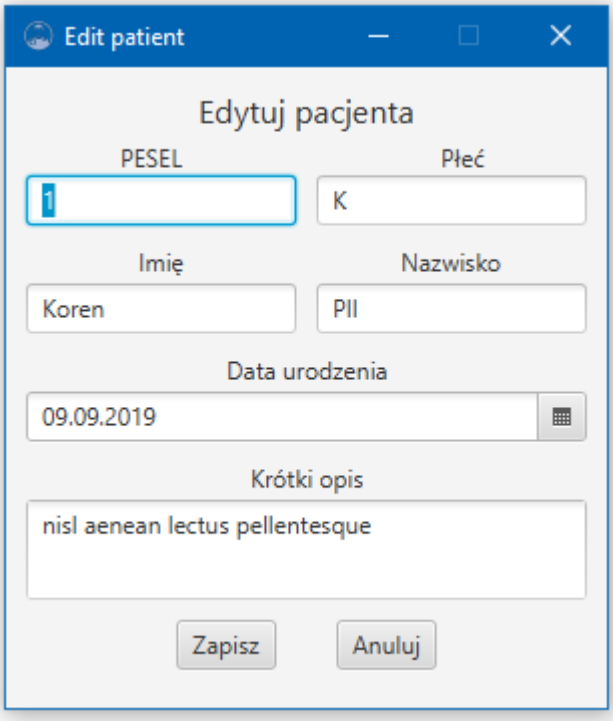
Aby dodać pacjenta należy kliknąć przycisk „Dodaj pacjenta”, wypełnić pojawiający się kwestionariusz oraz zatwierdzić przyciskiem „Dodaj”. Akcję tą można anulować odpowiednim przyciskiem. W analogiczny sposób można postąpić w przypadku dodawania wizyty.

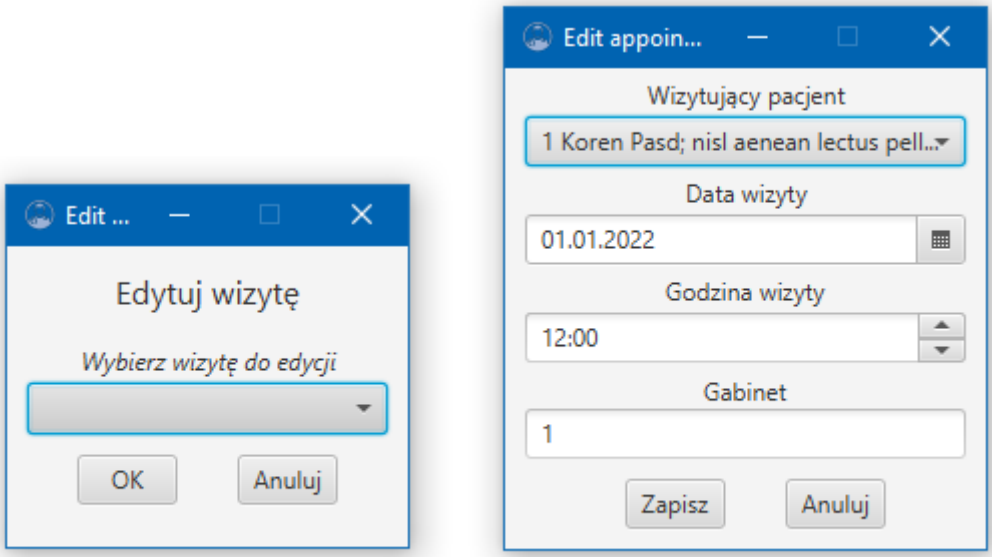




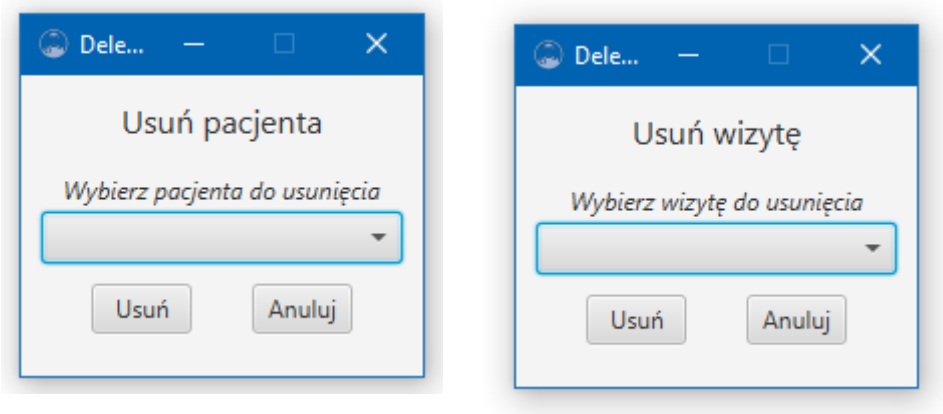
W celu edycji pacjentów i wizyt należy najpierw wybrać pacjenta, który będzie edytowany z listy, nacisnąć przycisk „OK”, a następnie przystąpić do edycji w pojawiającym się kolejnym oknie. Analogicznie należy postąpić w przypadku edycji wizyt.







Usuwanie wizyt odbywa się poprzez wybór pacjenta lub wizyty z listy i naciśnięcie przycisku „Usuń”.



Testowanie

Testowanie odbywało się poprzez wykonywanie czynności w programie i porównywaniu ich z efektami, które były oczekiwane w bazie danych oraz poprzez kontrolę zachowania interfejsu graficznego. Kod był również na bieżąco sprawdzany przez wszystkich uczestników projektu w ramach krzyżowej weryfikacji, aby uniknąć potencjalnych błędów i niedociągnięć.

Zasługi indywidualne

Bartłomiej Dudek:

- Stworzenie repozytorium
- Stworzenie koncepcji interfejsu graficznego
- Utworzenie klas Doctor, Patient, Appointment
- Dodanie do projektu bibliotek do obsługi plików FXML
- Stworzenie konsolowego interfejsu
- Stworzenie elementów interfejsu graficznego służących do dodawania, edytowania, usuwania wizyt i pacjentów oraz ekranu głównego
- Dodanie obsługi wyżej wymienionych elementów do aplikacji
- Ustawienie przekazywania danych pomiędzy wszystkimi elementami interfejsu graficznego
- Dodanie połączenia z bazą danych do niektórych elementów interfejsu graficznego
- Stworzenie dokumentacji

Jakub Marcowski:

- Stworzenie pierwszej wersji projektu, która umożliwiła wszystkim członkom zespołu sprawniejsze wdrożenie się w system plików oraz technologie użyte w aplikacji
- Rozbudowanie klas reprezentujących ludzi, pacjentów, lekarzy oraz wizyty
- Napisanie docstringów do powyższych klas
- Stworzenie logo oraz ikony aplikacji (wykorzystana paleta barw wybrana przez wszystkich członków zespołu)
- Naprawa bugów spoza przydzielonej części aplikacji (np. łączność z bazą danych)
- Stworzenie funkcji do ładnego wyświetlania tekstu w ramach CLI

- Aktualizowanie plików FXML tworzonych przez innego członka zespołu (zmiany przede wszystkim wizualne, ale czasami również zmiana wykorzystywanych bloków JavaFX użytych w ramach danego okna)
- Liczne refaktoryzacje kodu w celu usprawnienia jego czytania i w ramach trzymania się pewnej konwencji nazewnictwa
- Stworzenie okna ładowania (wraz z animacją w formacie .gif)
- Stworzenie klasy Populate, dzięki której można było testować działanie prototypu aplikacji na prawdziwych danych (które jeszcze nie pochodziły z bazy danych, ale były tworzone lokalnie)
- Dodanie funkcjonalności poprawnego wyświetlania danych wewnątrz tabelek (z użyciem JavaFX'owego TableView oraz TableRow), wykorzystując zakładki do przełączania się między tabelkami
- Dodanie przydatnych informacji o zalogowanym lekarzu i nadchodzących wizytach obok tabelki
- Wyeksportowanie aplikacji do pliku .exe
- Dodanie funkcjonalności trybu ciemnego
- Dodanie funkcjonalności zmiany języka między polskim a angielskim
- Stworzenie okna adminView, będącego doctorView rozbudowanym o dodatkowe funkcje przysługujące administratorowi
- Zaprojektowanie (koncepcyjne) modelu relacyjnej bazy danych użytej w projekcie
- Redakcja README.md oraz dokumentów i obrazów zawartych w repozytorium
- Zredagowanie dokumentacji

Jakub Niezabitowski:

- Implementacja struktur bazy danych, wykorzystanych do realizacji aplikacji
- Implementacja rekordów bazy danych, wykorzystanych w aplikacji

- Implementacja mechanizmów zabezpieczających przed wprowadzeniem niewłaściwych danych do bazy
- Empiryczne testowanie GUI

Jan Rusak:

- Implementacja klasy DBContext odpowiedzialnej za połączenie oraz modyfikacje bazy danych.
- Oprogramowanie przycisków aplikacji.
- Dodanie funkcjonalności odświeżania list wizyt i pacjentów po ich modyfikacji.