Politechnika Śląska
Wydział Automatyki, Elektroniki i Informatyki

System obsługi przychodni

Sprawozdanie z projektu zaliczeniowego realizowanego w ramach przedmiotu Programowanie Komputerów 4

Autor: Błażej Wylęgły

Prowadzący: mgr. Inż. Krzysztof Pasterak

Rok akademicki: 2019/2020

Kierunek: Informatyka

Rodzaj studiów: SSI

Semestr IV

Termin laboratorium: czwartek, 10:15 – 11:45

Sekcja 2

Data oddania sprawozdania 2020-06-19

1. Temat

Zaprojektowana aplikacja jest uproszczonym modelem systemu obsługującego przychodnię medyczną. Jej działanie symuluje realizację podstawowych zadań wynikających z prowadzenia i nadzorowania działania placówki medycznej.

2. Analiza zadania

Przed przystąpieniem do faktycznego tworzenia aplikacji, przeprowadzona została analiza problemu.

Jej celem, było stworzenie uproszczonego diagramu najważniejszych klas program, określenie najważniejszych funkcjonalności, które powinna dostarczać aplikacja oraz zdefiniowanie sposobu, w jaki powinny być realizowane poszczególne zadania.

W celu zapewnienia wygodnego sposobu korzystania z aplikacji, do systemu został wdrożony interfejs graficzny, który został podzielony na ekrany odpowiadające różnym rodzajom użytkowników oraz zakładki, które reprezentują poszczególne zagadnienia.

Ze względu na określone wymagania oraz zaproponowane sposoby ich realizacji, do stworzenia aplikacji został wykorzystany C++ QML, będący zestawem narzędzi frameworka QT.

Naturalną potrzebą wynikającą z tematyki problemu, jest konieczność przechowywania danych w zewnętrznym źródle. Ze względu na liczne powiązania, występujące pomiędzy poszczególnymi klasami reprezentującymi rzeczywistość, do magazynowania informacji została wykorzystana baza danych MySql.

- **Struktury danych**

W programie wykorzystane zostały m.in. kontenery biblioteki standardowej C++ oraz kontenery framework'u QT, których działanie jest bliźniacze do standardowych.

  - **vector** – kontener wykorzystywany do przechowywania obiektów modelu, takich jak pacjenci, lekarze, zabiegi, konta
  - **list** – (m.in. w implementacji frameworka QT) kontener wykorzystywany do eksportowania listy obiektów modelu do klas-kontrolerów, obsługujących interfejs graficzny

- **Biblioteki, narzędzia oraz technologie zewnętrzne**
  - **QML –** język programowania, który umożliwia tworzenie rozbudowanych interfejsów graficznych. Dodatkowo, jest to język
  - **Framework QT –** framework dostarczający wiele klas, które ułatwiają wykorzystanie klas standardowych, poprzez np. ich rozszerzenie.

- **Klasy główne**
  - **Person, Doctor, MedicalProcedure** – klasy odpowiadające za odwzorowanie podstawowych w kontekście zagadnienia klas – pacjenta, doktora oraz zabiegów.

  - **AccountData** – klasa reprezentująca użytkownika aplikacji. Przechowuje podstawowe dane dotyczące konta. W obrębie użytkowników, wyróżniamy administratora oraz normalnego użytkownika.

  - **Clinic** – klasa główna, odpowiedzialna za sprawowanie pieczy nad całą obsługą logiki aplikacji

- **DAO** – generyczny interfejs operujący na klasach implementujących (również generyczny) interfejs SqlAdapter. Zadaniem konkretnej implementacji DAO jest obsługa danej encji występującej w bazie danych – wyróżniamy PatientDAO, AccountDAO, etc.

- **Organiser** – klasa odpowiedzialna za zebranie konkretnych implemtacji interfejsu DAO. Jest swoistym interfejsem pomiędzy zewnętrzem logiki a implementacją warstwy dostępu do bazy danych.

- **Klasy typu „Controller”** – klasy odpowiedzialne za szeroko rozumianą kontrolę interfejsu graficznego. Ich działanie dotyczy również przekształcania klas modelu w obiekty, które mogą zostać zaprezentowane w obrębie interfejsu.

3. Specyfikacja zewnętrzna

Po uruchomieniu aplikacji wyświetlany jest ekran logowania. W przypadku pomyślnej próby logowania do istniejącego w bazie konta, następuje przeniesienie do panelu użytkownika lub panelu administratora.

W obrębie panelu użytkownika, dostępne są zakładki umożliwiające wykonanie takich operacji jak rejestracja nowego pacjenta, usunięcie i edycja istniejącego pacjenta, wyświetlenie listy pacjentów, umówienie nowego zabiegu, przełożenie lub odwołanie już istniejącego.

W obrębie kompetencji administratora, istnieje również zarządzanie spisem lekarzy wchodzących w skład personelu kliniki oraz zarządzanie kontami.

4. Specyfikacja wewnętrzna

Aplikacja została zaimplementowana zgodnie z paradygmatem programowania obiektowego oraz z wykorzystaniem wzorca projektowego MVC, którego zadaniem jest odseparowanie od siebie warstw modelu, widoku oraz kontroli (model-view-controller)

1. Tabele bazy danych

- accounts – tabela przechowująca dane kont – loginy, zahashowane hasła, unikalne, losowo generowane „salty” oraz informację czy dane konto jest kontem administratora

- patients – tabela przechowująca dane personalne pacjentów oraz informacje na temat lekarza prowadzącego danego pacjenta

- doctors – tabela przechowująca dane personalne lekarzy oraz informacje dotyczące organizacji ich pracy (godzina rozpoczęcia, godzina zakończenia)

- procedures – tabela przechowująca informacje dotyczące zabiegów wykonywanych w przychodni – zarówno już zrealizowanych, jak i tych zaplanowanych

2. Wykorzystane techniki obiektowe

- Dziedziczenie
- Polimorfizm
- Interfejsy
- Programowanie generyczne

3. Wykorzystane zagadnienia technologiczne z zajęć laboratoryjnych

- Wyrażenia regularne
- Inteligentne wskaźniki
- Wzorce
- Kontenery i iteratory STL

4. Uproszczony diagram klas

Diagram klas umieszczony został pośród innych materiałów wstępnych dotyczących projektu, na platformie GitHub, do której link znajduje się na końcu tego dokumentu.

5. Testowanie

W trakcie tworzenia aplikacji, prowadzone były niezależne od innych modułów testy poszczególnych funkcjonalności oraz poprawność modeli.

Największe trudności dotyczyły poprawnego przekształcenia modelu w obiekty, które należało zaprezentować w ramach interfejsu graficznego („podpięcie” kontrolerów do poszczególnych widoków oraz dostarczenie odpowiednych danych do kontrolerów).

6. Wnioski

Podczas tworzenia projektu, uświadomiłem sobie jak ważne są etap przygotowania do tworzenia aplikacji oraz testowanie poszczególnych modułów.

Opisywany projekt, był moim pierwszym zetknięciem z tworzeniem interfejsów graficznych. Na to zagadnienie poświęciłem najwięcej czasu. Niewątpliwie, to w tym zakresie poszerzyłem swoje umiejętności w największym stopniu.

Niezależne testy elementów, oraz przemyślenie problemu przed jego rozwiązywaniem, zapobiegło ewentualnym błędom na etapie późniejszych pracy, co z pewnością przyśpieszyło proces tworzenia oprogramowania.

Niezastąpionym narzędziem okazał się system kontroli wersji GIT, który umożliwił m.in. zabezpieczenie już działającej wersji projektu.

Znacząco rozszerzyłem swoje umiejętności w zakresie programowania w języku C++ oraz projektowania aplikacji. Dodatkowo, miałem okazję poznać popularny framework QT oraz język QML.

7. Rozwijanie aplikacji

Nie wszystkie przewidziane przeze mnie funckjonalności aplikacji zostały wdrożone – spowodowane jest to ilością czasu, którą musiałem poświęcić na naukę projektowania i implementacji interfejsu graficznego.

Nie mniej jednak, wszystkie podstawowe funkcjonalności zostały zaimplementowane.

Aplikacja będzie rozwijana dalej na potrzeby własne.

8. Link do repozytorium w platformie GitHub

https://github.com/bwylegly/CMS