1.Apka kliencka

Główna klasa aplikacji musi zawierać:

- Klasę głównego okna aplikacji którego używał będzie użytkownik (opis klasy w 1.1)
- Klasę do komunikacji z serwerem i interpretacją poleceń (opis w 1.2)
- Klasa przechowująca informację o aktualnie zalogowanym użytkowniku – ustali się jakie parametry są potrzebne, na razie trzeba postawić program „mechanicznie”

Klasa aplikacji dziedziczy klasę QApplication (mogę się zająć jej przygotowaniem)

1.1. Główna klasa okna

Klasa będzie zawierać :

- pasek komunikatów (już mam gotową klasę tylko trzeba ją dostosować do naszych potrzeb, udostępnię na GitHubie),
- pasek menu (QMenuBar + QMenu + QAction),
- pasek narzędzi,
- pasek stanu (QStatusBar),
- Pasek stanu połączenia z serwerem – klasa dziedzicząca po QLabel albo QFrame i dodać do tego tekst + ikonę pokazującej stan serwera,
- union przechowujący klasy stanów programu (opis 1.1.1)

1.1.1 Stany okna głównego

Główna klasa okna będzie miała 3 główne stany (osobne klasy których wskaźniki będą zapisane w unionie).

Stan 1: stan logowania użytkownika - dziedziczy jak każdy element okna po QFrame albo QWidget (Wasz wybór), a zawiera:

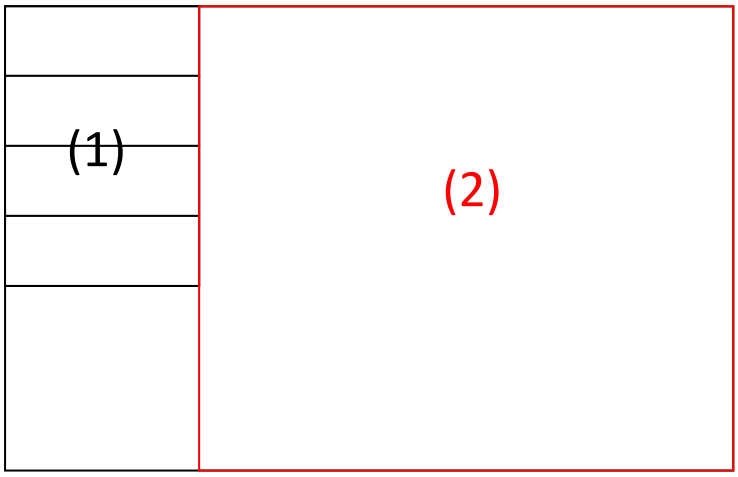
- Informacje dla użytkownika co się dzieje, czyli że ma się zalogować lub zarejestrować konto(na początek tekstową, jak nie będzie jakoś super wyglądać to się zrobi jakieś z obrazkiem)
- Linie edycji tekstu do podania nazwy użytkownika lub emaila (klasa QLineEdit) + tekst informacyjny (QLabel)
- Linie edycji tekstu do podania nazwy użytkownika lub emaila (klasa QLineEdit) + tekst informacyjny (QLabel) + checkBox do wybrania opcji wyświetlania hasła przy wpisywaniu i informacja o działaniu CheckBoxa (QCheckBox)
- Przycisk do zalogowania się (QPushButton)
- Przycisk do zarejestrowania się (QPushButton)

Stan 2: stan rejestracji konta - dziedziczy jak każdy element okna po QFrame albo QWidget (Wasz wybór), a zawiera:

- Listę danych które należy wypełnić aby się zalogować , dane do wpisania będą do ustalenia (Przygotuję taką automatyczną listę, żeby nie klepać niewidomo ile, wiadomo, że gotowa klasa się nie pomyli)
- Przycisk Zatwierdzenia rejestracji (QPushButton) (Nastąpi próba logowania. Jak się nie uda, pojawi się komunikat odpowiedni w pasku komunikatów i nie zmienia stanu. Jak się uda, przechodzi do **Stan 3**)
- Przycisk Anulowania (QPushButton) (Przechodzi do **Stan 1**)
- Informacje dla użytkownika co się dzieje, czyli że się rejestruje

Stan 3: stan zalogowanego użytkownika - dziedziczy jak każdy element okna po QFrame albo QWidget (Wasz wybór), a zawiera:

- (1).Pionowy pasek zakładek, który pozwala przełączać się miedzy funkcjami programu. Dostępne funkcje zależą od poziomu uprawnień użytkownika (jakie funkcje macie w dokumentacji wstępnej) . Dużo roboty przy tym nie ma. Jest to zwykła lista elementów graficznych (QLabel) wyświetlanych po sobie, a co będzie robić dany element zależeć będzie od poziomu uprawnień. Wyświetlane są funkcje dostępne tylko dla danego poziomu uprawnień. (**Wyświetla tyle kafelków ile jest potrzebnych, Wielkość okna głównego dopasuje się do maksymalnej ilości elementów**) (Opis 1.1.2)
- (2).Panel obsługi aktualnie wybranej funkcji programu. Wyświetla interfejs aktualnie wybranej funkcji. (Opis 1.1.3) (**Wyświetla się tylko w prawej części swojego elementu którego jest dzieckiem**)



1.1.2. Pionowy pasek zakładek

Dostępne funkcje zależą od poziomu uprawnień użytkownika (jakie funkcje macie w dokumentacji wstępnej) . Dużo roboty przy tym nie ma. Jest to zwykła lista elementów graficznych (QLabel) wyświetlanych po sobie, a co będzie robić dany element zależeć będzie od poziomu uprawnień. Wyświetlane są funkcje dostępne tylko dla danego poziomu uprawnień.

Proponowane funkcje dla czytelnika:

- Twoje konto
- Moje książki
- Biblioteka
- Wyloguj

Proponowane funkcje dla bibliotekarza:

- Twoje konto
- Czytelnicy
- Książki
- Wyloguj

Proponowane funkcje dla administratora:

- Twoje konto
- Użytkownicy
- Książki
- Wyloguj

1.1.3. Panel obsługi aktualnie wybranej funkcji

Podobnie jak okno główne składa się ze stanów (klas przechowywanych w unionie).

- Stan „Twoje konto” zawiera:
  - Listę danych (Jak pisałem przygotuję klasę listy danych), te dane jednak będą przełączalne między stanem wyświetlania a stanem edycji (Mam gotową klasę).
  - Przycisk który przełącza dane między stanem wyświetlania i edycji
  - Informacja w jakiej jest zakładce
- Stan „Moje książki” zawiera:
  - (ELEMENT ZOSTANIE WYKORZYSTANY TEŻ W INNYCH MIEJSCACH PROGRAMU + UWAGA 1) Listę książek które aktualnie użytkownik ma wypożyczone. **Ma zawierać również wyszukiwarkę jako niezależny element (nie umieszczać w QScrollArea)**. Elementy listy mają wyświetlać tylko podstawowe informacje. Typu nazwa książki, gatunek i termin oddania. Dodatkowo ma być przycisk wyświetlające opcje dotyczące interakcji z książkami(wystawianie opinii lub przejście do karty książki). Po rozwinięciu elementu można przeczytać dodatkowe informacje. Lista musi więc dostosować się do ilości elementów oraz tego czy są rozwinięte czy nie.
    Przykład można podejrzeć w pasku komunikatów głównego okna.
  - Przycisk przenoszący do przeglądania książek dostępnych w bibliotece.
- Stan „Biblioteka” zawiera:
  - (ELEMENT ZOSTANIE WYKORZYSTANY TEŻ W INNYCH MIEJSCACH PROGRAMU + UWAGA 1)Listę książek które są w bibliotece. **Ma zawierać również wyszukiwarkę jako niezależny element (nie umieszczać w QScrollArea)**. Elementy listy mają wyświetlać tylko podstawowe informacje. Typu nazwa książki, gatunek i aktualny

stan książki(wypożyczona, zarezerwowana, dostępna). Dodatkowo ma być przycisk wyświetlające opcje dotyczące interakcji z książkami(wystawianie opinii lub przejście do karty książki). Po rozwinięciu elementu można przeczytać dodatkowe informacje. Lista musi więc dostosować się do ilości elementów oraz tego czy są rozwinięte czy nie.

Przykład można podejrzeć w pasku komunikatów głównego okna.

- Informacje o aktualnej zakładce

- Stan „Książki" to praktycznie to co stan „Biblioteka". Przy rozwinięciu elementu listy można zmienić stan książki (QComboBox), przycisk do wejścia w kartę książki, wyświetlić opinie (dalsze rozszerzenie, opinie można je usuwać i edytować) oraz przycisk usuwania książki. Oprócz wyszukiwarki należałoby dorobić przycisk do dodawania książki.
- Stan „Czytelnicy" zawiera:
  - (ELEMENT ZOSTANIE WYKORZYSTANY TEŻ W INNYCH MIEJSCACH PROGRAMU + UWAGA 1) Listę czytelników, taka jak w stanach „Książki" czy „Biblioteka". Tylko opcje dotyczące czytelników to zmiana danych, wypożyczenie czy odebranie oddanej książki i usuwanie użytkownika. Przycisk dodania nowego czytelnika jak z książkami.
- Stan „Użytkownicy" tak jak „Czytelnicy" tylko zawiera również „Bibliotekarzy" i innych adminów.

1.2. Klasa do komunikacji z serwerem i interpretacją poleceń

To jeszcze opracuję i się pojawi.

UWAGI

1. Dotyczy uwagi ze stanów i listy książek i użytkowników. Klasa będzie klasą wirtualną, a dopiero ewentualne modyfikacje są to podklasy tej klasy wirtualnej. Przykład macie jak pisałem w pasku komunikatów. Główna klasa to „PromptPanel". Elementy będą wirtualne i też modyfikowane w zależności od potrzeb. Zastanawiam się jeszcze nad szablonami, ale z nimi jest czasem w cholerę problemów. Można spróbować, ale nie mam jeszcze zbyt dużej wiedzy o nich.
2. Uwaga dotycząca pisania kodu. Jeśli wiecie, że wasza klasa by uzyskać dane musiałaby skorzystać z funkcji zewnętrznej (co znaczy pobrać dane z serwera), ale nie ma jeszcze takiej funkcji proszę o wstawienie komentarza, by łatwiej mi lub wam było wrócić do miejsc w kodzie gdzie będzie trzeba to wstawić. Komentarz MUSI mieć postać

// _PH_ Co mam zrobić? Np. // _PH_ Pobrać dane na temat książki

Ważne aby po „//" była spacja oraz po „_PH_" była spacja, bo potem bardzo łatwo idzie wyszukać takie miejsca z pomocą wyszukiwarki fraz.

3. Kolejna uwaga co do pisania kodu. Również bardzo istotna. Proszę, aby stosować się do poniższej zasady:
   - W pliku nagłówkowym umieszczamy #include tylko dla tych klas, których obiekt deklarujemy w klasie lub obiekt jest w argumencie lub „zwrocie" funkcji lub gdy

dziedziczymy tą klasę. Jeśli używamy tylko wskaźników lub referencji dla danej klasy, wtedy stosujemy deklarację wstępną, czyli:

- class MyClass;

, a include dla tej klasy umieszczamy w pliku źródłowym gdzie będzie potrzebna.

- Każdy element, który ma nawet układać geometrię elementów proszę robić w nowej klasie. Uwierzcie mi nie chcecie wszystkiego robić „ręcznie”. Np. Wyszukiwarka to nowy element, więc nowa klasa. Każdy taki element graficzny powinien dziedziczyć po QWidget lub QFrame (co wam wygodniej).

  Jest to dość istotne, by unikać błędów kompliacji (nie błędów w kodzie, kod może być poprawny ale się nie skompiluje, bo gdzieś pliki będą się „gryzły”).

Jeśli potrzebujecie pomocy, to mówcie śmiało. Z chęcią pomogą. Chcę tez mniej wiecej z góry wiedzieć, co chcecie spróbować zrobić, żeby wiedzieć co będę musiał przygotować, by kod się miarę przyjemnie pisało