Opis zastosowanych rozwiązań

Dla oprogramowania do zarzadzania napiwkami w restauracji

Maciej Bajer
Pawet Janduta
Kirill Vereshchako
Yevhen Savchuk
2 June 2025

I. Język programowania

1. C++

Projekt został w całości zrealizowany w języku C++. Język ten zapewnia wysoką wydajność oraz pełną kontrolę nad zasobami systemowymi, co jest istotne w aplikacjach wymagających stabilności i szybkości działania. C++ doskonale współpracuje z biblioteką Qt, umożliwiając tworzenie aplikacji z graficznym interfejsem użytkownika, a także z bazą danych SQLite. Dodatkowym argumentem przemawiającym za wyborem C++ był fakt, że każdy członek zespołu biegle posługuje się tym językiem.

II. Biblioteki zewnętrzne

1. Qt

Wybrano bibliotekę Qt jako główny framework do budowy interfejsu graficznego (GUI) oraz warstwy logiki aplikacji. Współdziała ona z preferowanym przez nas językiem C++ i zapewnia gotowe komponenty GUI oraz intuicyjne w obsłudze narzędzia do tworzenia aplikacji desktopowych. Qt pozwala na łatwą integrację z bazami danych oraz przenoszenie aplikacji między różnymi systemami operacyjnymi. Wybór tej biblioteki był motywowany przede wszystkim prostotą działania oraz możliwością szybkiego tworzenia funkcjonalnych interfejsów użytkownika.

2. SQLite

Jako system zarządzania bazą danych zdecydowano się wykorzystać SQLite, ponieważ jest to lekka, wbudowana baza danych, idealna do niewielkich aplikacji desktopowych, w których nie jest wymagany osobny serwer bazy danych. SQLite umożliwia przechowywanie danych w jednym pliku, tworzenie kopii zapasowych. Integracja z Qt przebiega bezproblemowo, dzięki czemu mogliśmy w prosty sposób połączyć logikę aplikacji z warstwą danych. Dodatkową zaletą jest brak potrzeby instalacji zewnętrznego oprogramowania serwerowego, co zwiększa mobilność i przenośność rozwiązania.

3. Qt Test

Testowanie jednostkowe zostało zaimplementowane z wykorzystaniem wbudowanego frameworka QTest. Każdy test został zaimplementowany jako osobna klasa dziedzicząca po QObject, z metodami testującymi umieszczonymi w private slots. Do weryfikacji różnych danych wejściowych zastosowano mechanizm *_data() w połączeniu z QTest::newRow oraz QFETCH. Same testy wykorzystują makra QCOMPARE, QVERIFY i inne, w celu sprawdzenia poprawności działania.
W celu zapewnienia powtarzalności testów oraz minimalizacji zależności od środowiska zewnętrznego, testy wykorzystujące bazę danych korzystają z osobnej bazy testowej SQLite. Pozwala to odizolować logikę biznesową od czynników zewnętrznych i umożliwia automatyczne wykonywanie testów podczas budowania projektu.
Uruchamianie testów odbywa się za pomocą QTEST_MAIN() lub polecenia ctest, jeśli projekt jest zintegrowany z CMake.

III. Algorytmy i podejście

W ramach realizacji projektu nie korzystano z zaawansowanych algorytmów ani struktur danych, jednak zastosowano odpowiednie podejście do przetwarzania informacji związanych z napiwkami, takie jak grupowanie po dacie, filtrowanie danych z wykorzystaniem zapytań SQL. Do analizy metryk kelnerów wykorzystano funkcje agregujące SQLite, co pozwoliło utrzymać prostotę implementacji przy zachowaniu odpowiedniej wydajności.