МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ФАХОВИЙ КОЛЕДЖ РАКЕТНО-КОСМІЧНОГО МАШИНОБУДУВАННЯ

ДНІПРОВСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ім. О. ГОНЧАРА

Циклова комісія програмної інженерії

**КУРСОВИЙ ПРОЕКТ**

з навчальної дисципліни

«ОБ’ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ»

на тему: «Автоматизоване робоче місце менеджера хімчистки»

(вказати тему курсового проекту)

Студента IVкурсу  ПЗ-20-1  групи

галузь знань 12 «Інформаційні технології»

спеціальності 121 «Інженерія програмного забезпечення»

    Патоки Д.Є

(прізвище та ініціали студента)

Керівник   
викладач Гапоненко Н.В.

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_ Оцінка ECTS: \_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_ Любохинець В.М.

(підпис) (прізвище та ініціали) .

\_\_\_\_\_\_\_\_\_ Ланська С.С. .

(підпис) (прізвище та ініціали) .

\_\_\_\_\_\_\_\_\_ Гапоненко Н.В. .

(підпис) (прізвище та ініціали) .

м. Дніпро - 2023 рік

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ФАХОВИЙ КОЛЕДЖ РАКЕТНО-КОСМІЧНОГО МАШИНОБУДУВАННЯ

ДНІПРОВСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ім. О. ГОНЧАРА

Цикловакомісія програмної інженерії

|  |  |
| --- | --- |
|  | **ЗАТВЕРДЖУЮ**  Голова комісії ПІ  \_\_\_\_\_\_\_\_\_\_\_\_ С.С.Ланська  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2023 р. |

**ЗАВДАННЯ**

**на виконання курсового проекту**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| з дисципліни | | | Об’єктно-орієнтоване програмування | | | | | | | |
| студенту | Патоці Даніілу Євгеновичу | | | | | | | | | |
|  | (прізвище, ім'я та по батькові) | | | | | | | | | |
| Відділення | | Комп'ютерної та програмної інженерії | | | | | | | | |
| Спеціальність | | | | 121 Інженерія програмного забезпечення | | | | | | |
| Курс | IV | | | | | |  | Група (шифр) | | ПЗ-20-1 |
| 1 Тема проекту | | | | | «Автоматизоване робоче місце менеджера хімчистки» | | | | | |
|  | | | | | | | | | | |
|  | | | | | | | | | | |
| 2 Початкові дані | | | | | | Вид послуг(назва, опис), Послуги(назва, вид, зображення, | | | | |
| опис, вартість), Замовлення(номер, клієнт, менеджер, список послуг, дата та | | | | | | | | | | |
| час замовлення, дата виконання, вартість, статус, примітка), Клієнти(ПІБ, | | | | | | | | | | |
| номер телефону, електронна пошта), Менеджер(ПІБ, номер телефону). | | | | | | | | | | |
| Розглянуто і ухвалено на засіданні циклової комісії | | | | | | | | | програмної інженерії | |
| Протокол № 2 від 21.09.2021 р. | | | | | | | | | | |

Керівник КП \_\_\_\_\_\_\_\_\_\_\_\_\_ Н.В.Гапоненко

(підпис) (ініціали та прізвище)

Завдання до виконання

одержав студент \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (ініціали та прізвище)

Дата видачі 21 вересня 2021 р.

Термін виконання 22 листопада 2021 р.

ЗМІСТ

[ВСТУП 4](#_Toc152071408)

[1 ПОСТАНОВКА ЗАДАЧІ 5](#_Toc152071409)

[2 ХАРАКТЕРИСТИКА ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ ПРОЕКТУ 8](#_Toc152071410)

[2.1 Опис середовища програмування 8](#_Toc152071411)

[2.2 Опис мови програмування 10](#_Toc152071412)

[2.3 Опис СУБД MySQL 12](#_Toc152071413)

[2.4 Опис основних принципів ООП 14](#_Toc152071414)

[2.5 Опис подібних програмних продуктів 16](#_Toc152071415)

[3 ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ 21](#_Toc152071416)

[3.1 Опис бази даних 21](#_Toc152071417)

[3.2 Проєктування користувацького інтерфейсу 24](#_Toc152071418)

[4 ІНСТРУКЦІЯ З КОРИСТУВАННЯ ПРОГРАМНОЮ СИСТЕМОЮ 36](#_Toc152071419)

[ВИСНОВКИ 50](#_Toc152071420)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 51](#_Toc152071421)

[Додаток А 58](#_Toc152071422)

ВСТУП

Сучасний світ вимагає від підприємств у галузі послуг нових підходів до управління та оптимізації бізнес-процесів. Однією з найважливіших галузей, яка обслуговує населення і підприємства, є хімчистка. З плином часу вимоги споживачів до цієї послуги зростають, і для успішного функціонування хімчистки необхідно постійно вдосконалювати бізнес-процеси.

Однією з ключових складових оптимізації хімчистки є автоматизація робочих місць менеджерів. У цьому курсовому проекті ми дослідимо, які конкретні вимоги ставляться до робочого місця менеджера хімчистки та які технологічні інструменти можуть сприяти покращенню ефективності їхньої роботи. Ми також розглянемо переваги автоматизації у цій сфері, а також можливі труднощі і виклики, з якими можуть стикнутися підприємства під час впровадження цих технологій.

Цей проект буде корисним для керівників хімчисток, інженерів-програмістів, студентів та всіх, хто цікавиться вдосконаленням бізнес-процесів у сфері послуг. Ми спробуємо розкрити потенціал автоматизації робочих місць менеджера хімчистки і допоможемо зрозуміти, як ця ініціатива може сприяти підвищенню якості послуг і збільшенню конкурентоспроможності хімчисток у сучасному ринковому середовищі.

1 ПОСТАНОВКА ЗАДАЧІ

Розробка «Автоматизоване робоче місце менеджера хімчистки». Програмна система передбачає:

* перелік видів послуг;
* ведення інформації про послуги;
* ведення персональних даних клієнта;
* ведення інформації про менеджерів;
* ведення обліку замовлень;
* ведення аналітики та звітності за наданинми послугами;
* формування електроних чеків.

Детальний опис функціоналу:

1. Організація ведення переліку видів послуг: надання можливості додавання, редагування, видалення та перегляду видів послуг.
2. Організація ведення інформації про послуги: додавання, редагування, видалення та перегляд послуг.
3. Організація ведення обліку клієнтів: надання можливості додавання, редагування, видалення та перегляду інформації про клієнтів.
4. Організація ведення обліку менеджерів: надання можливості додавання, редагування, видалення та перегляду інформації про менеджерів.
5. Організація ведення інформації про замовлення: додавання, редагування, видалення та перегляд.
6. Ведення аналітики та звітності стосовно замовлень: можливість перегляду замовлень за період, розрахунку сумарної вартості та кількості як разом, так і по видам, ведення статистики найзатребованіших послуг.
7. Формування електроних чеків до кожного замовлення.
8. Надання усіх необхідних функцій для зручного перегляду і пошуку інформації: програма повинна надати засоби пошуку та фільтрації інформації всюди, де надано доступ для перегляду інформації.

Початкові дані містять:

1. Вид послуг:
   1. назва;
   2. опис.
2. Послуги:
   1. назва;
   2. вид;
   3. зображення;
   4. опис;
   5. вартість.
3. Замовлення:
   1. номер;
   2. клієнт;
   3. менеджер;
   4. список послуг;
   5. дата та час замовлення;
   6. дата виконання;
   7. вартість;
   8. статус;
   9. примітка.
4. Клієнти:
   1. ПІБ;
   2. номер телефону;
   3. електронна пошта.
5. Менеджер:
   1. ПІБ;
   2. номер телефону.

Мінімальні вимоги до конфігурації ПК для користування цією програмою:

* центральний процесор Pentium IV або відповідний AMD;
* оперативна пам’ять 512 МБ;
* об’єм пам’яті жорсткого диску 256мб для клієнта і 5гб для сервера;

операційні системи Windows 7, 8, 10 або Linux.

2 ХАРАКТЕРИСТИКА ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ ПРОЕКТУ

2.1 Опис середовища програмування

Програма курсового проекту написана в середовищі Qt Creator Community Edition.

Середа розробки Qt (Qt Development Environment) - це потужний і популярний набір інструментів для створення кросплатформенних програм, які включають графічний інтерфейс користувача (GUI), а також інші функціональні можливості. Qt була розроблена компанією Qt Company (раніше Digia, а потім Nokia) та розповсюджується під вільною ліцензією, що дозволяє розробникам вільно використовувати її для створення як комерційних, так і вільних програм.

Кросплатформенність: Однією з основних переваг Qt є здатність створювати програми, які можуть працювати на різних операційних системах, включаючи Windows, macOS, Linux, Android та інші. Це забезпечує широку аудиторію та зменшує витрати часу на адаптацію для різних платформ.

Об'єктно-орієнтований підхід: Qt сприяє об'єктно-орієнтованому програмуванню, що дозволяє розробникам створювати програми, використовуючи об'єкти та класи для реалізації функціональності.

Графічний інтерфейс користувача: Qt надає набір інструментів для створення візуально привабливих та інтерактивних графічних інтерфейсів користувача (GUI). Інтегрований дизайнер Qt спрощує створення вікон, діалогових вікон, кнопок та інших елементів інтерфейсу.

Мова програмування C++: Qt підтримує мову програмування C++, що дозволяє розробникам використовувати потужний та високопродуктивний код для розробки програм.

Багатофункціональність: Qt включає в себе багато бібліотек та інструментів, які допомагають розробникам реалізовувати функціональність, таку як робота з мережами, базами даних, зображеннями, звуками, паралельність та багато іншого.

Легка інтеграція з мовою Python: Qt також надає інтерфейс для інтеграції з мовою програмування Python, що відкриває нові можливості для розробки на Qt.

Підтримка мобільних платформ: Qt дозволяє створювати мобільні додатки для Android та iOS, використовуючи спільний код.

Широка спільнота та документація: Qt має велику та активну спільноту розробників, яка надає підтримку та інформацію. Є також багата документація та навчальні матеріали.

Qt використовується для розробки різноманітних додатків, включаючи настільні програми, мобільні додатки, вбудовані системи, ігри, рішення для автоматизації промислових процесів, а також додатки для встрачі, обробки та аналізу даних.

Велика кількість відомих компаній та проектів використовують Qt в своїх продуктах, включаючи такі бренди, як Autodesk, Adobe, VirtualBox, Skype, Spotify, и Tesla.

Qt дозволяє розробникам створювати програми з високою продуктивністю та швидкодією, що робить її популярним вибором для вимогливих застосунків, таких як відеоігри та графічні редактори.

Велика спільнота розробників Qt допомагає вирішувати питання та пропонувати рішення для різних викликів, з якими зіштовхуються розробники.

Qt Creator: Офіційне інтегроване середовище розробки для Qt. Надає зручний інтерфейс для створення, редагування та відлагодження коду Qt-програм.

Qt Widgets: Набір віджетів, які використовуються для створення класичних настільних додатків з інтерфейсом, який включає в себе вікна, кнопки, текстові поля, таблиці та інші елементи.

Qt Quick: Фреймворк для розробки додатків з модернізованим та анімованим інтерфейсом. Використовує мову програмування QML для опису інтерфейсу.

Qt Multimedia: Бібліотека для роботи з аудіо та відео. Дозволяє відтворювати та записувати мультимедійний контент.

Qt Network: Бібліотека для роботи з мережевими протоколами, включаючи HTTP, FTP, та інші.

Qt SQL: Бібліотека для роботи з реляційними базами даних, яка підтримує SQL-запити.

Qt Concurrent: Бібліотека для роботи з багатозадачністю та паралельністю.

Qt Test: Бібліотека для автоматизованого тестування програм на Qt.

Qt розповсюджується під двома основними ліцензіями: комерційною та LGPL (GNU Lesser General Public License).

Ліцензія LGPL дозволяє використовувати Qt для створення вільних та відкритих програм, але при цьому вимагає публічного вільного розповсюдження змін до бібліотеки Qt, якщо вона була модифікована.

Комерційна ліцензія надає більше свободи та підтримку від Qt Company, і може бути використана для створення пропрієтарних програм.

Qt є однією з найпопулярніших інструментальних середовищ для розробки кросплатформенних програм. Вона надає зручність і продуктивність для розробників, що працюють над проектами різної складності, від настільних додатків до мобільних програм і вбудованих систем. Багата функціональність та гнучкість Qt роблять її потужним інструментом для розробників з різних галузей.

2.2 Опис мови програмування

Програма курсового проекту написана мовою програмування С++. Текст програми знаходиться у додатку Д.

C++ була розроблена в кінці 1970-х та початку 1980-х років Бьярне Страуструпом в якості розширення мови програмування C. Спочатку цю мову називали "C з класами", а згодом вона стала відомою як C++.

Назва "C++" вказує на інкремент оператора C, що символізує приріст функціональності порівняно з C.

Перша стандартна версія мови C++, відома як C++98 (або ISO/IEC 14882:1998), була прийнята в 1998 році. Пізніше було випущено багато інших версій стандарту, таких як C++11, C++14, C++17 та C++20, які вводили нові можливості та покращення.

C++ підтримує основні концепції ООП, такі як класи, об'єкти, спадкування, поліморфізм та інкапсуляція.

Класи дозволяють розробникам створювати власні типи даних з властивостями та методами, які працюють з цими даними.

Спадкування дозволяє створювати нові класи на основі існуючих, успадковуючи їхні властивості та методи.

Поліморфізм дозволяє одному методу мати різні реалізації, що дозволяє створювати загальні інтерфейси для різних типів об'єктів.

Інкапсуляція дозволяє обмежити доступ до деяких даних та методів класу, забезпечуючи конфіденційність та безпеку даних.

Шаблони в C++ дозволяють розробникам створювати загальні програмні конструкції, які можуть працювати з різними типами даних.

Шаблони використовуються для створення загальних структур даних (наприклад, списків або стеків), алгоритмів та інших програмних об'єктів.

Вони дозволяють підвищити перевикористання коду та розв'язати багато типових завдань з меншими зусиллями.

C++ постачається з розширеною стандартною бібліотекою (STL), яка включає в себе багато готових структур даних та алгоритмів.

STL надає реалізації векторів, списків, асоціативних контейнерів (наприклад, масивів і дерев), сортування, пошук та інші корисні інструменти.

Використання STL спрощує і прискорює розробку, оскільки розробники можуть використовувати готові компоненти.

C++ надає можливість створення багатьох паралельних потоків в одній програмі.

Це дозволяє розробникам створювати багатозадачні програми та використовувати багатозадачні бібліотеки для оптимізації продуктивності.

C++ легко інтегрується з кодом, написаним на мові C. Розробники можуть використовувати бібліотеки та функції, написані на C, в своєму коді на C++.

C++ використовується в різних областях, включаючи системне програмування, вбудовані системи, ігри, графіку, фінанси, робототехніку, інтернет-розробку та багато інших.

Вона є популярною в розробці операційних систем (наприклад, Linux), графічних програм, баз даних та великих наукових проектів.

C++ є однією з найпоширеніших мов для розробки вбудованих систем і програмного забезпечення для мікроконтролерів.

C++ може бути складною для вивчення для початківців через велику кількість можливостей та складний синтаксис.

Поганий код на C++ може призвести до пам'яткових витоків та інших проблем з безпекою.

Розробка програм на C++ може вимагати більше часу і зусиль у порівнянні з іншими мовами програмування.

C++ - це мова програмування з вражаючим потенціалом та багатьма можливостями для розробки різноманітних програм та систем. Вона залишається однією з найважливіших мов програмування, особливо в галузях, де потрібна висока продуктивність та доступ до системних ресурсів.

2.3 Опис СУБД MySQL

MySQL — це відкрите відносно реляційна система управління базами даних (СУБД), яка забезпечує ефективне зберігання, управління та доступ до даних. MySQL розроблена і підтримується компанією Oracle Corporation, і є однією з найпопулярніших СУБД в світі, використовується як великими корпораціями, так і невеликими проектами та веб-сайтами.

Реляційна СУБД: MySQL базується на реляційній моделі даних, що дозволяє зберігати дані у вигляді таблиць з рядками та стовпцями. Ця модель дозволяє легко створювати структуровані та добре організовані бази даних.

SQL (Structured Query Language): MySQL підтримує мову SQL, яка використовується для створення, зміни, видалення та запитів до даних. SQL є стандартною мовою для взаємодії з реляційними базами даних.

Швидкість та продуктивність: MySQL славиться високою продуктивністю та швидкістю доступу до даних. Вона володіє оптимізованим ядром та покращеними алгоритмами обробки запитів.

Підтримка великих обсягів даних: MySQL може впоратися з великими обсягами даних та високою навантаженістю завдяки покращеній оптимізації та кластеризації.

Кросплатформенність: MySQL підтримується на різних операційних системах, включаючи Windows, Linux, macOS та інші, що дозволяє розробникам використовувати її на різних платформах.

Спільнота та підтримка: MySQL має велику спільноту розробників, що надає безліч документації, форумів і онлайн-ресурсів для підтримки та навчання.

Відкрита ліцензія: MySQL розповсюджується під ліцензією GPL (GNU General Public License), що дозволяє використовувати її безкоштовно та вільно.

Розширюваність: MySQL підтримує розширення та плагіни, які дозволяють розробникам додавати нову функціональність та підтримку різних типів даних.

Безпека: MySQL має вбудовану систему автентифікації та авторизації, а також підтримку шифрування даних для забезпечення безпеки даних.

MySQL використовується в різних галузях, включаючи веб-розробку, розробку програмного забезпечення, обробку даних, аналітику та багато інших.

Великі веб-сайти, такі як Facebook, Twitter, YouTube та інші, використовують MySQL для зберігання та управління своїми даними.

MySQL також популярна серед розробників вбудованих систем та пристроїв, де вимоги до невеликого розміру та швидкості важливі.

Вона використовується в системах управління контентом, електронних комерційних платформах, банківських системах та інших галузях.

MySQL Workbench: Офіційна інтегрована середа розробки для MySQL, яка надає зручний інтерфейс для створення та управління базами даних.

Командний рядок MySQL: Розробники також можуть взаємодіяти з MySQL за допомогою командного рядка, використовуючи SQL-запити та команди.

MySQL є потужною та популярною системою управління базами даних, яка надає ефективний спосіб зберігання та обробки даних. Вона застосовується в різних проектах та галузях завдяки своїй продуктивності, швидкості та відкритій ліцензії. MySQL є важливим інструментом для розробників, які працюють з базами даних у своїх програмах та проектах.

2.4 Опис основних принципів ООП

Об'єктно-орієнтоване програмування або ООП (object — oriented programming) — методологія програмування, заснована на представленні програми у вигляді сукупності об'єктів кожен з яких являється реалізацією визначеного типу, що використовує механізм пересилки повідомлень і класи організовані в ієрархію наслідування.

Центральний елемент ООП — абстракція. Дані за допомогою абстракції перетворяться в об'єкти, а послідовність обробки цих даних перетворюється на набір повідомлень, що передаються між цими об'єктами. Кожен з об'єктів має свою власну унікальну поведінку. До об'єктів можна звертатися як до окремих сутностей, які реагують на повідомлення, що наказують їм виконати якісь дії.

ООП характеризується наступними принципами (по Алану Кею) :

1. Усе являється об'єктом;
2. Обчислення здійснюються шляхом взаємодії (обміну даними) між об'єктами, при якому один об'єкт вимагає, щоб інший об'єкт виконав деяку дію; об'єкти взаємодіють, посилаючи і отримуючи повідомлення; повідомлення - це запит на виконання дії, доповнений набором аргументів, які можуть знадобитися при виконанні дії;
3. Кожен об'єкт має незалежну пам'ять яка складається з інших об'єктів
4. Кожен об'єкт є представником класуякий виражає загальні властивості об'єктів даного типу
5. У класі задається функціональність (поведінка об'єкту); тим самим усе об'єкти, які являються екземплярами одного класу, можуть виконувати одні і ті ж дії;
6. Класи організовані в єдину деревовидну структуру із загальним коренем, звану ієрархією спадкоємства; пам'ять і поведінка, пов'язана з екземплярами певного класу, автоматично доступні будь-якому класу, розташованому нижче в ієрархічному дереві.

Абстрагування (abstraction) — метод рішення задачі, при якому об'єкти різного роду об'єднуються загальним поняттям (концепцією), а потім згруповані суті розглядаються як елементи єдиної категорії. Абстрагування дозволяє відокремити логічний сенс фрагмента програми від проблеми його реалізації, розділивши зовнішній опис (інтерфейс) об'єкту і його внутрішню організацію (реалізацію).

Інкапсуляція (encapsulation) — техніка, при якій несуттєва з точки зору інтерфейсу об'єкту інформація ховається усередині нього.

Спадкоємство (inheritance) — властивість об'єктів, за допомогою якої екземпляри класу дістають доступ до даних і методів класів-предків без їх повторного визначення. Спадкоємство дозволяє різним типам даних спільно використовувати один і той же код, призводячи до зменшення його розміру і підвищення функціональності.

Поліморфізм (polymorphism) —- властивість, що дозволяє використовувати один і той же інтерфейс для різних дій; поліморфній змінній, наприклад, може відповідати декілька різних методів.

Поліморфізм перекроює загальний код, що реалізовує деякий інтерфейс, так, щоб задовольнити конкретним особливостям окремих типів даних. Клас (class) - безліч об'єктів, пов'язаних спільністю структури і поведінка; абстрактний опис даних і поведінки (методів) для сукупності схожих об'єктів, представники якої називаються екземплярами класу.

Об'єкт (object) — конкретна реалізація класу, що володіє характеристиками стану, поведінки і індивідуальності, синонім екземпляра.

2.5 Опис подібних програмних продуктів

Odoo

Odoo (раніше відома як OpenERP) - це інтегрована система управління підприємством (ERP), яка надає різноманітні бізнес-функції та додаткові модулі для автоматизації різних аспектів підприємництва. Ось огляд ключових аспектів системи управління Odoo:

1. Модульна структура: Odoo побудована на основі модульної архітектури, що дозволяє підприємствам вибирати індивідуальні функціональність, які вони потребують, та налаштовувати систему під свої потреби. Модулі охоплюють такі сфери, як облік, продажі, закупівлі, кадри, склад, фінанси, виробництво, CRM, та багато інших.

2. Управління клієнтами (CRM): Odoo надає модуль управління відносинами з клієнтами, який допомагає вести базу клієнтів, здійснювати маркетинг та продажі, а також слідкувати за поточними проектами та взаємодією з клієнтами.

3. Управління складом: Odoo включає модуль для управління запасами, дозволяючи вести облік товарів на складі, здійснювати поставки та слідкувати за рівнем запасів.

4. Фінансовий облік: Система має модуль для обліку фінансових операцій, створення фінансових звітів, обліку податків та платежів.

5. Кадровий облік та управління персоналом: Odoo дозволяє вести облік співробітників, створювати графіки роботи, відслідковувати відпустки, оплату праці та інші кадрові аспекти.

6. Продажі та закупівлі: В системі є модулі для управління продажами та закупівлями, створення замовлень, слідкування за постачальниками та замовниками

7. Виробництво: Odoo надає можливість управляти виробництвом, включаючи планування виробництва, керування робочими процесами та виробництвом продукції.

8. Аналітика та звіти: В системі доступні засоби створення звітів та аналітики, які допомагають аналізувати бізнес-процеси та приймати ефективні рішення.

9. Мобільні додатки: Odoo пропонує мобільні додатки для зручного доступу до інформації та функцій системи на смартфонах та планшетах.

10. Спільнота та розширення: Odoo має активну спільноту розробників, яка постійно розширює можливості системи шляхом створення нових модулів та розширень.

11. Відкритий вихідний код: Odoo доступний під відкритою ліцензією, що дозволяє підприємствам власну розробку та налаштування системи

Odoo дозволяє підприємствам автоматизувати багато аспектів своєї діяльності, підвищуючи продуктивність та ефективність бізнес-процесів. Безперечно, система стала однією з популярних платформ ERP завдяки своїй модульній структурі та гнучкості налаштувань.

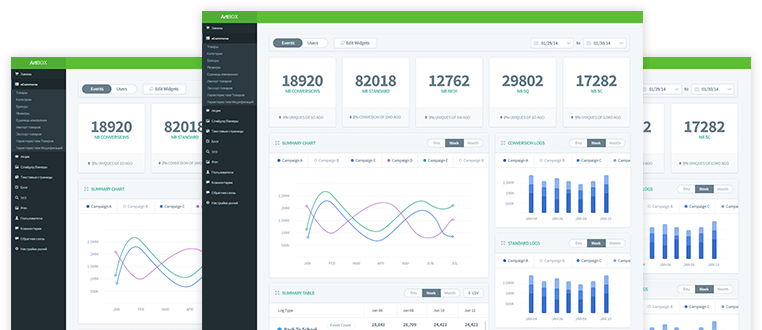


Рисунок 2.1 – Зображення з сайту розробника

QuickBooks

QuickBooks - це популярна програма для обліку та фінансового управління, розроблена компанією Intuit. Вона призначена для ведення фінансового обліку малого та середнього бізнесу та надає різноманітні функції для ефективного управління фінансами та бухгалтерією. Ось ключові аспекти системи учета QuickBooks:

1. Облік фінансів: QuickBooks дозволяє підприємствам вести облік доходів і витрат. Ви можете створювати рахунки, виписувати чеки, слідкувати за банківськими операціями та вести облік фінансів підприємства.

2. Управління рахунками і платежами: QuickBooks надає можливість створювати та відстежувати рахунки до клієнтів, керувати заборгованістю та слідкувати за платежами від клієнтів.

3. Обробка податків: QuickBooks допомагає підприємствам розраховувати, відстежувати та сплачувати податки. Вона надає інструменти для підготовки податкових декларацій та відстежування податкових зобов'язань.

4. Управління запасами: QuickBooks дозволяє вести облік товарів на складі, відстежувати рівень запасів та керувати поставками.

5. Заробітна плата та кадри: Система включає функції для розрахунку заробітної плати співробітників, ведення обліку годин праці, а також підтримує дані про співробітників.

6. Звітність і аналітика: QuickBooks надає можливість створювати фінансові звіти та аналізувати фінансовий стан підприємства. Ви можете генерувати різні звіти, такі як баланс, звіт про прибуток і збиток, звіт про готовність до оплати та інші.

7. Підтримка для онлайн-платежів: QuickBooks дозволяє приймати онлайн-платежі від клієнтів та використовувати різні платіжні системи.

8. Мобільні додатки: QuickBooks пропонує мобільні додатки для зручного доступу до фінансової інформації зі смартфонів та планшетів.

9. Відкритий доступ для бухгалтерів: Бухгалтери можуть легко працювати з даними своїх клієнтів, що використовують QuickBooks, завдяки спеціальному доступу.

QuickBooks володіє інтуїтивним інтерфейсом та широким набором функцій, що робить її популярною серед малих та середніх підприємств. Вона допомагає ефективно вести облік, спрощує податкову звітність та допомагає зробити фінансовий управління більш доступним та ефективним.

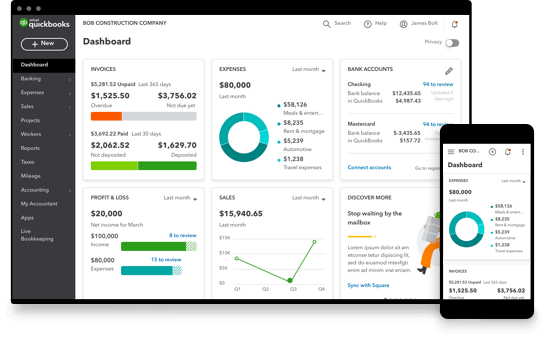


Рисунок 2.2 – Зображення з сайту розробника

Проте, незважаючи на усі переваги наведених систем, вважаю розробку проєкту цілком доцільною, оскільки наявні рішення зосередженні на покриття усіх потреб компаній в різних сферах, що в певних випадках може стати складним завданням, а розробка спеціалізованого додатку значно спростить його інтеграцію у роботу компанії.

3 ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ

3.1 Опис бази даних

Зважаючи на необхідність зберігати великі об’єми інформації, під час проектування доцільно розділити базу даних на декілька сутностей (таблиць).

База даних складається з 8-и таблиць: clients – клієнти, managers – менеджери, types\_of\_services – види послуг, services – послуги, files – файли, orders – замовлення, orders\_services – проміжна таблиця списку послуг у замовленні, statuses – можливі статуси замовлень. ER-діаграма бази даних представлена на рисунку 3.1, перелік таблиць, їх поля та призначення полів наведено у таблицях 3.1 — 3.7.

Таблиця 3.1 – Таблиця clients

|  |  |
| --- | --- |
| Поле | Призначення |
| id | Ідентифікатор |
| pib | ПІБ |
| phone\_number | Номер телефону |
| email | Електронна пошта |

Таблиця 3.2 – Таблиця managers

|  |  |
| --- | --- |
| Поле | Призначення |
| id | Ідентифікатор |
| pib | ПІБ |
| phone\_number | Номер телефону |

Таблиця 3.3 – Таблиця types\_of\_services

|  |  |
| --- | --- |
| Поле | Призначення |
| id | Ідентифікатор |
| title | Назва |
| description | Опис |

Таблиця 3.4 – Таблиця services

|  |  |
| --- | --- |
| Поле | Призначення |
| id | Ідентифікатор |
| title | Назва |
| idType | Ідентифікатор виду послуги |
| idImage | Ідентифікатор зображення |
| description | Опис |
| cost | Вартість |

Таблиця 3.5 – Таблиця files

|  |  |
| --- | --- |
| Поле | Призначення |
| id | Ідентифікатор |
| file | Файл |

Таблиця 3.6 – Таблиця orders

|  |  |
| --- | --- |
| Поле | Призначення |
| number | Номер замовлення |
| idClient | Ідентифікатор клієнта |
| idManager | Ідентифікатор менеджера |
| dateTime\_order | Дата та час замовлення |
| date\_of\_completion | Дата виконання замовлення |
| total\_cost | Сума замовлення |
| idStatus | Ідентифікатор статусу замовлення |
| comment | Коментар до замовлення |

Таблиця 3.7 – Таблиця orders\_services

|  |  |
| --- | --- |
| Поле | Призначення |
| idOrder | Ідентифікатор замовлення |
| idService | Ідентифікатор послуги |
| number | Кількість |

Таблиця 3.8 – Таблиця statuses

|  |  |
| --- | --- |
| Поле | Призначення |
| id | Ідентифікатор |
| title | Назва |

Изображение выглядит как текст, диаграмма, Параллельный, число

Автоматически созданное описание

Рисунок 3.1 – ER-діаграма бази даних

3.2 Проєктування користувацького інтерфейсу

Задачею графічного інтерфейсу програми є забезпечення комфортної взаємодії користувача з програмним додатком. Графічний інтерфейс складається з форм та елементів, які розташовуються на формах. Розробка інтерфейсу програми починається зі створення нової, стартової форми, шляхом внесення змін до властивостей автоматично згенерованої форми (centralWidget), а також шляхом додавання на форму нових необхідних елементів. На форму можуть бути додані різноманітні компоненти для взаємодії з текстом, числовими даними, базами даних тощо. Для перегляду і зміни значень властивостей форми і її компонентів використовується вікно Object Inspector, у якому знаходиться більшість даних про об’єкт, який є активним на даний момент (ім’я об’єкту, шрифт, ширина, висота тощо). Під час проектування дуже важливо цілеспрямовано обрати компоненти.

Інтерфейс програми складається з основної форми, яка має у собі усі необхідні елементи для взаємодії користувача із інформацією, та сторінок, які в залежності від дій користувача змінюють одна одну і мають необхідні компоненти для виконання бажаної дії користувачем.

Інтерфейс програми налічує декілька вікон:

* MainWindows — головне вікно програми, яке в залежності від дій користувача показує відповідні компоненти для виконання бажаної дії. Налічує у собі контролюючі компоненти StackedWidget для відтворення механізму зміни сторінок. Усі сторінки створені у схожими за розташуванням елементів керування, для зручного користування программою;
* AddEditClientsDialog — вікно, що реалізовує інтерфейс програми для додавання або редагування інформації про клієнтів;
* AddEditManagersDialog — вікно, що реалізовує інтерфейс програми для додавання або редагування інформації про менеджерів;
* AddEditTypesOfServicesDialog — вікно, що реалізовує інтерфейс програми для додавання або редагування інформації про типи послуг;
* AddEditServicesDialog — вікно, що реалізовує інтерфейс програми для додавання або редагування інформації про послуги;
* AddEditOrdersDialog — вікно, що реалізовує інтерфейс програми для додавання або редагування інформації про замовлення;
* AddEditStatusesDialog — вікно, що реалізовує інтерфейс програми для додавання або редагування інформації про статуси замовлень;
* SelectFilesDialog — вікно, що реалізовує інтерфейс програми для вибору зображення для послуги;
* SelectServicesDialog — вікно, що реалізовує інтерфейс програми для додавання послуг до замовлення при його створенні.

Головне вікно програми розділене, умовно на дві частини, ліворуч знаходиться панель меню програми і замає найменшу частину простору вікна, а праворуч відображаеться робоча область, на якій в залежності від обраного пункту меню відображається відповідна сторінка.

На рисунку 3.2 зображено головне вікно програми на сторінці привітання, де на робочій області відображено напис вітання та запрошення почати роботу із вибора необхідного пункту меню.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 3.2 – Стартова сторінка програми

На рисунку 3.3 зображено стоірнку перегляду клієнтів, на якій зверху показано напис для цієї сторінки, далі йде панель пошуку із полями введення ключових даних для пошуку, таблиця для відображення даних клієнтів та кнопки керування.

Изображение выглядит как снимок экрана, текст, программное обеспечение, мультимедиа

Автоматически созданное описание

Рисунок 3.3 – Сторінка перегляду клієнтів

На рисунку 3.4 зображено сторінку перегляду даних менеджерів, на якій теж зоюражено заголовок вікна, панель пошуку, табличне представлення для відображення даних про менеджерів та кнопки керування.

Изображение выглядит как текст, снимок экрана, программное обеспечение, мультимедиа

Автоматически созданное описание

Рисунок 3.4 – Сторінка перегляду менеджерів

На рисунку 3.5 зображено сторінку перегляду типів послуг, що розроблена відповідно, до поперелніх сторінок, проте замість табличного представлення має спискове.

Изображение выглядит как текст, снимок экрана, программное обеспечение, мультимедиа

Автоматически созданное описание

Рисунок 3.5 – Сторінка перегляду типів послуг

На рисунку 3.6 зображено сторінку перегляду послуг, вона так само має титульний напис, панель пошуку, табличне представлення та кнопки керування.

Изображение выглядит как снимок экрана, текст, программное обеспечение, мультимедиа

Автоматически созданное описание

Рисунок 3.6 – Сторінка перегляду послуг

На рисунку 3.7 зображено сторінку ведення замовлень, розроблена в тому ж стилі, що і попередні, має титульний напис, панель пошуку, табличне представлення та кнопки керування.

Изображение выглядит как текст, снимок экрана, программное обеспечение, мультимедиа

Автоматически созданное описание

Рисунок 3.7 – Сторінка перегляду замовлень

На рисунку 3.8 зображено сторінку ведення обліку статусів замовлень, вона має титульний напис, панель пошуку, спискове представлення та кнопки керування.

Изображение выглядит как текст, снимок экрана, программное обеспечение, мультимедиа

Автоматически созданное описание

Рисунок 3.8 – Сторінка перегляду статусів замовлень

На рисунку 3.9 зображено вікно додавання, або едагування клієнта. Має титульний напис, що змінюється в залежності від виконуваної функції, поля для введення даних про клієнта та кнопки керування.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 3.9 – Вікно додавання та редагування клієнтів

На рисунку 3.10 зображено вікно додавання або редагування менеджерів. Воно так самое має титульний напис в залежності від функції для якої вікно було відкрите, поля для введення інформації та кнопки керування.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Рисунок 3.10 – Сторінка додавання та редагування менеджерів

На рисунку 3.11 зображено вікно для додавання або редагування іфномарції про замовлення. Має титульний запис, що змінюється в залежності від виконуваної дії, таблицю в якій відображаються обрані в замовленні послуги та поля для введення інших даних про замовлення. У вікні також присутні кнопки керування.

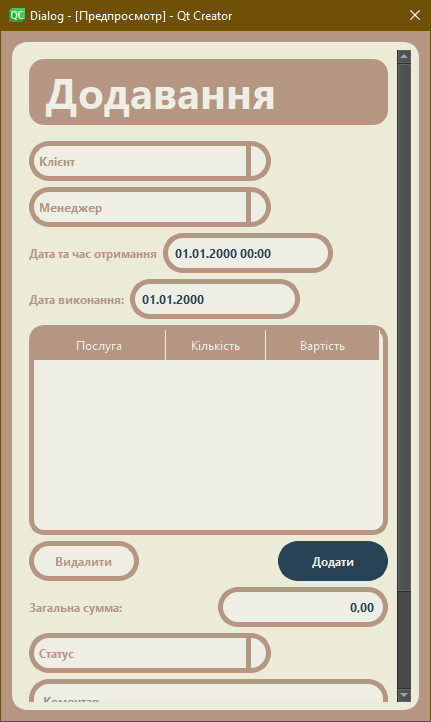


Рисунок 3.11 – Сторінка додавання та редагування замовлень

На рисунку 3.12 зображено вікно додавання або редагування послуги, яке міє титульний напис, що змінюється в залежності від виконуваної дії. Вікно мість такод панель із зображенням та кнопку для його завантаження, крім уього на формі присутні компоненти для введення інших даних про послугу а також кнопки керування.

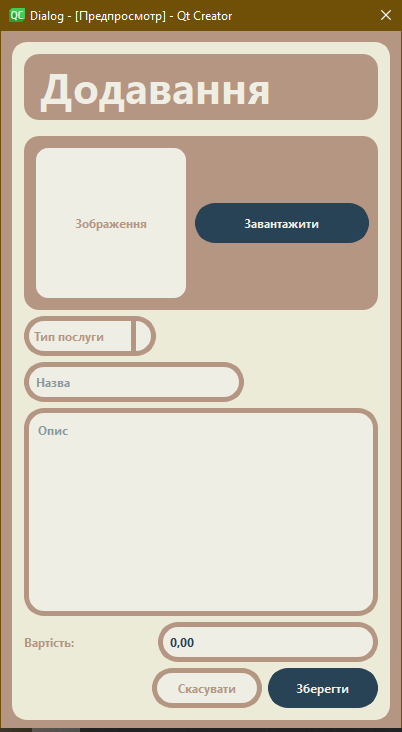


Рисунок 3.12 – Сторінка додавання та редагування послуг

На рисунку 3.13 зображено вікно додавання або редагування статусу замовлення. Воно має титульний напис, що змінюється в залежності від виконуваної діє, поле для введення даних статусу та кнопки керування.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 3.13 – Сторінка додавання та редагування статусів замовлень

На рисунку 3.14 зображено вікно додавання або редагування типу послуги. Воно має титульний напис, поля для ведення даних типу послуги та кнопки керування.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 3.14 – Вікно додавання та редагування типів послуг

На рисунку 3.15 зображено вікно вибору та завантаження зображень. Воно має титульний напис, списочне представлень зображень які вже є у базі даних, та кнопку для завантаження нового зображення, присутні також і кнопки керування.

Изображение выглядит как текст, снимок экрана, мультимедиа

Автоматически созданное описание

Рисунок 3.15 – Вікно вибору та завантаження зображень

На рисунку 3.16 зображено вікно додавання послуг до замовлення, воно має титульний напис, таблицю відображення наявних послуг та поле введення кількості послуг, присутні і кнопки керування.

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 3.16 – Вікно додавання послуги до замовлення

4 ІНСТРУКЦІЯ З КОРИСТУВАННЯ ПРОГРАМНОЮ СИСТЕМОЮ

Робота з програмою починається у головному вікні, в якому відображено стартову сторінку із запитом, щодо вибору необхідної дії у меню ліворуч. Оскільки программа призначена для одного користувача, то авторизація не потребується. У меню ліворуч відображені усі доступні пункти меню, а при виборі конкретного, він буде підсвічуватись, демонструючи на якій сторінці знаходиться користувач. При відкритті программи та відображенні стартової сторінки, жоден з пунктів меню не підсвічується. Стартову сторінку зображено на рисунку 4.1.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Рисунок 4.1 – Стартова сторінка головного вікна

На сторінці клієнтів, зображено таблицю з поточними даними про клієнтів, над нею розташоване поле пошуку, а під табличкою кнопки для взаємодії з даними. Сторінку перегляду даних клієнтів зображено на рисунку 4.2.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 4.2 – Сторінка перегляду даних клієнтів

Для пошуку записів у таблиці, необхідно заповнити одне або декілька полей пошуку та натиснути відповідну кнопку. Роботу пошуку зображено на рисунку 4.3.

Изображение выглядит как текст, снимок экрана, программное обеспечение, мультимедиа

Автоматически созданное описание

Рисунок 4.2 – Сторінка перегляду даних клієнтів

Для редагування запису, необхідно обрати потрібний запис із таблиці та натиснути кнопку «Редагувати», після чого відкриється вікно редагування даних клієнта. Демонстрацію редагування зображено на рисунку 4.3.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 4.3 – Вікно редагування даних клієнта

Для збереження змін необхідно натиснути кнопку «Зберегти», для скасування «Скасувати».

Для видалення запису про користувача слід обрати його із загальної таблички користувачів та натиснути кнопку «Видалити».

За додавання нових записів клієнтів відповідає однойменна кнопка на сторінці, при натисканні на яку вікриється вікно додавання клієнтів, робота з ним аналогічна роботі із вікном редагування, тому демонстрації непотребує.

Наступною йде сторінка роботи з менеджерами, яка зроблена відовідно до сторінки з клієнтами, де в центрі відображено усі записи про менеджерів компанії, зверху поля для пошуку інфформації, а з низу кнопки керування. Сторінку зображено на рисунку 4.4.

Изображение выглядит как текст, снимок экрана, программное обеспечение, мультимедиа

Автоматически созданное описание

Рисунок 4.4 – Сторінка роботи з менеджерами

Для пошуку інформації необхідно заповнити необхідні поля та натиснути кнопку пошуку. Для скидання параметрів пошуку та відображення усіх записів необхідно звільнити усі поля та повторно натиснути кнопку пошуку.

Для редагування інформації необхідно обрати запис для редагування у таблиці та натиснути кнопку «Редагувати», після чого відкриється вікно редагування запису про менеджера. Відовідне вікно зображено на рисунку 4.5.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Рисунок 4.5 – Вікно редагування запису менеджера

Для видалення так само необхідно обрати запис для видалення та натиснути відповідну кнопку.

Кнопка «Додати» відкриє вікно додавання запису. Воно цілком ідентичне вікну редагування запису тому демонстрації не потребує.

Наступною йде сторінка роботи з типами послуг. Як і попередні, вона розроблена за схожою схемою розташування компонентів, проте дані про типи відображені у вигляді списку їх назв. Зверху надано можливість пошуку за назвою типу послуги а зних узнаходяться кнопки керування. Сторінку зображено на рисунку 4.6.

Изображение выглядит как снимок экрана, текст, программное обеспечение, мультимедиа

Автоматически созданное описание

Рисунок 4.6 – Вікно редагування запису менеджера

Для додавання запису необхдіно натиснути відповідну кнопку під списком записів.

Для редагування необхідно сопчатку обрати запи для редагування та натиснути кнопку «Редагувати», після чого відкриється вікно редагування. Вікно редагування зображено на рисунку 4.7.

Для видалення необхідно обрати запис для видалення та натиснути кнопку «Видалити».

Пошук здійнюється за допомогою запомненням відповідного поля та натиском на кнопку «Пошук». Для відміни пошуку необхідно очистити поле пошуку та знову натиснути кнопку «Пошук».

У вікні редагування запису для збереження змін необхідно натиснути кнопку «Зберегти», а для скасування редагування натиснути кнопку «Скасувати».

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 4.6 – Вікно редагування типу послуги

Наступною йде сторінка обліку послуг. Розміщення компонентів зроблено відповідно, до попередніх сторінок, де зверху йде поле пошуку, наступним таблиця зі списком послуг, а нижче кнопки керування.

Для пошуку необхідно заповнити відповідні поля на натиснути кнопку «Пошук». Для відміни фільтра треба зкинути значення усіх полів пошуку та снову натиснути кнопку «Пошук». Сторінку перегляду послуг зображено на рисунку 4.7.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 4.7 – Сторінка перегляду послуг

Для редагування запису необхідно обрати запис для редагування та натиснути кнопку «Редагування». Вікно редагування послуги зображено на рисунку 4.8.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

Рисунок 4.8 – Вікно редагування послуги

У вікні додавання або редагування послуг є можливість додати зображення, до послуги, для цього необхідно натиснути кнопку «Завантажити» після чого відкриється вікно вибору зображення. Вікно вибору зображень зображено на рисунку 4.9.

Изображение выглядит как текст, снимок экрана, мультфильм

Автоматически созданное описание

Рисунок 4.9 – Вікно вибору зображень

У цьому вікні буде показано список усіх зображень у базі даних, які можна додати до послуги. Для цього необхідно натиснути на цікавляче зображення та натиснути кнопку «Обрати». Якщо потрібного зображення немає у списку, його можно завантажити з комп’ютера користувача, для цього необхідно натиснути кнопку «Завантажити» та обрати у файлах комп’ютера бажане зображення. Після цього воно з’явиться у списку доступних для вибору і його можна буде додати, до послуги.

Для збереження змін після додавання або редагування замовлення необхідно натиснути кнопку «Зберегти», для скасування змін кнопку «Скасувати».

Для видалення послуги необхідно обрати послугу для видалення та натиснути кнопку «Видалити».

Наступна сторінка ведення обліку замовлень. Створена за принципами попередніх сторінок, має поле пошуку, таблицю замовлень та кнопки керування. Сторінку зображено на рисунку 4.10.

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Рисунок 4.10 – Сторінка перегляду замовлень

Для додавання запису необхідно натиснути відповідну кнопку, після чого відкриється вікно додавання замовлення. Вікно додавання замовлення зображено на рисунку 4.11.

Изображение выглядит как текст, снимок экрана, мультимедиа, программное обеспечение

Автоматически созданное описание

Рисунок 4.11 – Вікно додавання замовлення

Для додавання послуги у замовлення необхідно натиснути кнопку «Додати» під таблицею замовлень, після чого відкриється вікно вибору послуги. Вікно зображено на рисунку 4.12.

У вікні вибору послуги необхідно обрати послугу зі списку та ввести кількість цих послуг, після чого натиснути кнопку «Додати». Для скасування вибору послуги слід натиснути кнопку «Скасувати».

Якщо раптово до замовлення була додана непотрібна послуга, її можна видалити обравши необхідну зі списку і натиснувши кнопку видалити. В ході додавання та видалення послуг в замовленні автоматично підраховується загальна вартість замовлення.

Для збереження замовлення необхідно натиснути кнопку «Зберегти», для скасування «Скасувати».

Изображение выглядит как текст, снимок экрана, мультимедиа, программное обеспечение

Автоматически созданное описание

Рисунок 4.12 – Вікно додавання послуги до замовлення

Вікно редагування замовлення відповідне вікну додавання, проте редагувати можливо лише статус замовлення.

Для видалення замовлення слід обрати замовлення для видалення та натиснути кнопку «Видалити».

Наступна сторінка це облік можливих статусів замовлень. Має поле пошуку статусів, список записів та кнопки керування. Сторінку зображено на рисунку 4.13.

Изображение выглядит как текст, снимок экрана, программное обеспечение, мультимедиа

Автоматически созданное описание

Рисунок 4.13 – Сторінка обліку статусів замовлень

Для редагування запису необхідно обрати відповідний запи зі списку та натиснути кнопку «Редагувати», після чого відкриється вікно редагування запису. Вікно редагування зображено на рисунку 4.14.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 4.14 – Вікно редагування статусів

Для збереження змін необхідно натиснути кнопку «Зберегти», щоб скасувати зміни кнопку «Скасувати».

Для додавання статусів слід натиснути кнопку «Додати». Вікно додавання записів теж саме що і для редагування, тому демонстрація непотрібна.

Для видалення статусу слід обрати запис для видалення та натиснути кнопку «Видалити».

Для завершення роботи програми слід натиснути кнопку «Вихід».

ВИСНОВКИ

У результаті виконання курсового проекту, було розглянуто та розроблено автоматизоване робоче місце для менеджера хімчистки, яке включає в себе широкий спектр функцій та інструментів для ефективного управління бізнес-процесами. Дане рішення спрямоване на покращення оперативності, точності та зручності управління хімчисткою, забезпечуючи підтримку ключових аспектів її діяльності.

Розроблена система дозволяє зручно вести обіг замовлень у хімчисці, має можливість вести базу клієнтів та співробітників, змінювати наявні послуги в залежності від специфіки роботи закладу та безпосередньо відслідковувати виконання отриманих замовлень.

Впровадження такої системи в роботу закладу хімчистки значно спростило би та скоротило затрати на оформлення та відстеження замовлень, а за допомогою вбудованої звітності надати актульну оцінку фінансової складової закладу.

В процесі розробки програмного забезпечення було закріплено навички роботи з інструментрієм QtFramework, MySQL та візуального оформлення багатовіконних додатків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Відео та лекції по виченню Qt Creatorehttps://resources.qt.io/resources-by-content-type-videos-demos-tutorials.

2 Електронний научно-практический журнал «Современная техника и технологии» [Електронний ресурс] // technology.snauka.ru. Режим доступу: http://technology.snauka.ru/2016/12/11465

3 М. Саммерфілд «Qt. Профессиональное программирование. Разработка кроссплатформенных приложений на С++», Символ-Плюс, 2011

4 Ж. Бланшет, М. Саммерфілд «QT 4: программирование GUI на С++», КУДИЦ-Пресс, 2008

5 М. Шлеє «Qt 5.3. Профессиональное программирование на C++», БХВ-Петербург, 2015

6 Офіційна документація — doc.qt.io

Додаток А

Лістинг програми

Лістинг А.1 – Клас роботи з базою даних

#ifndef DATABASEMODULE\_H

#define DATABASEMODULE\_H

#include <QObject>

#include <QSqlDatabase>

#include <QSqlTableModel>

#include <QSqlError>

#include <QSqlQuery>

#include "clientsmodel.h"

#include "managersmodel.h"

#include "typeofservicesmodel.h"

#include "servicesmodel.h"

#include "filesmodel.h"

#include "ordersmodel.h"

#include "ordersservicesmodel.h"

#include "statusesmodel.h"

class DatabaseModule : public QObject

{

Q\_OBJECT

QSqlDatabase db;

public:

DatabaseModule(QObject \*parent = nullptr);

bool connect();

void disconnect() { db.close(); }

QSqlError lastError() { return db.lastError(); }

void setDbName(QString name) { db.setDatabaseName(name); }

void setHostAddress(QString address) { db.setHostName(address); }

void setHostPort(int port) { db.setPort(port); }

void selectTables();

ClientsModel\* clientsModel;

ManagersModel\* managersModel;

TypeOfServicesModel\* typeOfServicesModel;

ServicesModel\* servicesModel;

FilesModel\* filesModel;

OrdersModel\* ordersModel;

OrdersServicesModel\* ordersServicesModel;

StatusesModel\* statusesModel;

};

#endif // DATABASEMODULE\_H

#include "databasemodule.h"

#include <QDebug>

DatabaseModule::DatabaseModule(QObject \*parent)

{

db = QSqlDatabase::addDatabase("QMYSQL");

db.setHostName("localhost");

db.setPort(3306);

db.setDatabaseName("dry\_cleaning");

db.setUserName("dry\_cleaning\_user");

db.setPassword("dry\_cleaning\_user");

}

bool DatabaseModule::connect()

{

if(db.open())

{

clientsModel = new ClientsModel(this);

clientsModel->setTable("clients", &db);

managersModel = new ManagersModel(this);

managersModel->setTable("managers", &db);

typeOfServicesModel = new TypeOfServicesModel(this);

typeOfServicesModel->setTable("types\_of\_services", &db);

filesModel = new FilesModel(this);

filesModel->setTable("files", &db);

servicesModel = new ServicesModel(this);

servicesModel->setTable("services", &db);

servicesModel->typeModel = typeOfServicesModel;

servicesModel->fileModel = filesModel;

statusesModel = new StatusesModel(this);

statusesModel->setTable("statuses", &db);

ordersModel = new OrdersModel(this);

ordersModel->setTable("orders", &db);

ordersModel->clients = clientsModel;

ordersModel->managers = managersModel;

ordersModel->statuses = statusesModel;

ordersServicesModel = new OrdersServicesModel(this);

ordersServicesModel->setTable("orders\_services", &db);

return true;

}

else

return false;

}

void DatabaseModule::selectTables()

{

clientsModel->select();

managersModel->select();

typeOfServicesModel->select();

servicesModel->select();

filesModel->select();

ordersModel->select();

ordersServicesModel->select();

statusesModel->select();

}

Лістинг А.2 – Клас головного вікна

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

#include <QMessageBox>

#include "databasemodule.h"

#include "clientsfiltermodel.h"

#include "managersfiltermodel.h"

#include "typeofservicesfiltermodel.h"

#include "servicesfiltermodel.h"

#include "ordersfiltermodel.h"

#include "statusesfiltermodel.h"

#include "addeditclientsdialog.h"

#include "addeditmanagersdialog.h"

#include "addedittypesofservicesdialog.h"

#include "addeditservicesdialog.h"

#include "addeditordersdialog.h"

#include "addeditstatusesdialog.h"

QT\_BEGIN\_NAMESPACE

namespace Ui { class MainWindow; }

QT\_END\_NAMESPACE

class MainWindow : public QMainWindow

{

Q\_OBJECT

DatabaseModule dbm;

ClientsFilterModel\* clientsFilterModel;

ManagerFilterModel\* managerFilterModel;

TypeOfServicesFilterModel\* typeOfServicesFilterModel;

ServicesFilterModel\* servicesFilterModel;

OrderFilterModel\* ordersFilterModel;

StatusesFilterModel\* statusesFilterModel;

enum PAGES {

HELLO = 0,

CLIENTS,

MANAGERS,

TYPES\_OF\_SERVICES,

SERVICES,

ORDERS,

STATUSES,

STAT

};

public:

MainWindow(QWidget \*parent = nullptr);

~MainWindow();

private slots:

void on\_pushButton\_clients\_clicked();

void on\_pushButton\_managers\_clicked();

void on\_pushButton\_type\_of\_services\_clicked();

void on\_pushButton\_services\_clicked();

void on\_pushButton\_orders\_clicked();

void on\_pushButton\_statuses\_clicked();

void on\_pushButton\_clients\_search\_clicked();

void on\_pushButton\_managers\_search\_clicked();

void on\_pushButton\_types\_of\_services\_search\_clicked();

void on\_pushButton\_services\_search\_clicked();

void on\_pushButton\_orders\_search\_clicked();

void on\_pushButton\_statuses\_search\_clicked();

void on\_pushButton\_exit\_clicked();

void on\_pushButton\_clients\_add\_clicked();

void on\_pushButton\_clients\_edit\_clicked();

void on\_pushButton\_clients\_del\_clicked();

void on\_pushButton\_managers\_add\_clicked();

void on\_pushButton\_managers\_edit\_clicked();

void on\_pushButton\_managers\_del\_clicked();

void on\_pushButton\_types\_of\_services\_add\_clicked();

void on\_pushButton\_ypes\_of\_services\_edit\_clicked();

void on\_pushButton\_types\_of\_services\_del\_clicked();

void on\_pushButton\_services\_add\_clicked();

void on\_pushButton\_services\_edit\_clicked();

void on\_pushButton\_services\_del\_clicked();

void on\_pushButton\_orders\_add\_clicked();

void on\_pushButton\_orders\_edit\_clicked();

void on\_pushButton\_statuses\_add\_clicked();

void on\_pushButton\_statuses\_edit\_clicked();

void on\_pushButton\_statuses\_del\_clicked();

void on\_pushButton\_clicked();

void on\_pushButton\_2\_clicked();

void on\_pushButton\_3\_clicked();

void on\_pushButton\_4\_clicked();

private:

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

#include "mainwindow.h"

#include "ui\_mainwindow.h"

MainWindow::MainWindow(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

if(!dbm.connect())

{

QMessageBox::critical(nullptr, "Критична помилка!", "Неможливо під'єднатися до бази даних.\nПрограму буде завершено.");

close();

}

ui->stackedWidget->setCurrentIndex(PAGES::HELLO);

dbm.selectTables();

clientsFilterModel = new ClientsFilterModel(this);

managerFilterModel = new ManagerFilterModel(this);

typeOfServicesFilterModel = new TypeOfServicesFilterModel(this);

servicesFilterModel = new ServicesFilterModel(this);

ordersFilterModel = new OrderFilterModel(this);

statusesFilterModel = new StatusesFilterModel(this);

clientsFilterModel->setSourceModel(dbm.clientsModel);

managerFilterModel->setSourceModel(dbm.managersModel);

typeOfServicesFilterModel->setSourceModel(dbm.typeOfServicesModel);

servicesFilterModel->setSourceModel(dbm.servicesModel);

ordersFilterModel->setSourceModel(dbm.ordersModel);

statusesFilterModel->setSourceModel(dbm.statusesModel);

ui->tableView\_clients->setModel(clientsFilterModel);

ui->tableView\_managers->setModel(managerFilterModel);

ui->listView\_types\_of\_services->setModel(typeOfServicesFilterModel);

ui->tableView\_services->setModel(servicesFilterModel);

ui->tableView\_orders->setModel(ordersFilterModel);

ui->listView\_statuses->setModel(statusesFilterModel);

ui->comboBox\_clients\_orders\_search->setModel(clientsFilterModel);

ui->comboBox\_type\_services\_search->setModel(typeOfServicesFilterModel);

ui->comboBox\_status\_orders\_search->setModel(statusesFilterModel);

ui->tableView\_zap\_1->setModel(dbm.zap1\_model);

ui->tableView\_zap\_2->setModel(dbm.zap2\_model);

ui->tableView\_zap\_3->setModel(dbm.zap3\_model);

}

MainWindow::~MainWindow()

{

delete ui;

}

void MainWindow::on\_pushButton\_clients\_clicked()

{

ui->stackedWidget->setCurrentIndex(PAGES::CLIENTS);

}

void MainWindow::on\_pushButton\_managers\_clicked()

{

ui->stackedWidget->setCurrentIndex(PAGES::MANAGERS);

}

void MainWindow::on\_pushButton\_type\_of\_services\_clicked()

{

ui->stackedWidget->setCurrentIndex(PAGES::TYPES\_OF\_SERVICES);

}

void MainWindow::on\_pushButton\_services\_clicked()

{

ui->stackedWidget->setCurrentIndex(PAGES::SERVICES);

}

void MainWindow::on\_pushButton\_orders\_clicked()

{

ui->stackedWidget->setCurrentIndex(PAGES::ORDERS);

}

void MainWindow::on\_pushButton\_statuses\_clicked()

{

ui->stackedWidget->setCurrentIndex(PAGES::STATUSES);

}

void MainWindow::on\_pushButton\_clients\_search\_clicked()

{

if(ui->lineEdit\_pib\_clients\_search->text().size() > 0)

{

clientsFilterModel->setEnabledFilterParam(ClientsFilterModel::FilterParam::PIB, true);

clientsFilterModel->setFilterParam(ClientsFilterModel::FilterParam::PIB, ui->lineEdit\_pib\_clients\_search->text());

}

else

clientsFilterModel->setEnabledFilterParam(ClientsFilterModel::FilterParam::PIB, false);

if(ui->lineEdit\_phoneNumber\_clients\_search->text().size() > 0)

{

clientsFilterModel->setEnabledFilterParam(ClientsFilterModel::FilterParam::PHONE\_NUMBER, true);

clientsFilterModel->setFilterParam(ClientsFilterModel::FilterParam::PHONE\_NUMBER, ui->lineEdit\_phoneNumber\_clients\_search->text());

}

else

clientsFilterModel->setEnabledFilterParam(ClientsFilterModel::FilterParam::PHONE\_NUMBER, false);

if(ui->lineEdit\_email\_clients\_search->text().size() > 0)

{

clientsFilterModel->setEnabledFilterParam(ClientsFilterModel::FilterParam::EMAIL, true);

clientsFilterModel->setFilterParam(ClientsFilterModel::FilterParam::EMAIL, ui->lineEdit\_email\_clients\_search->text());

}

else

clientsFilterModel->setEnabledFilterParam(ClientsFilterModel::FilterParam::EMAIL, false);

}

void MainWindow::on\_pushButton\_managers\_search\_clicked()

{

if(ui->lineEdit\_pib\_managers\_search->text().size() > 0)

{

managerFilterModel->setEnabledFilterParam(ManagerFilterModel::FilterParam::PIB, true);

managerFilterModel->setFilterParam(ManagerFilterModel::FilterParam::PIB, ui->lineEdit\_pib\_managers\_search->text());

}

else

managerFilterModel->setEnabledFilterParam(ManagerFilterModel::FilterParam::PIB, false);

if(ui->lineEdit\_phoneNumber\_managers\_search->text().size() > 0)

{

managerFilterModel->setEnabledFilterParam(ManagerFilterModel::FilterParam::PHONE\_NUMBER, true);

managerFilterModel->setFilterParam(ManagerFilterModel::FilterParam::PHONE\_NUMBER, ui->lineEdit\_phoneNumber\_managers\_search->text());

}

else

managerFilterModel->setEnabledFilterParam(ManagerFilterModel::FilterParam::PHONE\_NUMBER, false);

}

void MainWindow::on\_pushButton\_types\_of\_services\_search\_clicked()

{

if(ui->lineEdit\_title\_types\_of\_services\_search->text().size() > 0)

{

typeOfServicesFilterModel->setEnabledFilterParam(TypeOfServicesFilterModel::FilterParam::TITLE, true);

typeOfServicesFilterModel->setFilterParam(TypeOfServicesFilterModel::FilterParam::TITLE, ui->lineEdit\_title\_types\_of\_services\_search->text());

}

else

typeOfServicesFilterModel->setEnabledFilterParam(TypeOfServicesFilterModel::FilterParam::TITLE, false);

}

void MainWindow::on\_pushButton\_services\_search\_clicked()

{

if(ui->lineEdit\_title\_services\_search->text().size() > 0)

{

servicesFilterModel->setEnabledFilterParam(ServicesFilterModel::FilterParam::TITLE, true);

servicesFilterModel->setFilterParam(ServicesFilterModel::FilterParam::TITLE, ui->lineEdit\_title\_services\_search->text());

}

else

servicesFilterModel->setEnabledFilterParam(ServicesFilterModel::FilterParam::TITLE, false);

if(ui->checkBox\_enable\_type\_combo->isChecked())

{

servicesFilterModel->setEnabledFilterParam(ServicesFilterModel::FilterParam::TYPE, true);

servicesFilterModel->setFilterParam(ServicesFilterModel::FilterParam::TYPE, dbm.typeOfServicesModel->index(typeOfServicesFilterModel->mapToSource(typeOfServicesFilterModel->index(ui->comboBox\_type\_services\_search->currentIndex(), 0)).row(), TypeOfServicesModel::Column::ID).data(TypeOfServicesModel::Role::Read));

}

else

servicesFilterModel->setEnabledFilterParam(ServicesFilterModel::FilterParam::TYPE, false);

}

void MainWindow::on\_pushButton\_orders\_search\_clicked()

{

if(ui->lineEdit\_number\_orders\_search->text().size() > 0)

{

ordersFilterModel->setEnabledFilterParam(OrderFilterModel::FilterParam::NUMBER, true);

ordersFilterModel->setFilterParam(OrderFilterModel::FilterParam::NUMBER, ui->lineEdit\_number\_orders\_search->text());

}

else

ordersFilterModel->setEnabledFilterParam(OrderFilterModel::FilterParam::NUMBER, false);

if(ui->checkBox\_enabled\_clients\_combo->isChecked())

{

ordersFilterModel->setEnabledFilterParam(OrderFilterModel::FilterParam::CLIENT, true);

ordersFilterModel->setFilterParam(OrderFilterModel::FilterParam::CLIENT, dbm.clientsModel->index(clientsFilterModel->mapToSource(clientsFilterModel->index(ui->comboBox\_clients\_orders\_search->currentIndex(), 0)).row(), ClientsModel::Column::ID).data(ClientsModel::Role::Read));

}

else

ordersFilterModel->setEnabledFilterParam(OrderFilterModel::FilterParam::CLIENT, false);

if(ui->checkBox\_enable\_statuses\_combo->isChecked())

{

ordersFilterModel->setEnabledFilterParam(OrderFilterModel::FilterParam::STATUS, true);

ordersFilterModel->setFilterParam(OrderFilterModel::FilterParam::STATUS, dbm.statusesModel->index(statusesFilterModel->mapToSource(statusesFilterModel->index(ui->comboBox\_status\_orders\_search->currentIndex(), 0)).row(), StatusesModel::Column::ID).data(StatusesModel::Role::Read));

}

else

ordersFilterModel->setEnabledFilterParam(OrderFilterModel::FilterParam::STATUS, false);

}

void MainWindow::on\_pushButton\_statuses\_search\_clicked()

{

if(ui->lineEdit\_title\_statuses\_search->text().size() > 0)

{

statusesFilterModel->setEnabledFilterParam(StatusesFilterModel::FilterParam::TITLE, true);

statusesFilterModel->setFilterParam(StatusesFilterModel::FilterParam::TITLE, ui->lineEdit\_title\_statuses\_search->text());

}

else

statusesFilterModel->setEnabledFilterParam(StatusesFilterModel::FilterParam::TITLE, false);

}

void MainWindow::on\_pushButton\_exit\_clicked()

{

close();

}

void MainWindow::on\_pushButton\_clients\_add\_clicked()

{

AddEditClientsDialog dialog;

dialog.exec();

if(!dialog.isSave())

return;

dbm.clientsModel->appendRow(dialog.getData());

dbm.clientsModel->saveChanges();

dbm.clientsModel->select();

}

void MainWindow::on\_pushButton\_clients\_edit\_clicked()

{

if(ui->tableView\_clients->currentIndex().row() == -1)

return;

AddEditClientsDialog dialog;

dialog.setDialog(dbm.clientsModel->getObjectByRow(clientsFilterModel->mapToSource(ui->tableView\_clients->currentIndex()).row()));

dialog.exec();

if(!dialog.isSave())

return;

dbm.clientsModel->updateRow(clientsFilterModel->mapToSource(ui->tableView\_clients->currentIndex()).row(), dialog.getData());

dbm.clientsModel->saveChanges();

dbm.clientsModel->select();

}

void MainWindow::on\_pushButton\_clients\_del\_clicked()

{

if(ui->tableView\_clients->currentIndex().row() == -1)

return;

dbm.clientsModel->removeRow(clientsFilterModel->mapToSource(ui->tableView\_clients->currentIndex()).row());

dbm.clientsModel->saveChanges();

dbm.clientsModel->select();

}

void MainWindow::on\_pushButton\_managers\_add\_clicked()

{

AddEditManagersDialog dialog;

dialog.exec();

if(!dialog.isSave())

return;

dbm.managersModel->appendRow(dialog.getData());

dbm.managersModel->saveChanges();

dbm.managersModel->select();

}

void MainWindow::on\_pushButton\_managers\_edit\_clicked()

{

if(ui->tableView\_managers->currentIndex().row() == -1)

return;

AddEditManagersDialog dialog;

dialog.setDialog(dbm.managersModel->getObjectByRow(managerFilterModel->mapToSource(ui->tableView\_managers->currentIndex()).row()));

dialog.exec();

if(!dialog.isSave())

return;

dbm.managersModel->updateRow(managerFilterModel->mapToSource(ui->tableView\_managers->currentIndex()).row(), dialog.getData());

dbm.managersModel->saveChanges();

dbm.managersModel->select();

}

void MainWindow::on\_pushButton\_managers\_del\_clicked()

{

if(ui->tableView\_managers->currentIndex().row() == -1)

return;

dbm.managersModel->removeRow(managerFilterModel->mapToSource(ui->tableView\_managers->currentIndex()).row());

dbm.managersModel->saveChanges();

dbm.managersModel->select();

}

void MainWindow::on\_pushButton\_types\_of\_services\_add\_clicked()

{

AddEditTypesOfServicesDialog dialog;

dialog.exec();

if(!dialog.isSave())

return;

dbm.typeOfServicesModel->appendRow(dialog.getData());

dbm.typeOfServicesModel->saveChanges();

dbm.typeOfServicesModel->select();

}

void MainWindow::on\_pushButton\_ypes\_of\_services\_edit\_clicked()

{

if(ui->listView\_types\_of\_services->currentIndex().row() == -1)

return;

AddEditTypesOfServicesDialog dialog;

dialog.setDialog(dbm.typeOfServicesModel->getObjectByRow(typeOfServicesFilterModel->mapToSource(ui->listView\_types\_of\_services->currentIndex()).row()));

dialog.exec();

if(!dialog.isSave())

return;

dbm.typeOfServicesModel->updateRow(typeOfServicesFilterModel->mapToSource(ui->listView\_types\_of\_services->currentIndex()).row(), dialog.getData());

dbm.typeOfServicesModel->saveChanges();

dbm.typeOfServicesModel->select();

}

void MainWindow::on\_pushButton\_types\_of\_services\_del\_clicked()

{

if(ui->listView\_types\_of\_services->currentIndex().row() == -1)

return;

dbm.typeOfServicesModel->removeRow(typeOfServicesFilterModel->mapToSource(ui->listView\_types\_of\_services->currentIndex()).row());

dbm.typeOfServicesModel->saveChanges();

dbm.typeOfServicesModel->select();

}

void MainWindow::on\_pushButton\_services\_add\_clicked()

{

AddEditServicesDialog dialog;

dialog.setDialog(dbm.filesModel, dbm.typeOfServicesModel);

dialog.exec();

if(!dialog.isSave())

return;

dbm.servicesModel->appendRow(dialog.getData());

dbm.servicesModel->saveChanges();

dbm.servicesModel->select();

}

void MainWindow::on\_pushButton\_services\_edit\_clicked()

{

if(ui->tableView\_services->currentIndex().row() == -1)

return;

AddEditServicesDialog dialog;

dialog.setDialog(dbm.servicesModel->getObjectByRow(servicesFilterModel->mapToSource(ui->tableView\_services->currentIndex()).row()), dbm.filesModel, dbm.typeOfServicesModel);

dialog.exec();

if(!dialog.isSave())

return;

dbm.servicesModel->updateRow(servicesFilterModel->mapToSource(ui->tableView\_services->currentIndex()).row(), dialog.getData());

dbm.servicesModel->saveChanges();

dbm.servicesModel->select();

}

void MainWindow::on\_pushButton\_services\_del\_clicked()

{

if(ui->tableView\_services->currentIndex().row() == -1)

return;

dbm.servicesModel->removeRow(servicesFilterModel->mapToSource(ui->tableView\_services->currentIndex()).row());

dbm.servicesModel->saveChanges();

dbm.servicesModel->select();

}

void MainWindow::on\_pushButton\_orders\_add\_clicked()

{

AddEditOrdersDialog dialog;

dialog.setDialog(dbm.clientsModel, dbm.managersModel, dbm.servicesModel, dbm.ordersServicesModel, dbm.statusesModel);

dialog.exec();

if(!dialog.isSave())

return;

dbm.ordersModel->appendRow(dialog.getOrderData());

QVariant number = dbm.ordersModel->saveChanges();

dbm.ordersModel->select();

QList<orderServiceObject> listServices = dialog.getOrderServicesData();

for(int i = 0; i < listServices.size(); i++)

listServices[i][OrdersServicesModel::Column::ID\_ORDER] = number;

for(int i = 0; i < listServices.size(); i++)

dbm.ordersServicesModel->appendRow(listServices[i]);

dbm.ordersServicesModel->saveChanges();

dbm.ordersServicesModel->select();

}

void MainWindow::on\_pushButton\_orders\_edit\_clicked()

{

if(ui->tableView\_orders->currentIndex().row() == -1)

return;

AddEditOrdersDialog dialog;

dialog.setDialog(dbm.ordersModel->getObjectByRow(ordersFilterModel->mapToSource(ui->tableView\_orders->currentIndex()).row()), dbm.clientsModel, dbm.managersModel, dbm.servicesModel, dbm.ordersServicesModel, dbm.statusesModel);

dialog.exec();

if(!dialog.isSave())

return;

dbm.ordersModel->updateRow(ordersFilterModel->mapToSource(ui->tableView\_orders->currentIndex()).row(), dialog.getOrderData());

dbm.ordersModel->saveChanges();

dbm.ordersModel->select();

}

void MainWindow::on\_pushButton\_statuses\_add\_clicked()

{

AddEditStatusesDialog dialog;

dialog.exec();

if(!dialog.isSave())

return;

dbm.statusesModel->appendRow(dialog.getData());

dbm.statusesModel->saveChanges();

dbm.statusesModel->select();

}

void MainWindow::on\_pushButton\_statuses\_edit\_clicked()

{

if(ui->listView\_statuses->currentIndex().row() == -1)

return;

AddEditStatusesDialog dialog;

dialog.setDialog(dbm.statusesModel->getObjectByRow(statusesFilterModel->mapToSource(ui->listView\_statuses->currentIndex()).row()));

dialog.exec();

if(!dialog.isSave())

return;

dbm.statusesModel->updateRow(statusesFilterModel->mapToSource(ui->listView\_statuses->currentIndex()).row(), dialog.getData());

dbm.statusesModel->saveChanges();

dbm.statusesModel->select();

}

void MainWindow::on\_pushButton\_statuses\_del\_clicked()

{

if(ui->listView\_statuses->currentIndex().row() == -1)

return;

dbm.statusesModel->removeRow(statusesFilterModel->mapToSource(ui->listView\_statuses->currentIndex()).row());

dbm.statusesModel->saveChanges();

dbm.statusesModel->select();

}

void MainWindow::on\_pushButton\_clicked()

{

dbm.zap1.exec();

dbm.zap2.exec();

dbm.zap3.exec();

ui->stackedWidget\_2->setCurrentIndex(0);

ui->stackedWidget->setCurrentIndex(PAGES::STAT);

}

void MainWindow::on\_pushButton\_2\_clicked()

{

ui->stackedWidget\_2->setCurrentIndex(0);

}

void MainWindow::on\_pushButton\_3\_clicked()

{

ui->stackedWidget\_2->setCurrentIndex(1);

}

void MainWindow::on\_pushButton\_4\_clicked()

{

ui->stackedWidget\_2->setCurrentIndex(2);

}

Лістинг А.3 – QSS стилі головного вікна

.var {

cherniy: #000d0f;

siniy: #284256;

rozoviy: #b59683;

temn\_bejeviy: #ebebd8;

svetl\_bejeviy: #eeeee4;

}

#centralwidget {

background-color: #b59683;

}

#centralwidget #splitter {

border-radius: 15%;

background-color: #ebebd8;

}

#frame\_main\_menu {

min-width: 200px;

border: 3px solid #b59683;

border-radius: 15%;

}

#frame\_main\_menu QPushButton[objectName\*="exit"] {

width: 120;

height: 30;

padding: 5px;

font: bold;

color: #284256;

border: solid;

border-width: 5px;

border-radius:20px;

border-color: #284256;

background-color: #eeeee4;

}

#frame\_main\_menu QPushButton[objectName\*="exit"]::pressed {

color: #eeeee4;

background-color: #284256;

}

#frame\_workspace, #stackedWidget QFrame{

border: 3px solid #b59683;

border-radius: 15%;

}

#stackedWidget QFrame[objectName^="frame"]{

background-color: #ebebd8;

}

QObject[objectName\*="page"] {

background-color: #ebebd8;

}

QLabel {

color: #b59683;

}

QLabel[objectName\*="title"] {

padding: 5px;

color: #eeeee4;

border-radius: 15%;

background-color: #b59683;

}

QLabel[objectName\*="error"] {

color: #284256;

font-style: italic;

text-align: center right;

margin-bottom: 10px;

}

#frame\_workspace QPushButton {

width: 90;

height: 20;

padding: 5px;

font: bold;

color: #eeeee4;

border: solid;

border-width: 5px;

border-radius:20px;

border-color: #b59683;

background-color: #b59683;

}

#frame\_workspace QPushButton::pressed {

color: #b59683;

background-color: #eeeee4;

}

#frame\_workspace QPushButton[objectName\*="search"], #frame\_workspace QPushButton[objectName\*="add"]

{

width: 90;

height: 20;

padding: 5px;

font: bold;

color: #eeeee4;

border: solid;

border-width: 5px;

border-radius:20px;

border-color: #284256;

background-color: #284256;

}

#frame\_workspace QPushButton[objectName\*="search"]::pressed, #frame\_workspace QPushButton[objectName\*="add"]::pressed {

border-width: 5px;

color: #284256;

background-color: #eeeee4;

}

#frame\_workspace QPushButton[objectName\*="del"] {

width: 90;

height: 20;

padding: 5px;

font: bold;

color: #284256;

border: solid;

border-width: 5px;

border-radius:20px;

border-color: #284256;

background-color: #eeeee4;

}

#frame\_workspace QPushButton[objectName\*="del"]::pressed {

color: #eeeee4;

background-color: #284256;

}

QLineEdit {

width: 200;

height: 20;

padding: 5px;

font: bold;

color: #284256;

border: solid;

border-width: 5px;

border-radius:20px;

border-color: #b59683;

background-color: #eeeee4;

}

QLineEdit::focus {

border-color: #284256;

}

QLineEdit::placeholder {

font-style: italic;

}

QComboBox {

width: 90;

height: 20;

padding: 5px;

font: bold;

color: #b59683;

border: solid;

border-width: 5px;

border-radius:20px;

border-color: #b59683;

background-color: #eeeee4;

}

QComboBox:hover {

}

QComboBox::drop-down {

subcontrol-origin: padding;

subcontrol-position: top right;

width: 15px;

border-left: 5px solid #b59683;

}

QComboBox::down-arrow {

}

QComboBox QListView {

padding: 5;

}

QComboBox::selected {

color: #ebebd8;

background-color: #284256;

}

QAbstractItemView {

border:30px solid #b59683;

border-radius: 15%;

background-color: #eeeee4;

}

QAbstractItemView::item {

color: #b59683;

border: 1px solid #b59683;

}

QAbstractItemView::item::selected {

color: #eeeee4;

background-color: #284256;

border-color: #284256;

}

QAbstractItemView::selected {

color: #eeeee4;

background-color: #284256;

}

QAbstractItemView QHeaderView {

}

QAbstractItemView QHeaderView::selected {

}

QAbstractItemView::section {

padding: 5px;

color:#eeeee4;

background-color: #b59683;

border: none;

border-right: 1px solid #ebebd8;

}

QGroupBox {

color: #b59683;

margin-top: 25px;

border: solid;

border-width: 3px;

border-radius:20px;

border-color: #b59683;

background-color: #eeeee4;

}

QGroupBox::title {

font-weight: bold;

}

QTableView::item {

color: #b59683;

border: 1px solid #b59683;

}

QTableView::item::selected {

color: #eeeee4;

background-color: #284256;

border-color: #284256;

}

QTableView::selected {

color: #eeeee4;

background-color: #284256;

}

QTableView QHeaderView, QTableView QHeaderView {

padding: 1px;

border: none;

}

QTableView QHeaderView::selected {

}

QTableView::section {

padding: 5px;

color:#eeeee4;

background-color: #b59683;

border: none;

border-right: 1px solid #ebebd8;

border-radius: 2px;

}

#frame\_main\_menu QPushButton, #frame QPushButton{

width: 120;

height: 30;

padding: 5px;

font: bold;

color: #b59683;

border: solid;

border-width: 5px;

border-radius:20px;

border-color: #b59683;

background-color: #eeeee4;

}

#frame\_main\_menu QPushButton::checked, #frame QPushButton::checked {

color: #eeeee4;

background-color: #b59683;

}

#frame\_main\_menu QPushButton::hover, #frame QPushButton::hover {

color: #eeeee4;

background-color: #b59683;

}