НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

**Факультет інформаційних технологій**

**УДК 004.9:339.3**

| | |
|---|---|
| **«ПОГОДЖЕНО»** | **«ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ»** |
| Декан факультету інформаційних технологій | Завідувач кафедри комп'ютерних наук |
| Глазунова О.Г., д.п.н., професор | Голуб Б.Л., к.т.н., доцент |
| «____» _____ 2021 р. | «____» _____ 2021 р. |

**МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА**

**на тему** «Аналітична система управління POS-системи торгової мережі»

Спеціальність 122 «Комп'ютерні науки»

Освітня програма «Інформаційні управляючі системи та технології»

Орієнтація освітньої програми освітньо-професійна

**Гарант освітньої програми «Інформаційні управляючі системи та технології»**

д.т.н, доцент _____ Бондаренко В.Є.

**Керівник магістерської кваліфікаційної роботи**

_____ Ящук Д.Ю.,

**Виконав**

_____ Адаменко А.Ю.

**КИЇВ-2021**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

**Факультет інформаційних технологій**

**ЗАТВЕРДЖУЮ**
**Завідувач кафедри комп'ютерних наук**
**Голуб Б.Л., к.т.н., доцент**

"29" жовтня 2020 року

**З А В Д А Н Н Я**

**ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ**

Адаменку Андрію Юрійовичу

Спеціальність 122 «Комп'ютерні науки»
Освітня програма Інформаційні управляючі системи та технології
Орієнтація освітньої програми освітньо-професійна
Тема магістерської кваліфікаційної роботи «Аналітична система управління POS-системи торгової мережі»
затверджена наказом ректора НУБіП України від "29" жовтня 2020р. №1634 «С»

Термін подання завершеної роботи на кафедру "30" листопада 2021р.

Вихідні дані до магістерської кваліфікаційної роботи:
1) дані про дату і час транзакції торгової мережі;
2) дані про дату і тривалість кожного проміжку обслуговування.

Перелік питань, що підлягають дослідженню:

| № з/п | Питання, що підлягає дослідженню | Строк виконання | Примітка |
|---|---|---|---|
| 1. | Аналіз предметної області | 21.09.2020 - 24.10.2020 | |
| 2. | Дослідження інструментів Qt Creator | 30.11.2020 - 31.12.2020 | |
| 3. | Проектування системи | 01.02.2021-06.03.2021 | |
| 4. | Дослідження інструментів MS SQL | 11.03.2021-10.04.2021 | |
| 5. | Розробка алгоритмів аналізу даних | 22.04.2021-22.05.2021 | |
| 6. | Дослідження отриманих результатів | 01.09.2021-13.11.2021 | |
| 7. | Попередній захист | 30.11.2021 | |
| 8. | Захист | 14.12.2021 | |

**Дата видачі завдання** "29"жовтня 2020 р.

**Керівник магістерської кваліфікаційної роботи** _____ Ящук Д. Ю.
( підпис ) (прізвище та ініціали)

**Завдання прийняв до виконання** _____ Адаменко А. Ю.
(підпис) (прізвище та ініціали)

# ЗМІСТ

# ВСТУП

У сучасному світі інформація є одним з основних ресурсів і водночас, однією з рушійних сил розвитку людського суспільства, як правило, в часі потоки інформації збільшуються тому в будь-якій організації, як великій або маленькій, виникає проблема такої організації управління даними, яка забезпечила б найбільш ефективну роботу.

У зв'язку зі зростаючою складністю роботи сучасного торгового підприємства, що функціонує в жорстких умовах ринкової економіки, системи автоматизації роздрібної торгівлі стають об'єктивною необхідністю.

В даний час системи автоматизації торгових підприємств дозволяють відрегулювати і впорядкувати роботу з постачальниками, організувати кількісний і сумовий облік реалізованого товару за рахунок маркування товару внутрішнім ( локальним) кодом.

Щоб забезпечити повну автоматизацію торгового підприємства, необхідно придбати відповідне програмно-апаратне забезпечення і на його основі створити локальну обчислювальну мережу у локальній обчислювальній мережі торгового підприємства об'єднані комп'ютери (робочі станції), сервер, на якому встановлено програмне забезпечення автоматизації торгового підприємства, а також касові POS термінали.

Робочі станції призначені для роботи операторів торгово-сервісної системи, бухгалтерії та комерційного відділу.

***Актуальність теми***: Наразі в світі не так багато систем для управління торговими мережами з POS-терміналами, які знаходяться у вільному доступі та надають користувачеві функціонал для побудови, управління та аналітики підприємства, а також системи яка може масштабуватися в залежності від потреб користувача.

***Об'єкт*** – роздрібна торгівельна мережа

***Предмет*** – програмна реалізація і дослідження аналітичної системи для торгової мережі

***Мета роботи***. Метою є підвищення рівня автоматизації процесів аналітики підприємства торгівлі та обслуговування гостей шляхом створення системи.

***Завдання*** – необхідно створити аналітичну систему для підвищення рівня планування робочого часу працівників.

***Методи дослідження.*** При проведенні досліджень були використані певні стандартні програмні продукти, такі як: SQL Server та Qt Creator.

***Наукова новизна***. У процесі дослідження були запропоновані вдосконалення для процесу аналітики торгової точки.

Апробація результатів дослідження. В процесі розробки магістерської роботи на тему «Аналітична система управління pos-системи торгової мережі» були проведені виступи на XII Міжнародній науково-практичній конференції студентів, аспірантів та молодих вчених «ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ: ЕКОНОМІКА, ТЕХНІКА, ОСВІТА».

***Структура роботи.***

Робота містить 56 сторінок.

В дослідженні використано 22 джерела.

Структурні елементи роботи розміщені в такій послідовності.

У першому розділі проведено системний аналіз предметної області магістерського дослідження, описані існуючі системи, поставлені задачі дослідження.

Другий розділ присвячено проєктуванню системи, описана розробка специфікації вимог SRS, змодельована діагарма прецедентів, побудована діаграма класів.

У третьому розділі описується реалізація системи, побудована діаграма діяльності

У четвертому розділі розглянуто результати дослідження, представлено тестування системи.

# 1. СИСТЕМНИЙ АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## І.1. Загальна характеристика POS-системи

Для управління роздрібним магазином необхідно володіти рядом спеціальних знань в області маркетингу, менеджменту та адміністративного управління. Забезпечення необхідної кількості запасів, складання щомісячних звітів про продажі і своєчасна виплата зарплати – це ті навички, без яких не обійтися в роботі будь-якого магазину.

Наявність ефективної системи торгових точок (Pos) має велике значення для забезпечення того, щоб всі ваші операції працювали в унісон, а торговці, які витрачають час на впровадження такої системи до того, поки вона їм не знадобиться, отримають вигоду.

POS є центральним елементом бізнесу, це центр, де все — наприклад, продажі, управління запасами, обробка платежів або управління клієнтами – об'єднується.

А за результатами дослідження, яке провела компанія Software Advice для представників роздрібних торговців минулої осені, майже половина опитаних бізнес—лідерів (22%) приділяли більше уваги своїй інфраструктурній системі (модернізація існуючої системи або інвестиція в нову систему POS) під час проведення COVID—19. У той же час, навіть під час глобальної пандемії 2/3 лідерів роздрібної торгівлі не побажали або не змогли знизити пріоритетність своїх систем POS.

Які б очевидні переваги не були у POS-системи – деяким підприємствам вдається збільшити продажі на цілих 200% тільки за рахунок функції звітності – все ще існують підприємства, які використовують певну комбінацію ручних методів,

касові апаратів, програмного забезпечення для бухгалтерського обліку та електронних таблиць для обробки транзакцій і реєстрації продажів.

## І.2. POS-системи у світі

Наразі у світі інформації існує безліч POS-систем заточених під певні задачі з якими може зіткнутися користувач. Як приклад існують такі як:

- Valigara – для ювелірних компаній

Valigara – це спеціальна платформа для управління електронною комерцією онлайн-ювелірних виробів для роздрібних торговців і виробників. Вона пропонує користувачам єдину централізовану платформу, на якій можна буде здійснювати управління всіма аспектами ювелірного бізнесу. Система управління інформацією про продукт (PIM) дозволяє користувачам перераховувати один об'єкт і публікувати його по декількох каналах. Оголошення з багатоканальними оголошеннями знижують ризик перепродажу товарів, завдяки автоматичному оновленню оголошень одночасно на всіх ринках. Для цього підприємства можуть використовувати платформу для управління веб-ресурсами, клієнтами, замовниками та маркетинговою діяльністю. Аналітика доступна, щоб допомогти маркетологам оптимізувати ефективність кампанії. За допомогою Платформи можна управляти запасами і обробляти виконання замовлень.

- Uzeli – для спа салонів

Uzel – це хмарне і місцеве рішення, розроблене для того, щоб допомогти малим і великим салоном і СПА-салонам управляти зустрічами клієнтів і операціями в точках продажів ( POS). У функції входять клієнтська база даних, нагадування про текстові повідомлення, реферальні програми, дані в реальному часі, управління списками очікування і звітність персоналу. Додаток містить кіоски самостійної реєстрації, які дозволяють співробітникам самостійно отримувати інформацію про клієнтів і вимоги з обслуговування. Салони можуть

використовувати Uzeli для проведення маркетингових кампаній з розповсюдження купонів і призначених для користувача винагород за лояльність. Це допомагає підприємствам керувати продажами продукції, відстежувати товарно-матеріальні запаси для поповнення і використовувати вбудований інструмент замовлення для моніторингу низького рівня запасів.

- Quant – роздрібна торгівля

Quant Retail – це хмарне програмне забезпечення для управління роздрібною торгівлею, яке включає в себе зручний редактор планів поверхів з розширеними функціями для ефективного управління торговими площами та управління категоріями. Платформа пропонує планограми, оптимізовані на основі продажів, розширену звітність, доступ до даних про продажі, Напівавтоматичне замовлення, багаторівневі плани поверхів, гнучку ієрархію категорій, поширення і зв'язок планограм, управління цінниками, 3D візуалізація, бібліотека продуктів, фотодокументація, управління завданнями і багато іншого....

Як можна побачити системи є дуже різні, але всіх їх об'єднує одне – вони всі використовують звітність в зручному вигляді.

## І.3. Постановка задач дослідження

Поставлену мету досягнемо шляхом розробки BackOffice системи для аналізу та створення аналітики по даним які надійдуть від POS-системи.

Сучасні BackOffice системи, що виникли у результаті злиття керівницьких інформаційних систем і систем керування базами даних, це системи, що максимально пристосовані до розв'язування задач щоденної керівницької діяльності, і є інструментом, щоб надати допомогу тим, хто не має достатнього рівня управління закладом.

В ході постановки задачі була змодельована наступна схема-макет взаємодії сервера з різними типами клієнтів системи, представлений на рис. 1.

Виходячи зі схеми – в системі присутні різні типи додатків потрібних для функціонування:

- Point of Sales (POS) – точка продажу товарів, каса;
- Thin Client (TCL) – тонкий клієнт для відображення на моніторах замовлень;
- Administrator Access Point (AAP) – програмне забезпечення, яке є вузлом для управління і програмування системи;
- Data base SQL – (DB SQL) – база даних, в якій будуть збережені всі дані про конфігурацію, продажі, продукти.

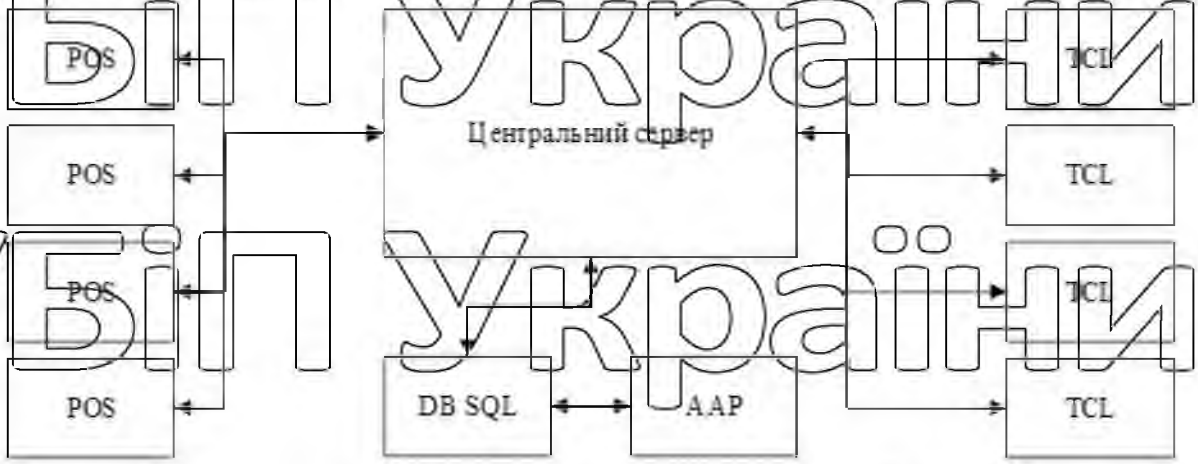


Рис. 1 Макет аналітичної системи

# 2. ПРОЕКТУВАННЯ СИСТЕМИ

«Думайте на папері. Кожна хвилина, витрачена на планування, економить 10 хвилин при здійсненні плану» – Брайан Трейсі.

## 2.1. Розробка специфікації вимог SRS

### 2.1.1. Вступ

2.1.1.1. Призначення специфікації

Ця специфікація вимог до ПЗ описує функціональні і не функціональні вимоги до додатку. Документ призначений для того, щоб встановити основні вимоги щодо того, як повинен функціонувати програмний продукт.

2.1.1.2. Область дії специфікації

Аналітична система призначена для перегляду даних про продажі, формування звітів на основі отриманих даних від POS-системи, та аналізу цих даних. Описаний функціонал системи буде автоматично і у зручному для користувача форматі подавати шукані дані в автоматичному режимі.

2.1.1.3 Визначення акронімів та скорочень

Мова програмування – мова, яка призначена для створення комп'ютерних програм. Мова програмування визначає набір лексичних синтаксичних та семантичних правил, який визначає зовнішній вигляд програми та дії, які виконає комп'ютер під її керуванням.

ПЗ (Програмне забезпечення) – програма чи сукупність програм, які використовуються для управління комп'ютером.

Інформаційна система – система для зберігання, пошуку і обробки інформації та супутні їй організаційні ресурси, які забезпечують та поширюють інформацію.

MVP (minimum viable product) — це мінімально життєздатний продукт, який дозволяє отримати осмислений зворотній зв'язок від користувачів, зрозуміти що їх необхідно та не створювати те, що їм нецікаво та за що вони не готові платити.

MPA (англ. Multi Page Application – багатосторінковий додаток) – це додаток який використовує більше ніж одне вікно для реалізації програмного продукту.

ACL (англ. Access control list - Список контролю доступу) – використовується для надання різних прав доступу різним користувачам.

2.1.1.4. Короткий огляд специфікації

SRS є достатньо важливою частиною процесу складання вимог в життєвому циклі ПЗ і використовується при проектуванні, реалізації, моніторингу проекту, верифікації та перевірці правильності. Розроблена SRS відноситься не до процесу його створення, а до самого програмного виробу. Для цього розроблена специфікація вимог яка містить два основних розділи:

– технічні вимоги – розділ описує загальні фактори, які впливають на продукт, включає в собі список необхідних для системи інтерфейсів і програмних компонентів, з якими система повинна взаємодіяти, а також вимоги до апаратного забезпечення. Забезпечує попередні відомості про ті вимоги, які більш докладно визначаються в розділі «Специфічні вимоги».

– специфічні вимоги – розділ який описує основні функції системи, які згруповані за типами користувачів, а також містять всі вимоги до програмного забезпечення на рівні деталізації, достатньої, щоб дати проектувальникам можливість розробити систему, що задовольняє цим вимогам, а її випробувачам – переконатися, що система задовольняє цим вимогам.

**2.1.2. Технічні вимоги**

2.1.2.1. Контекст виробу

Система на момент складання документу є абсолютно новим продуктом та не має попередніх її версій. Документ описує MVP функціонал системи.

2.1.2.2. Вимоги до системних, мережевих та апаратних інтерфейсів системи.

Через те, що тип ПЗ має на увазі взаємодію з мережею, виділено основні вимоги до операційного середовища, на якому повинна бути розгорнута програма:

- система повинна бути розгорнута на обладнанні з встановленою операційною системою Windows, не нижче Windows 10;
- для написання кодів програми повинна використовуватися мова програмування яка володіє сумісністю з системами сімейства Windows.
- мережевий зв'язок з віддаленими серверами повинен бути реалізований за допомогою протоколу TCP.
- для реалізації інтерфейсу користувача система повинна використовувати технологію MPA.

2.1.2.3. Компоненти та зовнішні інтерфейси програмного забезпечення.

Для реалізацій функцій, які система повинна надавати користувачам, вона під час створення MVP може складатися з одного компонента та точки продажу:

- клієнт-сервер – компонент, який відповідає за відображення графічного інтерфейсу користувача.
- Point of Sales (POS) – точка продажу товарів, каса.

2.1.2.4. Вимоги до операцій, що надаються користувачам.

Основними операціями, що надаються користувачам, є посилання запиту до бази даних та робота з отриманими даними для яких система повинна надавати користувачам графічний інтерфейс.

У разі виникнення помилок під час виконання операцій, система повинна сповістити користувача про можливі варіанти вирішення проблеми, а також вміти вживати заходи для їх запобігання в майбутньому, якщо це можливо.

2.1.2.5. Вимоги до інтерфейсу користувача.

Графічний інтерфейс користувача повинен бути захищений від нетипових дій користувача.

Якщо відбувається будь-яка помилка в результаті дій системи або користувальницьких некоректних дій, користувач повинен отримати повідомлення про те, що стався збій.

### 2.1.3. Специфічні вимоги

2.1.3.1. Характеристика користувачів за функціями які повинні їм надаватись

Так, як в системі передбачений не лише один користувач, то будуть передбачені різні рівні доступу за допомогою ACL.

2.1.3.2 Функціональні вимоги

Функціональні вимоги описують функції, які має виконувати ПЗ, яке буде розроблюватись.

Функціональна вимога 1 – Система повинна надавати функцію перегляду кількість обслугованих гостей за обрану величину проміжку часу. Для цього:

– система повинна надати інтерфейс з візуальним компонентом календаря на правій частині вікна. По кліку на вибрану дату – календар змінюється і система відправляє запит до бази даних, щоб отримати вибірку.

– внизу, після компонента календаря, повинен бути компонент для вибірки проміжків часу

– праворуч від компонента календаря, повинна бути таблиця з результатами.

Функціональна вимога 2 – Система повинна надавати дані в повному обсязі, отриманих від сервера. Для цього, таблиця з даними повинна містити такі поля:

– обраний проміжок часу

– проектовану кількість транзакцій

– фактичну кількість транзакцій

Функціональна вимога 3 – система повинна мати довідку у якій буде описано принцип користування системою.

Функціональна вимога 4 – система повинна мати дані для зворотного зв’язку. Для цього:

– Потрібно реалізувати окреме вікно з даними для зворотного зв’язку

Функціональна вимога 5 – Система повинна надавати функцію перегляду часу обслуговуваних гостей за обрану величину проміжку часу. Для цього:

– система повинна надати інтерфейс з візуальним компонентом календаря на правій частині вікна. По кліку на вибрану дату – календар змінюється і система відправляє запит до бази даних, щоб отримати вибірку.

– внизу, після компонента календаря, повинен бути компонент для вибірки проміжків часу

– праворуч від компонента календаря, повинна бути таблиця з результатами.

Функціональна вимога 6 – Система повинна надавати дані в повному обсязі, отриманих від сервера. Для цього, таблиця з даними повинна містити такі поля:

– обраний проміжок часу

– потрібний час обслуговування

– фактичний час обслуговування.

## 2.2. Моделювання системи

### 2.2.1. Вступ

Моделювання програмного забезпечення є важливим етапом в проектуванні системи [1]. Даний етап забезпечує побудову і вивчення моделей системи для отримання інформації про процеси, які в ній відбуваються. В процесу моделювання часто вносяться зміни в систему, спрощуючи або ускладнюючи її, шляхом видалення або додавання в неї нових функцій.

При моделювання програмних систем в більшості випадків використовують мову моделювання UML (англ. Unified Modeling Language) для визначення, візуалізації створення і документування компонентів системи [2].

### 2.2.2. Побудова діаграми використання системи

Діаграми використання описують функціональність та поведінку системи, що дозволяє замовнику, кінцевому користувачеві і розробнику спільно обговорювати проектовану або існуючу систему [3].

При моделюванні системи за допомогою діаграм прецедентів проектувальник прагне:

- чітко відокремити систему від її оточення;
- визначити дійових осіб (акторів), їх взаємодію з системою і очікувану функціональність системи;
- визначити в глосарії предметної області понятття, які стосуються детального опису функціональності системи (тобто прецедентів);

Для відображення моделі прецедентів на діаграмі використовуються:

- актор (англ. actor) – стилізований чоловічок, що позначає набір ролей користувача (розуміється в широкому сенсі: людина, зовнішня сутність, клас, інша система), що взаємодіє з деякою сутністю (системою, підсистемою, класом). Актори не можуть бути пов'язані один з одним (за виятком відносин узагальнення / успадкування);
- прецедент – еліпс з написом, що позначає виконувані системою дії (можуть включати можливі варіанти), що призводять до спостережуваних акторами результатами. Напис може бути ім'ям або описом (з точки зору акторів) того, «що» робить система. Ім'я прецеденту пов'язано з неперервним (атомарним) сценарієм – конкретної послідовністю дій, що ілюструє поведінку. В ході сценарію актори обмінюються з системою повідомленнями. З одним прецедентом може бути пов'язано кілька різних сценаріїв.

Діаграми сценаріїв використання відображають відносини між акторами і прецедентами системи [4]. Прецедентом є можливість системи, за допомогою якої користувач потрібний йому результат.

Для опису діаграм сценаріїв виконання роботи системи, використані наступні типи акторів, які були описані в специфікації вимог до системи:

– Касир – працівник який працює за POS-терміналом;
– Працівник на станції – пряцює на станціях з приготування продукції (додатковий актор);
– Аналітик – актор, безпосередньо взаємодіє з системою;
– Менеджер – актор, який є керівником аналітика (додатковий актор);
– Системний адміністратор – актор, який є налаштовувачем системи (додатковий актор).

Побудована діаграма прецедентів представлено на рис.2.1.

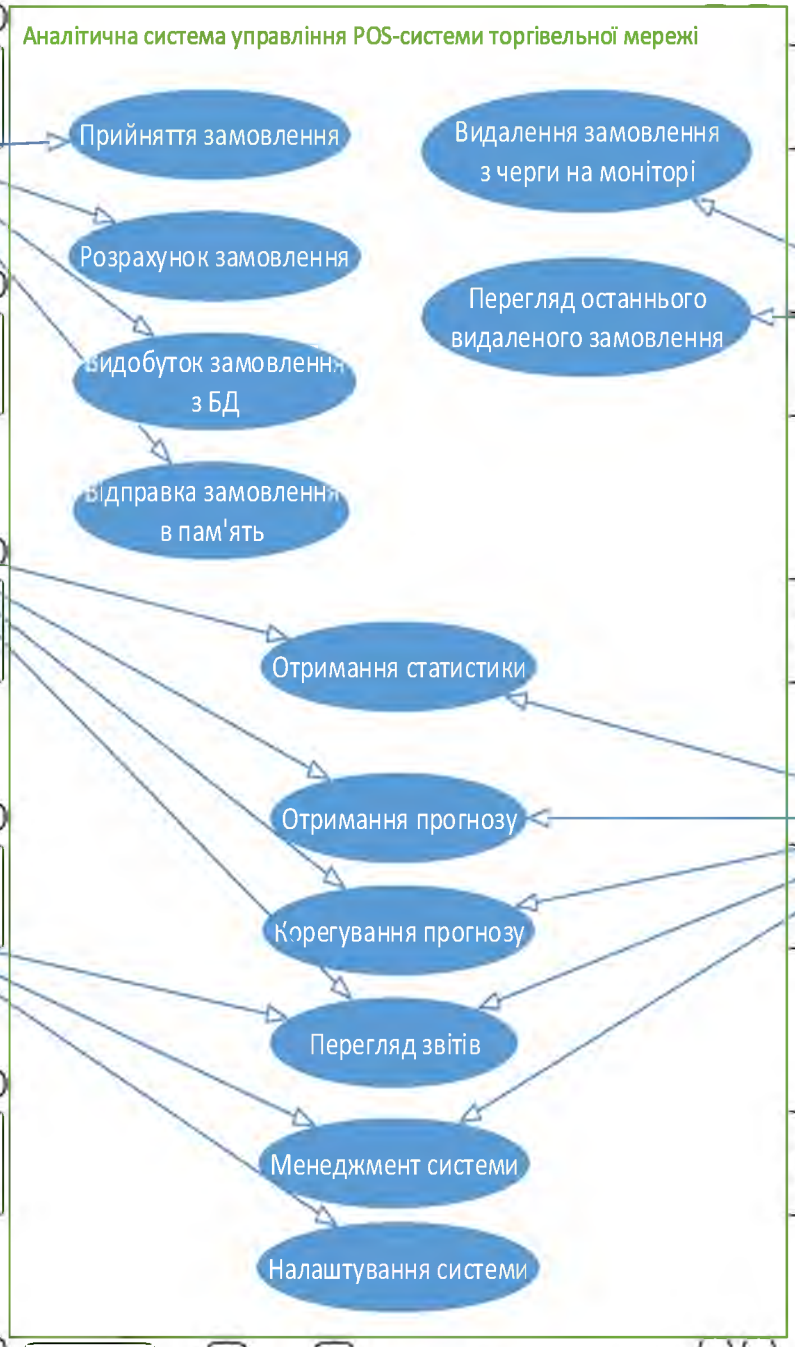


Рис. 2.1 Діаграма прецедентів

## 2.3. Проектування інтерфейсу користувача

«Є два способи створення дизайну програми. Один з них, це зробити його настільки простим, що в ньому, очевидно, не буде недоліків. Інший спосіб -

зробити його настільки заплутаним, що в ньому не буде очевидних недоліків.» - C.A. R. Hoare.

Графічний інтерфейс користувача (англ. Graphical user interface, GUI) – різновид призначеного для користувача інтерфейсу, в якому елементи інтерфейсу (меню, кнопки, значки, списки і т. п.), представлені користувачеві на дисплеї, виконані у вигляді графічних зображень.

Призначений для користувача інтерфейс повинен бути налаштований відповідно до мети. Як правило, перш за все потрібно зосередитися на графічному інтерфейсі функціональність, простота використання і естетика є вирішальними критеріями для хорошого користувальницького досвіду. При розробці графічного інтерфейсу не можна випускати з уваги фактор зручності використання: якщо додаток складно використовувати, привабливого дизайну буде недостатньо, для того щоб користувач продовжував використовувати програмний продукт. Дизайн інтерфейсу повинен підтримувати функціональність графічного інтерфейсу. Отже, його структура повинна бути ясною і добре організованою [5].

Добре розроблений графічний інтерфейс також робить позитивний вплив на користувача. Якщо користувачі зможуть користуватись інтерфейсом без проблем, то вони будуть добре і охоче витрачати час на його вивчення. Якщо користувачеві щось не подобається при першому відвідуванні, він відразу ж залишає додаток і починає шукати кращу альтернативу.

Додаток є багатовіконним, тому всі елементи будуть розташовані в різних вікнах. У головному меню будуть зображені (рис 2.2):

- Группа кнопок для формування звітів
- Коротенький DashBoard

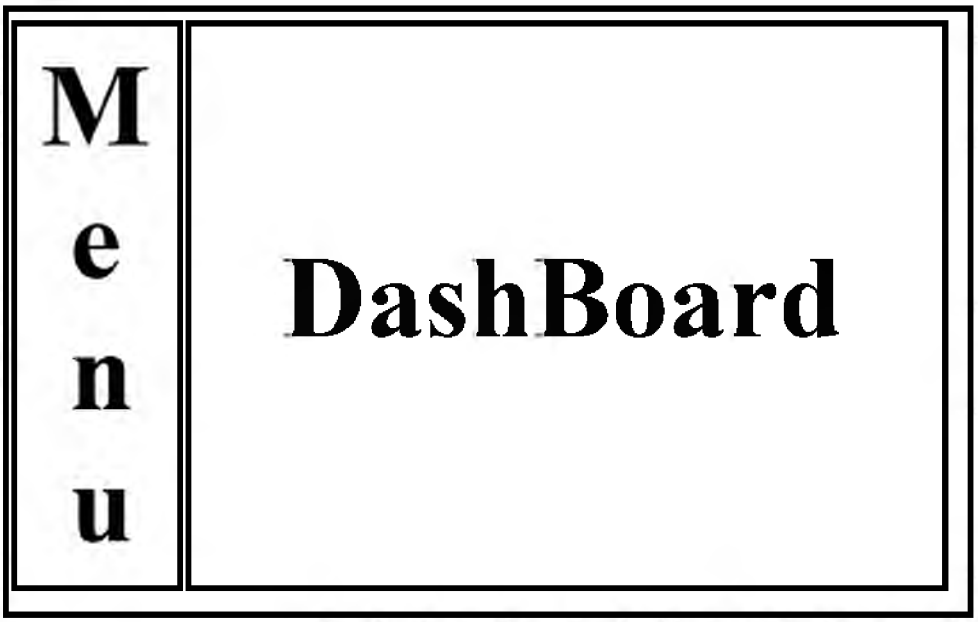


Рис. 2.2 Макет головного меню

Вікно звіту Hourly Sales Report (рис 2.3):

- Зона звіту
- Зона функціональний кнопок


Рис. 2.3 Макет вікна звіту

## 2.4. Проектування архітектури програмного забезпечення

### 2.4.1. Вступ.

В розділі специфікацій до вимог ПЗ (п. 2.1.3.) описано компоненти та зовнішні інтерфейси додатку, та вказано на застосування клієнто-направленої архітектури додатку. Частини системи, які описує рисунок 2.1 включає компоненти які підпадають під процес розробки та проектування архітектури, та ті що повинні застосовувати для реалізації функціональних вимог розробники шляхом взаємодії з ними через програмні інтерфейси які вони надають.

Компонент системи «Клієнт» підлягає процесу проектування та реалізації, всі інші компоненти вказані для того щоб відобразити зв'язок, який повинен бути передбаченим та реалізованим.

Отже, під проектуванням архітектури програмного забезпечення системи розуміється проектування саме цього компоненту та зв'язку з іншим.

В розділі специфікації вимог 2.1.2.2 була поставлена вимога до розробки інтерфейсу користувача у вигляді не одної сторінки. Тож на проектування архітектури ПЗ не було накладено певні обмеження.

Архітектура буде влаштована таким чином, що буде необхідності повторно завантажувати одні й ті ж елементи інтерфейсу. Основною перевагою такої розробки є:

– Легка розробка. Як правило для розробки багатосторінкового додатка потрібно менший стек технологій.
– Безліч рішень.

### 2.4.2. Вибір мови програмування.

«Є всього два типи мов програмування: ті, на які люди весь час лаються, і ті, які ніхто не використовує.» - Bjarne Stroustrup. Творець C++.

Для реалізації клієнтської частини необхідно обрати мову написання додатку, та за необхідністю середовище розробки.

На початковому етапі створення програмного продукту так чи інакше постає питання вибору мови програмування. Деякі розробники обирають мову тільки з особистих переваг а хтось навіть не замислюється. Однак, даних етап розробки є важливим, так як від нього в майбутньому визначається дуже багато критеріїв, наприклад – швидкість створення програми, швидкість тестування, можливість переносу на інші платформи, можливості внесення змін, швидкість роботи і так далі. При всьому цьому варто пам'ятати, що ідеальної мови програмування не існує, всі вони мають певні позитивні і негативні якості, які будуть так чи інакше впливати на процес розробки.

Отже, були встановлені основні критерії для вибору мови програмування:

– Швидкість роботи кінцевого продукту:

Вимогливими до швидкості виконання можуть бути програми з великим обсягом математичних обчислень, наприклад моделювання фізичних систем. Для даних цілей добрим прикладом будуть компільовані мови, такі як: Асемблер, С та С++, фортран і так далі. Саме ці мови після збірки не використовують нічого зайвого і містять у собі всі машинні команди які виконуються без затримок. Окрім цього одна зі схем роботи з даною мовою розробки має на увазі роботу з додатковими бібліотеками.

– Обсяг займаної оперативної пам'яті:

Дана вимога з'являється, коли програма розробляється наприклад, для мобільних платформ, вбудованих систем або ж це має бути невеликий віджет. В таких випадках, чим менше пам'яті витрачає додаток на надій мові – тим краще. До таких мов, можна віднести асемблер, С та С++, Objective-C та інші. Так як чим менше функціональних блоків в схемі виконання, тим менше займається і пам'яті комп'ютера. Якщо дана вимога некритична, то можна використовувати «істинно високорівневі мови» програмування.

– Швидкість розробки програми:

Дана вимога виникає тоді коли в цьому є необхідність. Через це вибір припадає на високорівневі мови. Це такі, як Java, Flash та подібні (рис.2.4). На даних мовах час розробки може істотно скорочуватися через велику кількість сторонніх бібліотек, і максимально простого синтаксису та подібних речей. Швидкість виконання програм, написаних на даних мовах страждає, причому часом досить відчутно. Отже, така розробка часто має на увазі повільне виконання додатку.


Рис. 2.4 Найпростіші для розробки мови програмування

– Наявність GUI

При наявності інтерфейсу користувача програма повинна володіти потужною графічною частиною, що відповідає вимогам дизайну. Розробка графічної частини часто вимагає досить багато часу, тому що відрізняється чималою складністю. Складність виникає в тому, що висновок графіки – це чимало математики, а отже, присутня вимогливість до швидкості виконання, а через складність розробки присутня необхідність у високорівневій мові. В даному випадку доцільним буде використання C++ та C# з їх одночасною високорівневістю та швидкістю виконання програм на них. Однак якщо графічна частина не дуже складна, але досить красива, можливе використання Java і Flash, на яких створення красивого інтерфейсу набагато простіше, ніж на C++. Якщо ж програма не орієнтована на користувача, тоді вибір прилягає на асемблер та C.

– Кросплатформеність

Кросплатформеність – це можливість роботи програми на різних платформах, в різних операційних системах з мінімальними змінами, або ж зовсім без них. У цій сфері можна виділити такі мови: Java, C#, Flash, C++ з різними бібліотеками та інші, менш популярні мови. Тут можна відзначити декілька мов, та їх принцип кросплатформеності:

– Java. Мова програмування створювалась з такою умовою, що програми на даній мові повинні працювати на будь-якій платформі де є JVM (англ. Java Virtual Machine – віртуальна машина Java). Програми на Java взагалі не вимагають ніяких змін для переносу на іншу платформу.

– C++. На чистому C++ написати кросплатформенну програму досить важко, у коді виникає велика надмірність, втрачається швидкість виконання. Полегшують завдання кросплатформенні бібліотеки, як наприклад Qt (рис.2.5), які дозволяють домогтися принципу «один код на всі платформи», однак на кожну платформу потрібно програму збирати окремо.

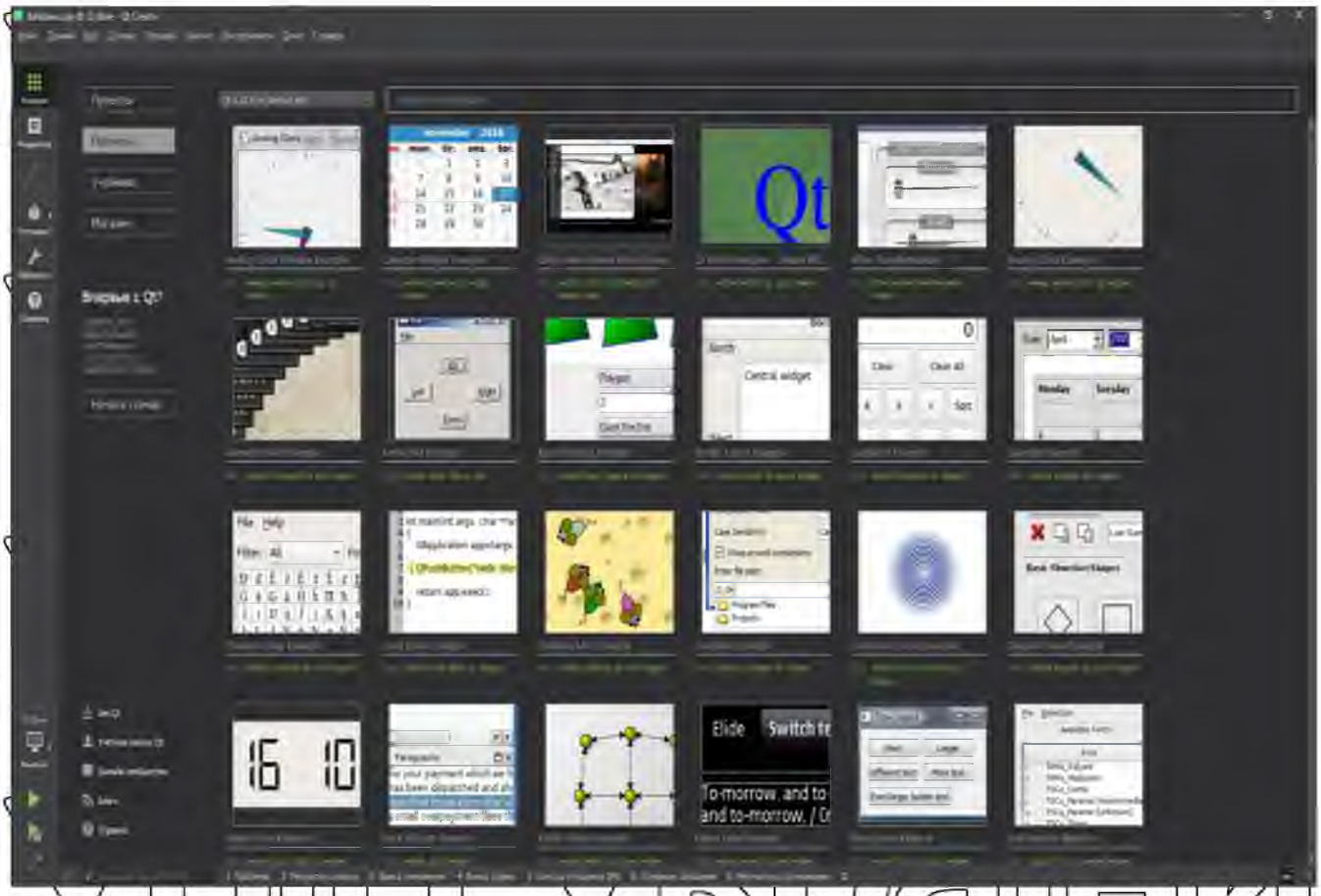

Рис. 2.5 Вікно Qt Creator

– Популярність

Так як у не дуже популярних мов програмування досить невелике ком'юніті це викликає деякі складноші коли виникає потреба вирішити проблему і звернутися за допомогою до «колег по цеху». Зазвичай чим популярніша мова програмування – тим більше по нім матеріалів, книжок, статей тощо. Якщо ж мова не дуже популярна то вона нажаль не має всього вище описаного.

Існує багато списків в порівняння популярності мов програмування, однак одним з найбільш дощільним вважається сайт TIOBE, за допомогою якого завжди можна побачити яка мова як розвивається. Даних графік складено за допомогою рейтингу з сайту TIOBE на початок листопаду (рис.2.6).

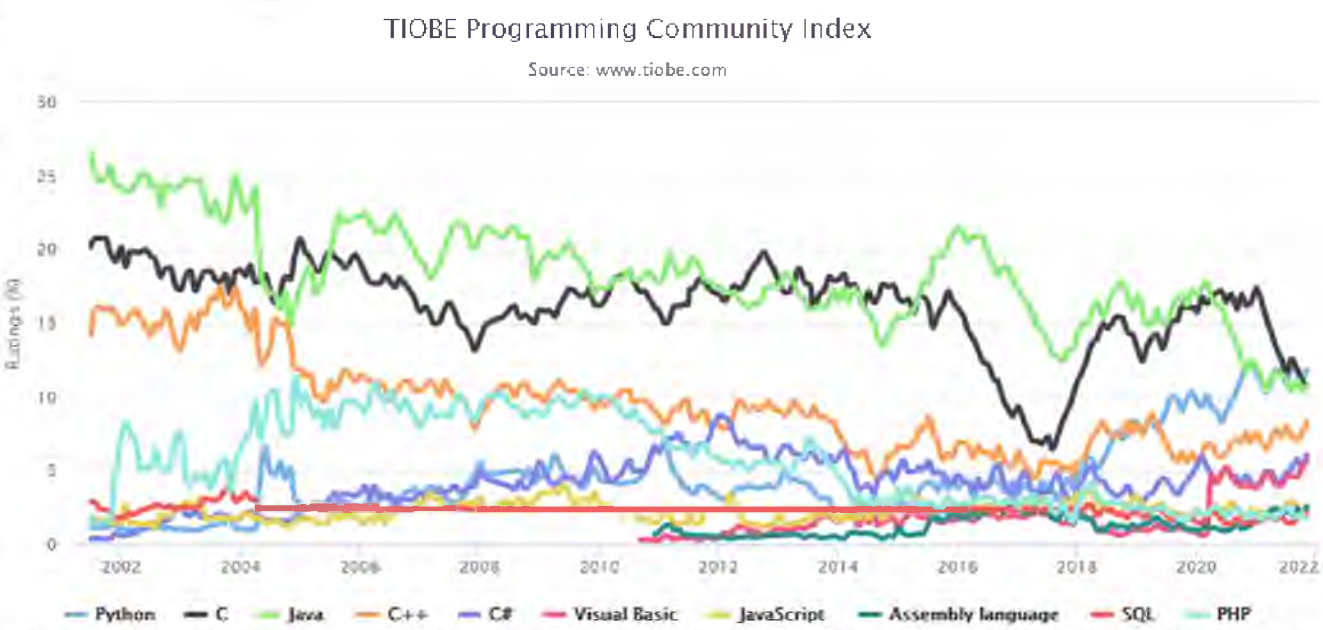


Рис. 2.6 Рейтинг популярності мов програмування по рейтингу сайту TIOBE

Отже, зважившись всі вимоги і критерії програмного продукту була обрана мова програмування C++[6] через те, що швидкість, обсяг займаної оперативної пам'яті та наявність інтерфейсу користувача є на першому місці, також гарною ознакою для додатку може стати орієнтованість в майбутньому на кросплатформеність. Ще Qt має велике ком'юніті, багато навчальних матеріалів прямо у них на сайті. Сам Qt/C++ досить легко вчиться, тому що в ньому є все що і в таких конкурентів як C# або Java, тільки набагато простіше в кросплатформеність.

Отже в якості середовища розробки було обрано найкращий варіант для Qt/C++, а саме середовище розробки з підтримкою бібліотек Qt, Qt Creator [7].

### 2.4.3. Розробка діаграми класів

Діаграма класів — статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення. Діаграма класів, також, може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших

типах діаграм. Діаграма класів служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування.

На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини.

В UML існують наступні типи зв'язків:

Асоціація показує, що об'єкти однієї сутності (класу) пов'язані з об'єктами іншої сутності.

Якщо між двома класами визначена асоціація, то можна переміщатися від об'єктів одного класу до об'єктів іншого. Цілком припустимі випадки, коли обидва кінці асоціації відносяться до одного і того ж класу. Це означає, що з об'єктом деякого класу дозволено зв'язати інші об'єкти з того ж класу. Асоціація, що зв'язує два класи, називається бінарною. Можна, хоча це рідко буває необхідним, створювати асоціації, що зв'язують відразу кілька класів; вони називаються n-арними. Графічно асоціація зображується у вигляді лінії, що з'єднує клас сам з собою або з іншими класами.

Асоціації може бути присвоєно ім'я, яке описує природу відносини. Зазвичай ім'я асоціації не вказується, якщо тільки ви не хочете явно задати для неї рольові імена або у вашій моделі настільки багато асоціацій, що виникає необхідність посилатися на них і відрізняти один від одного. Ім'я буде особливо корисним, якщо між одними і тими ж класами існує кілька різних асоціацій.

Клас, що бере участь в асоціації, грає в ній деяку роль. По суті, це "обличчя", яким клас, що знаходиться на одній стороні асоціації, звернений до класу з іншого її боку. Ви можете явно позначити роль, яку клас грає в асоціації.

Часто при моделюванні буває важливо вказати, скільки об'єктів може бути пов'язано допомогою одного примірника асоціації. Це число називається кратністю (англ. Multiplicity) ролі асоціації та записується або як вираз, значенням якого є діапазон значень, або в явному вигляді. Вказуючи кратність на одному кінці асоціації, ви тим самим говорите, що на цьому кінці саме стільки об'єктів повинно

відповідати кожному об'єкту на протилежному кінці. Кратність можна задати рівною одиниці (1), можна вказати діапазон: "нуль або одиниця" (0..1), "багато" (0 .. *), "одиниця або більше" (1 .. *). Дозволяється також вказувати певне число (наприклад, 3). За допомогою списку можна задати і більш складні кратності, наприклад 0. . 1, 3..4, 6 .. *, що означає "будь-яке число об'єктів, крім 2 і 5".

Діаграма класів побудована для розроблюваної системи предетавлена на рис.2.7.

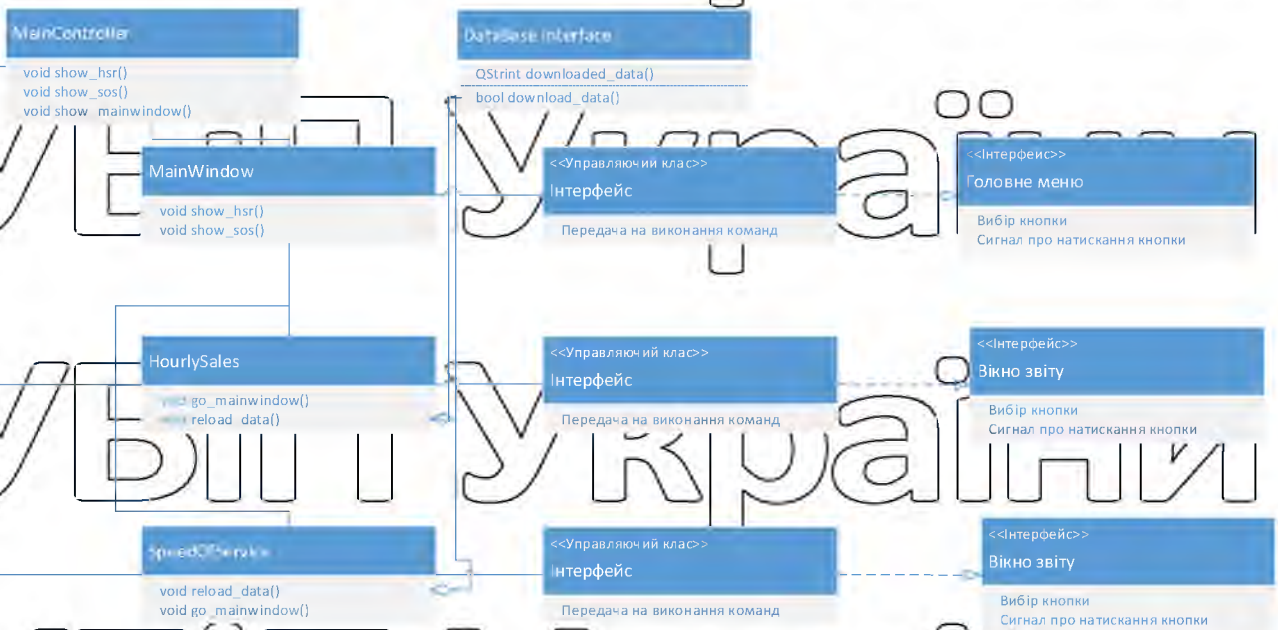


Рис. 2.7 Діаграма класів

# 3. РЕАЛІЗАЦІЯ СИСТЕМИ

«За допомогою C ви легко можете вистрілити собі в ногу. За допомогою C++ це зробити складніше, але якщо це станеться, вам відірве всю ногу цілком»

– Bjarne Stroustrup. Творець C++.

## 3.1. Вступ

Як описано вище, мовою програмування клієнтського застосунку була обрана C++ з використанням інструментарію Qt, який в першу чергу розроблявся як бібліотека для створення графічного інтерфейсу на мові C++. В наш час Qt значно переріс рамки звичайної бібліотеки, зараз це повноцінний framework, що дозволяє використовувати при написанні більшої частини додатку використовувати тільки «рідні» класи Qt и майже повністю відмовитися від написання системно-залежного коду, використання системних викликів або від виготовлення власних кросплатформенних обгорток. Класи Qt покривають майже всі необхідності програміста. В Qt передбачені класи для роботи з файлами, сіткою, базами даних а також для забезпечення багато поточного виконання [8].

Розробку почнемо з інтерфейсу і будемо йти по окремим функціям, але перед цим розглянемо діаграму діяльності на основі якої буде розроблюватися інтерфейс та показана взаємодія класів.

В діаграмі діяльності (рис.3.1) описано послідовність дій які викоиує програма коли взаємодіє з користувачем. Послідовність дій складається з завантаження даних по запиту користувача, обробка даних одразу після завантаження.

Після завантаження і обробки користувач повинен обрати звіт, на що додаток виведе вікно зі звітом, завантажить його і буде очікувати наступну дію від користувача. Що є досить легко та зрозуміло. Деталі розробки інтерфейсу будуть

описані в головному класі, а також у класах з вікнами звітів, до яких і належить інтерфейс.

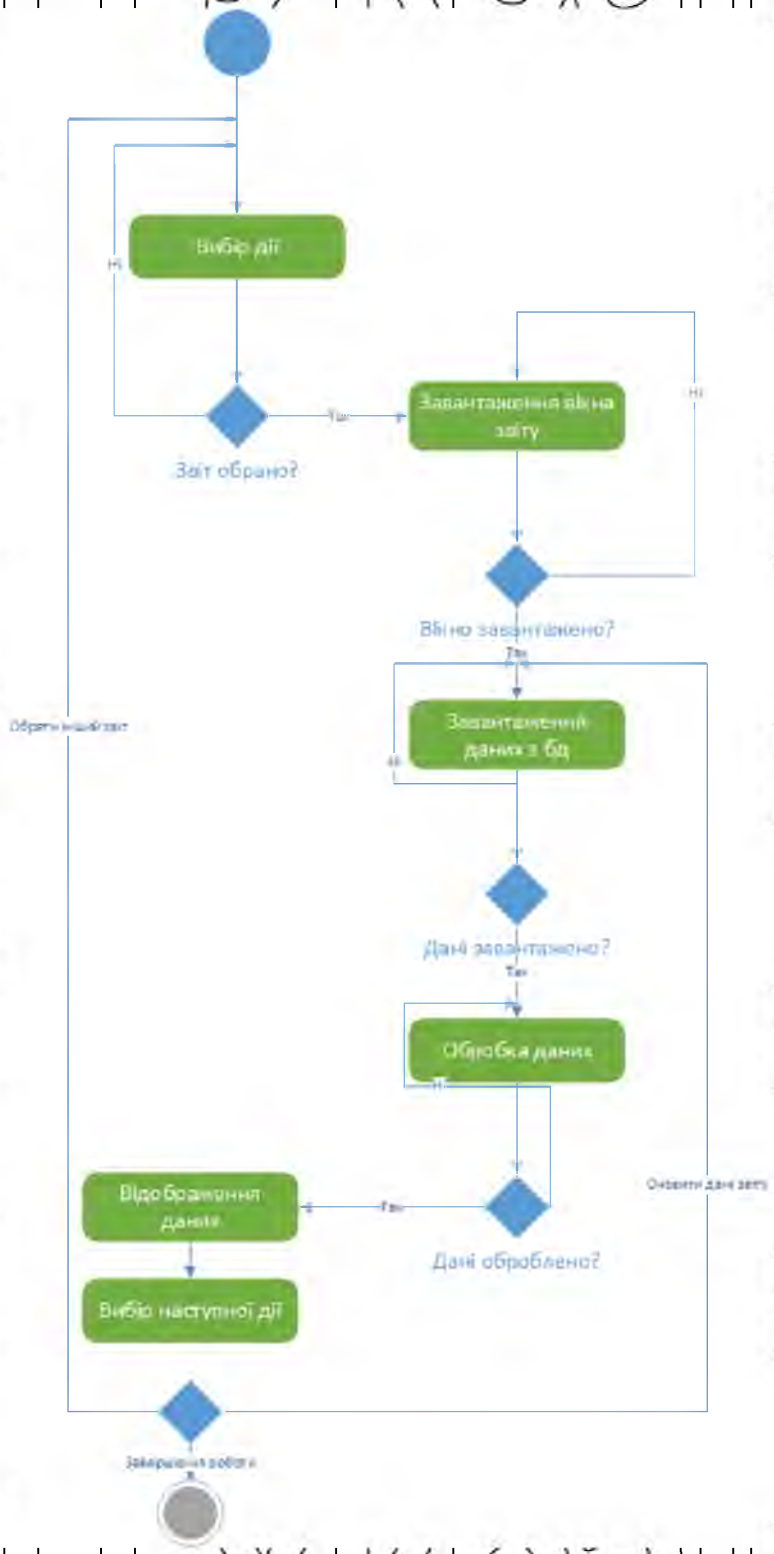


Рис. 3.1 Діаграма діяльності

## 3.2. Розробка POS-системи та бази даних

POS-система та база даних є невід'ємною частиною BackOffice системи для аналізу даних по продажам. Для тестів системи був розроблений додаток під кодовою назвою Q-POS, а також розвернута база даних MS SQL на хостингу.

Додаток Q-POS (рис.3.2) представляє собою просту POS-систему орієнтовану на мережу фаст-фуду, всі необхідні дані про замовлення направляються до заготовлених таблиць в базі даних.

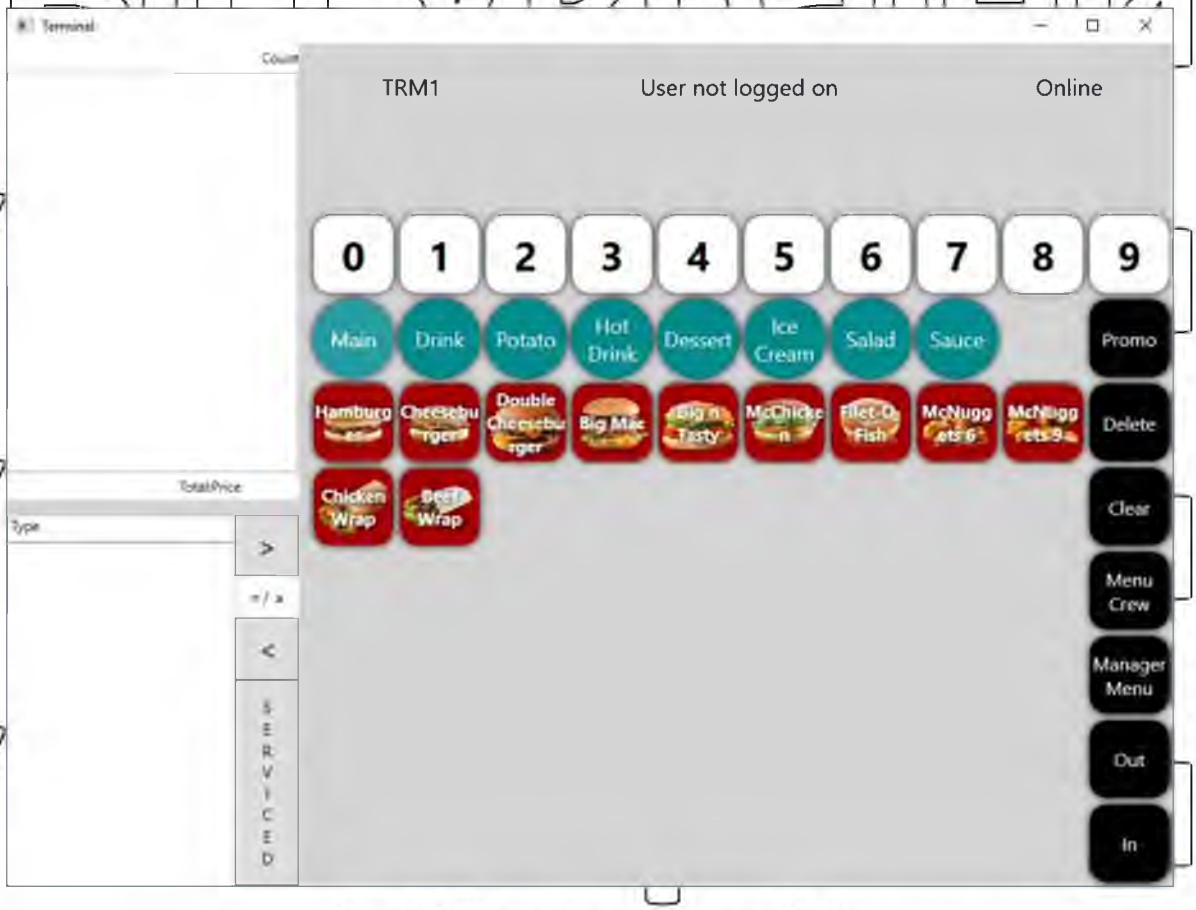


Рис. 3.2 Вигляд додатку для POS

База даних – сховище, куди додаток складає свої дані. Якщо додаток невеликий, окрема база не потрібна. Але потім це стає зручніше і вигідніше з точки зору пам'яті.

Місце в пам'яті обмежена. Тому коли даних багато, їх потрібно кудись скласти. Можна писати в файлики, а можна зберігати інформацію в базу даних (скорочено БД).

В даний час найбільш відомими СУБД є: Oracle Database, MS SQL Server, MySQL (MariaDB) і ACCESS. Остання входить до складу професійного офісного пакету Microsoft Office.

Це сучасні системи з великими можливостями, призначені для розробки складних програмних комплексів, і знайомство з ними для користувача ЕОМ виключно корисно.

База даних на даному етапі була створена власноруч за допомогою процедурного розширення мови SQL, а саме T-SQL[9] (англ Transact-SQL - процедурне розширення мови SQL, створене компанією Microsoft і Sybase).

Мова Transact-SQL є ключем до використання MS SQL Server. Усі програми, які взаємодіють з екземпляром MS SQL Server, незалежно від їх реалізації та інтерфейсу користувача, надсилають серверу інструкції Transact-SQL.[10]

У Transact-SQL існують спеціальні команди, які дозволяють управляти потоком виконання сценарію, перериваючи його або направляючи в потрібну гілку.

– Блок угруповання-структура, що об'єднує список виразів в один логічний блок (BEGIN ... END).

– Блок умови-структура, що перевіряє виконання певної умови (IF ... ELSE).

– Блок циклу-структура, що організує повторення виконання логічного блоку (WHILE ... BREAK ... CONTINUE).

– Перехід-команда, що виконує перехід потоку виконання сценарію на зазначену мітку (GOTO).

– Затримка-команда, що затримує виконання сценарію (WAITFOR).

– Виклик помилки-команда, що генерує помилку виконання сценарію (RAISERROR).

Створення таблиці для Hourly Sales Report представлено на рис. 3.3.

```
use [***]
create table hourlysales (
  ID INT NOT NULL IDENTITY(1,1) PRIMARY KEY,
  _DATE DATE NOT NULL,
  _TIMESTAMP TIME NOT NULL,
  _ZONE CHAR(3) NOT NULL
);
```

Рис. 3.3 Створення таблиці для Hourly Sales Report

Створення таблиці для Speed Of Service представлено на рис. 3.4.

```
CREATE TABLE speedofservice (
  ID INT NOT NULL IDENTITY(1,1) PRIMARY KEY,
  _DATE DATE NOT NULL,
  _TIMESTAMP_ORDER_DURATION TIME NOT NULL,
  _TIMESTAMP_CASH_DURATION TIME NOT NULL,
  _TIMESTAMP_COOK_DURATION TIME NOT NULL,
  _TIMESTAMP_PRESENT_DURATION TIME NOT NULL,
  _ZONE CHAR(3) NOT NULL
);
```

Рис. 3.4 Створення таблиці для Speed Of Service

За допомогою POS-системи було значно спрощено формування звіту, так як можна налаштувати систему таким чином, щоб вона сама рахування довжину кожного етапу взаємодії з гостем і вносила дані в базу даних уже дещо підготовлені.

## 3.3. Розробка класу контролера

Клас контролер – це клас який контролює інші класи, а також створює зв’язки між ними.

В першу чергу при розробці будь-якого класу потрібно ініціалізувати бібліотеки які будуть використовуватися в даному класі, деякі бібліотеки будуть

ініціалізовані в інших класах які після будуть підключені за допомогою ініціалізації цих класів в контролері.

Об’ява бібліотек представлено на рис.3.5.

```
#include <QMainWindow>
#include "mainwindow.h"
#include "hourlysalesreport.h"
#include "speedofservicereport.h"
#include "database.h"
#include <QMessageBox>
#include <QApplication>
#include <QSql>
```

Рис. 3.5 Об’ява бібліотек

Коротко про бібліотеки:

- QMainWindow та QWidget необхідні бібліотеки для реалізації інтерфейсу;
- Клас QMessageBox надає модальний діалог для інформування користувача або для того, щоб задати користувачеві питання і отримати відповідь;
- Клас QApplication управляє потоком управління графічним інтерфейсом програми і основними настройками.
- Qt SQL є важливим модулем, який забезпечує підтримку баз даних SQL.
- mainwindow.h – написаний власноруч клас, який використовується для відображення головного меню.
- hourlysalesreport.h – написаний власноруч клас, який використовується для відображення вікна з однойменним звітом
- speedofservicereport.h – написаний власноруч клас, який використовується для відображення вікна з однойменним звітом
- database.h – написаний власноруч клас, який використовується для створення запитів до бази даних, а також отримання відповідей з неї

Головним завданням класу контролера є обробка сигналів від різних класів та їх перенаправлення на потрібні класи, або ж виконання необхідних дій.

Для обробки сигналів використовується стандартна функція connect за допомогою якої можна зв'язати сигнал зі слотом [11], яка має наступний синтаксис

Синтаксис створення зв'язку сигнал-слот представлено на рис. 3.6.

```
connect(const QObject *sender, const char *signal, const QObject *receiver, const char *member)
```

Рис. 3.6 Синтаксис створення зв'язку сигнал-слот

Якщо спростити то, функція обробляє в режимі реального часу вказаних сигнал певного об'єкту, після цього викликає слот іншого об'єкта, таким чином можливо одноразово, на весь час виконання класу, підв'язати сигнал до слоту [12]. Схема зв'язку сигналу зі слотом зображено на рис.3.7.

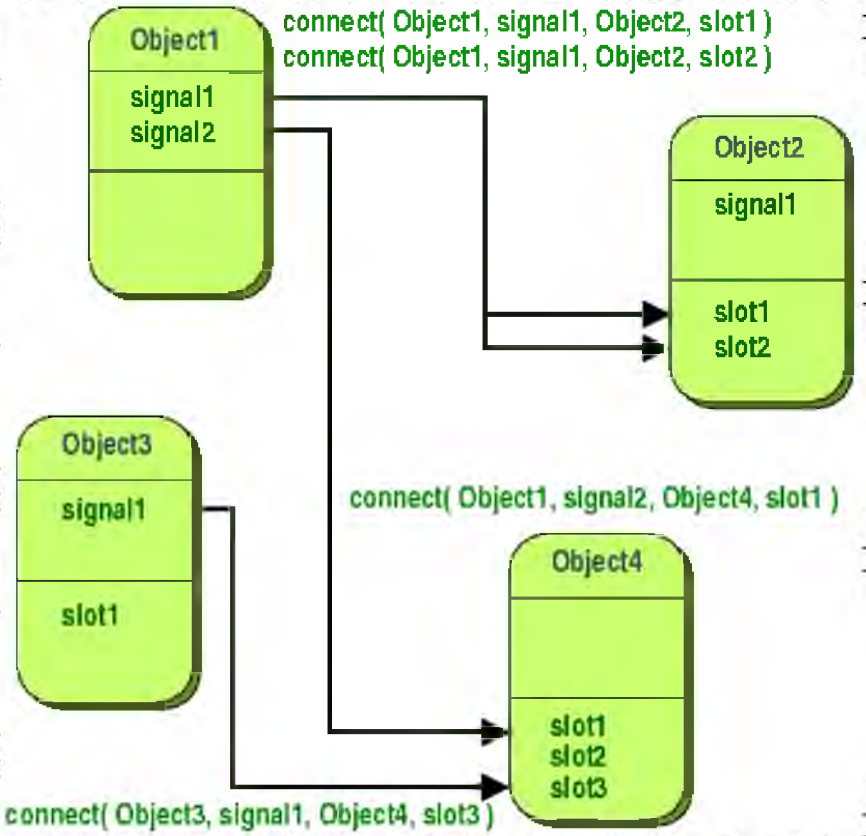


Рис. 3.7 Схема зв'язку сигналу зі слотом

Конструктор класу дуже важлива річ, тому що, саме в конструкторі проводяться дії які необхідно виконати при ініціалізації.

Конструктор класу представлено на рис.3.8.

```
MainController::MainController(QWidget *parent)
    : QMainWindow(parent)
{
    connect(&ui_HSR, &HourlySalesReport::NeedOpenMainWindow, this,
&MainController::ReturnToMainWindow);
    connect(&ui_HSR,       &HourlySalesReport::NeedAbout,       this,
&MainController::About);
    connect(&ui_HSR,       &HourlySalesReport::NeedSelect,      this,
&MainController::SelectHSR);
    connect(&ui_SOS,      &SpeedOfServiceReport::NeedOpenMainWindow,
this, &MainController::ReturnToMainWindow);
    connect(&ui_SOS,      &SpeedOfServiceReport::NeedAbout,     this,
&MainController::About);
    connect(&ui_SOS,     &SpeedOfServiceReport::NeedSelect,     this,
&MainController::SelectSOS);
    connect(&ui_MainWindow,      &MainWindow::NeedOpenHSL,      this,
&MainController::Open_HSR);
    connect(&ui_MainWindow,      &MainWindow::NeedOpenSOS,      this,
&MainController::Open_SOS);
    connect(&ui_MainWindow,      &MainWindow::NeedAbout,      this,
&MainController::About);
}
```

Рис.3.8 Конструктор класу

В конструкторі класу використовується «конекти» всіх необхідних сигналів.

Таким чином контролер має доступ до всіх необхідних функцій інших класів та свого.

З самих необхідних функцій в контролері є декілька:

– перехід між вікнами реалізовано за досить легким принципом, знаходиться активне вікно яке необхідно закрити, закривається і відкривається вікно яке запитав користувач

Функція повернення до головного меню представлена на рис.3.9.

```
void MainController::ReturnToMainWindow()
{
QWidget *activeWindow = QApplication::activeWindow();
                activeWindow->close()
  ui_MainWindow.show();
}
```

Рис.3.9 Функція повернення до головного меню

– перенаправлення запиту з викликаючого класу звіту до класу зв’язку з базою даних

Функція перенаправлення представлена на рис.3.10.

```
void MainController::SelectHSR()
{
    ui_HSR.answer          =          ui_db.DownloadFromDB("HSR",
ui_HSR.selected_date);
    emit ui_HSR.SelectIsDone();
}
```

Рис.3.10 Функція перенаправлення

Таким чином в даному класі були описані всі потрібні для нього функції.

## 3.4. Розробка класу звіту Hourly Sales Report

По аналогії з головним класом, спочатку проініціалізуємо необхідні бібліотеки (рис.3.11).

```
#include <QWidget>
#include <QShortcut>
#include <QDate>
#include <QtSql/QSqlQuery>
#include <QTableWidget>
```

Рис. 3.11 Ініціалізація бібліотек

Коротко про бібліотеки:

– QShortcut – бібліотека котра необхідна для реалізації та призначення гарячих клавіш

– QDate – призначена для виконання дій з датами

– QTableWidget – необхідна для роботи з таблицями

Після підключення бібліотек розроблюємо конструктор класу (рис.3.12).

```
HourlySalesReport::HourlySalesReport(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::HourlySalesReport)
{
    ui->setupUi(this);
    keyF1 = new QShortcut(this);
    keyF1->setKey(Qt::Key_F1);
    connect(this->keyF1, &QShortcut::activated, this, [=](){ emit
NeedAbout(); });
    keyESC = new QShortcut(this);
    keyESC->setKey(Qt::Key_Escape);
    connect(this->keyESC, &QShortcut::activated, this, [=](){ emit
NeedOpenMainWindow(); });
    connect(ui->GoBack_pushButton, &QPushButton::clicked, this,
[=](){ emit NeedOpenMainWindow(); });
    connect(this, &HourlySalesReport::SelectIsDone, this,
&HourlySalesReport::DrawInfo);
    connect(ui->Refresh_pushButton, &QPushButton::clicked, this,
[=](){ emit NeedSelect(); });
    ui->dateEdit->setDate(QDate::currentDate());
}
```

Рис. 3.12 Конструктор класу

В конструкторі описано створення гарячої клавіші ESC для повернення в головне меню, підключення клавіш до функцій, а також встановлення сьогоднішньої дати до поля вибору дати в звіті.

Головною функцією даного класу є форматування та відображення даних отриманих від бази даних.

Опис функції DrawInfo представлено на рис.3.13.

```
void HourlySalesReport::DrawInfo()
{
    QStringList Result[3];
    while (answer.next()) {…}
    int hours[24];
    for (int i = 0; i <24 ; i++ ) {hours[i] = 0;}
    for (int i = 0; i < Result[0].size(); i++ ) {
        qDebug() << Result[1].at(i).left(2);
        switch (Result[1].at(i).left(2).toInt()) {…}}
    ui->tableWidget->setRowCount(24);
    for(int i = 0; i < 24; i++){
        int project = 0;
        for (int x = 0; x < 4 ; x++ ) { //0 - 2
            QTableWidgetItem* item = new QTableWidgetItem;
            if (x == 0){
                item->setText(QString::number(i) + ":00");}
            else if (x == 1){…}
            else if (x == 2){…}
            else if (x == 3){…}
            item->setTextAlignment(Qt::AlignCenter);
            ui->tableWidget->setItem(i, x, item);}}}
```

Рис.3.13 Опис функції DrawInfo

В даній функції описується розбір отриманих даних по списку QStringList, який по своїй суті є двовимірним масивом з більш зручними елементами взаємодії.

Після розбору відповіді від серверу проходить підрахунок транзакцій за кожну годину, а вже після цього проходить додаток результатів до таблиці.

Перейдемо до створення графічного інтерфейсу (рис. 3.14), Qt дозволяє досить легко створювати інтерфейс завдяки вбудованому конструктору в якому є досить багато елементів.

Рис. 3.14 Вигляд вбудованого редактора форм для Qt Creator

Після розробки склад інтерфейсу наступний:

– Таблиця з результатами, яка містить наступні стовпці:

- Time – вказаний проміжок часу;
- Project – запланована кількість транзакцій;
- Fact – фактична кількість транзакцій;
- Difference – різниця між планом та фактом;

– Функціональна зона:

- Поле вибору дати;
- Кнопка оновлення даних;
- Вихід до головного меню.

Всі елементи графічного інтерфейсу в Qt можуть використовувати CSS (мова таблиці стилів), що значно полегшує оформлення. Дуже часто оформлення обирається від побажань замовника, однак якщо додаток авторський, то автор часто використовує свою авторську таблицю стилів.

Отже, даний додаток має повністю авторську табличку стилів котра ініціалізована під час запуску додатку в main.cpp.

Фінальний вигляд вікна в конструкторі представлено на рис.3.15.

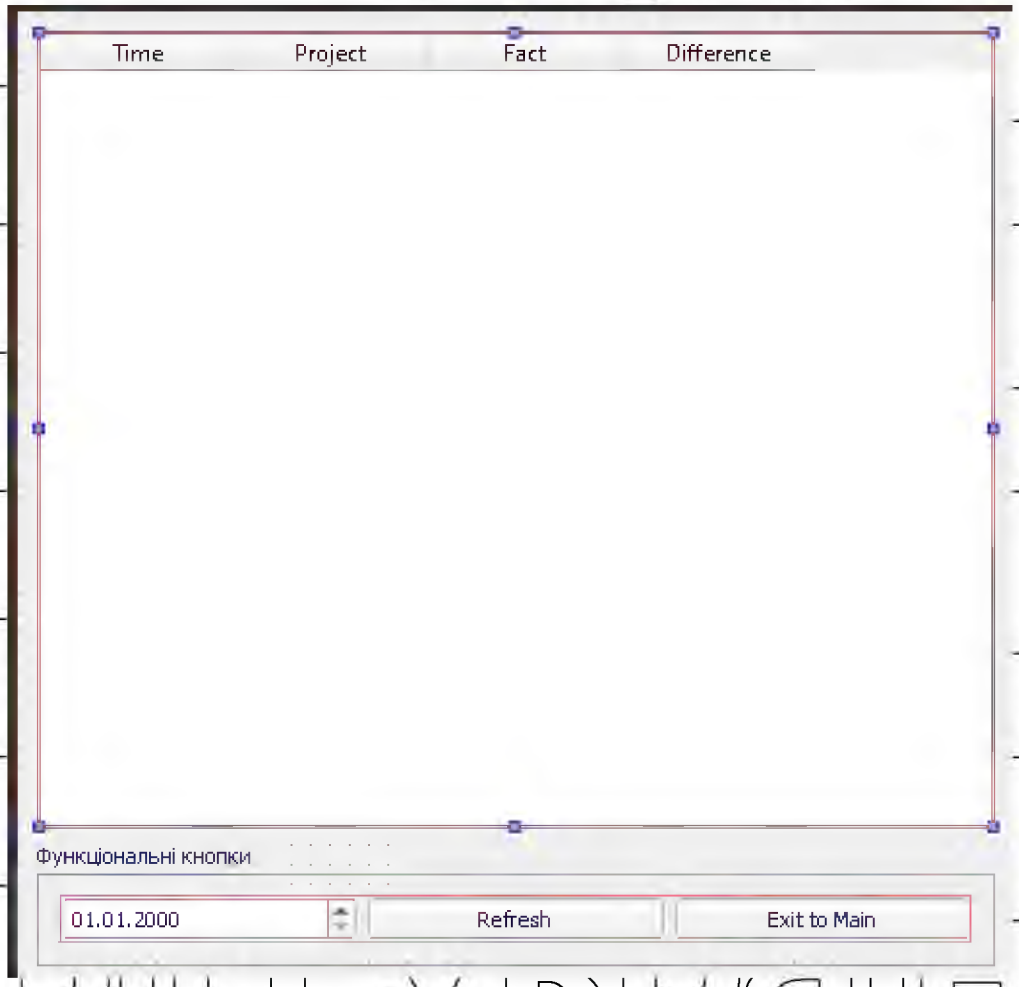


Рис. 3.15 Фінальний вигляд вікна в конструкторі

Так як, на момент створення дизайну вікна таблиця стилів ще не ініціалізовано, то в редакторі можна побачити не стилізований вигляд.

Фінальний вигляд вікна після завантаження представлено на рис.3.16.


Рис. 3.16 Фінальний вигляд вікна після завантаження

Таким чином в даному класі були описані всі потрібні для нього функції, а також етапи розробки дизайну додатку.

## 3.5. Розробка класу звіту Speed of Service

Розробка класу SpeedOfService проходить майже в повній аналогії з попереднім класом, однак з деякими уточненнями.

Але алгоритм обробки даних тут інший, за рахунок того, що під час формування даного звіту потрібно вилучити інформацію за цілий день — необхідно

вивести середні значення по всім критеріям оцінки ефективності обслуговування, а саме:

- Середній час прийняття замовлення
- Середній час оплати замовлення
- Середній час приготування замовлення
- Середній час представлення та передачі замовлення гостю

Отже, отримані дані діляться по зонам обслуговування, а далі вираховуються середні значення по часу обслуговування (рис.3.17).

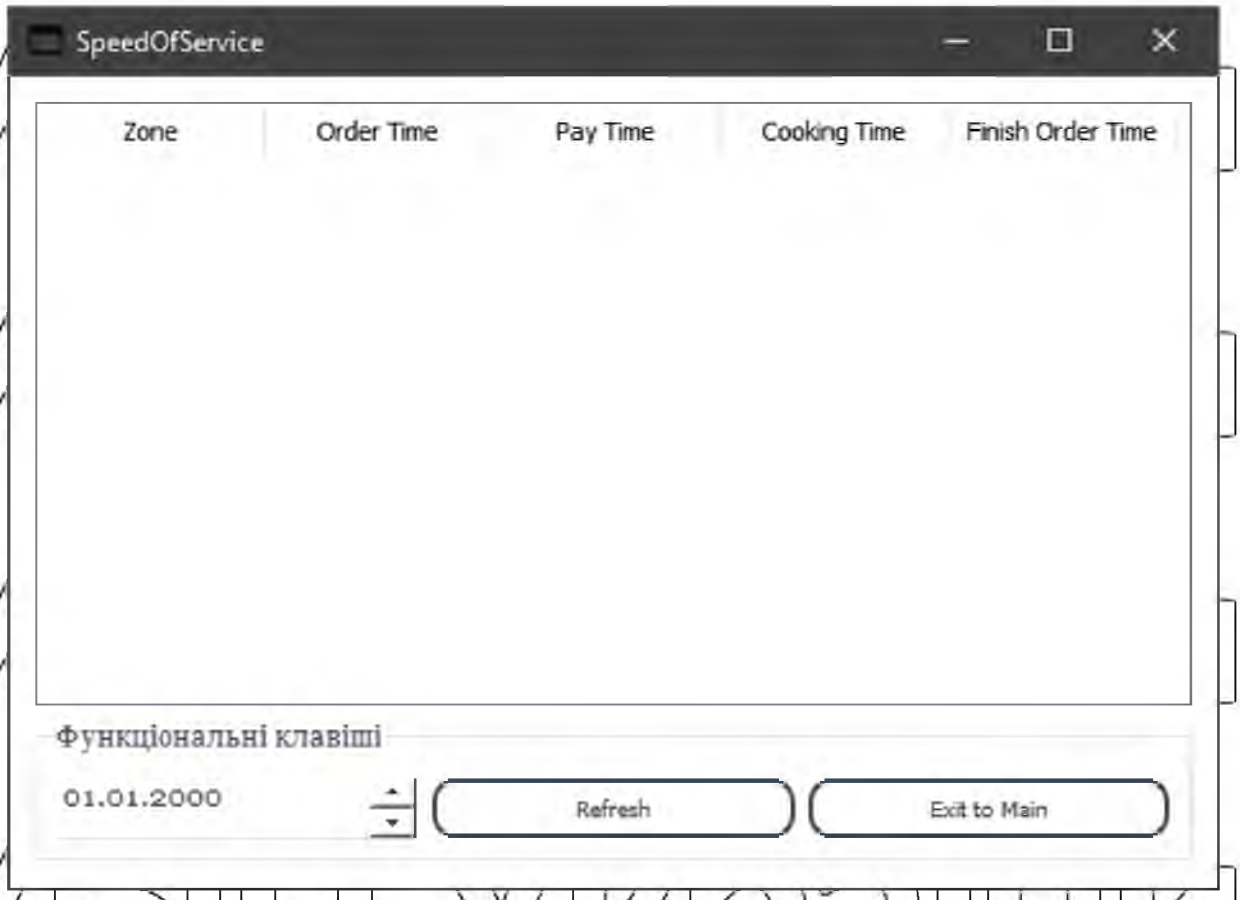


Рис. 3.17 Фінальний вигляд вікна після завантаження

## 3.6. Розробка класу головного меню

Клас головного меню не має багатьох функцій, за непотрібністю, головне що клас включає в себе інтерфейс користувача.

Конструктор класу представлено на рис.3.18.

```
MainWindow::MainWindow(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    keyF1 = new Qshortcut(this);
    keyF1->setKey(Qt::Key_F1);
    connect(this->keyF1, &Qshortcut::activated, this, [=](){
emit NeedAbout(); });
    connect(ui->Hourly_pushButton,      &QpushButton::clicked,
this, [=](){ emit NeedOpenHSL(); });
    connect(ui->SOS_pushButton,  &QpushButton::clicked,  this,
[=](){ emit NeedOpenSOS(); });
}
```

Рис.3.18 Конструктор класу

Головним в даному класі є переключення між вікнами, так як даний клас є так званим хабом. Вигляд головного меню представлено на рис.3.19.

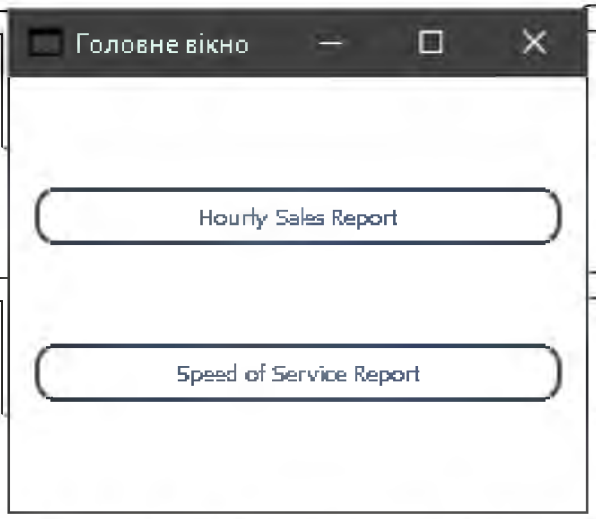


Рис.3.19 Вигляд головного меню

## 3.7. Розробка класу зв'язку з базою даних

У порівнянні з іншими функціональними класами – даний клас не має графічної оболонки через її непотрібність, але все що повинен робити даний клас

– це налагоджувати зв’язок з базою даних та надавати зворотні відповіді. Під’єднані бібліотеки представлені на рис.3.20.

```
#include <QSql>
#include <QSqlDriver>
#include <QtSql>
```

Рис. 3.20 Під’єднані бібліотеки

Даний клас (рис. 3.21)містить в собі декілька бібліотек для роботи з базами даних, їх драйверами та містить в собі всі необхідні типи змінних, що можуть знадобитися при роботі з даними.

На відміну від інших класів – даний клас в конструкторі не містить підключень до сигналів, натомість замість них тут проходить:

– перевірка драйверів і запис їх до списку;
– ініціалізація бази даних

```
DataBase::DataBase(QWidget *parent) :
    QMainWindow(parent)
{
    CheckDrivers();
    db = QSqlDatabase::addDatabase("QODBC", "MyConnect");
    db.setDatabaseName("DRIVER={SQL
Server};Server=sqlserver.admin.ua");
    db.setUserName("root");
    db.setPassword("root");
}
```

Рис. 3.21 Конструктор класу

Перевірка драйверів реалізована за допомогою окремою функції, так як не є доцільним використання великих сценаріїв в тілі конструктору.

Також в тілі цього класу реалізована функція повного логування всього що виконалось, або ж ні по якимось причинам.

Функція запису до логу представлена на рис.3.22.

```
void DataBase::WriteLog(QString ToLog)
{
    QFile proglog ("syslog.txt");
    proglog.open(QIODevice::Append);
    QString answerdb = CurrentDate.toString("yyyy.MM.dd") + " " +
ToLog + "\n";
    QByteArray ba = answerdb.toLocal8Bit();
    const char *c_str2 = ba.data();
    proglog.write(c_str2);
    proglog.flush();
    proglog.close();
}
```

Рис. 3.22 Функція запису до логу

Функція завантаження даних з бази представлена на рис.3.23.

```
QSqlQuery DataBase::DownloadFromDB(QString type, QDate date)
{
    bool db_error = !db.open();

    if (db_error == true)
        WriteLog("Non connect to DB: " + db.lastError().text());
    else
        WriteLog("Success connect");

    QSqlQuery *query = new QSqlQuery(db);
    QString table;
    if (type == "SOS")
        table = "speedofservice";
    else if (type == "HSR")
        table = "hourlysales";
    WriteLog("Query exec: " + QString::number(query->exec("use
[1gb_uclstat]; SELECT * FROM [" + table + "] WHERE _DATE = '" +
date.toString("yyyy-MM-dd") + "';")));
    WriteLog("LastError: " + query->lastError().text());
    WriteLog("Result: " + query->result()->handle().toString());
    return *query;
}
```

Рис. 3.23 Функція завантаження даних з бази

На вхід даної функції поступає тип шуканої інформації, а також дата за котру необхідно витягнути дані. На виході ж функція посилає посилання на об’єкт відповіді від серверу у сирому форматі, всі дані надалі будуть оброблятися в своєму класі за своїми сценаріями.

# 4. ТЕСТУВАННЯ РОЗРОБЛЕНОЇ СИСТЕМИ

«”Регресивне тестування”? Що це? Якщо система компілюється, то це добре, якщо завантажується, то це просто чудово!» - Лінус Торвальдс

## 4.1. Мануальне тестування

Основною ціллю тестування є доказ того, що продукт готовий до випуску, та усі заявлені розробником функції стабільно працюють. Тестування потрібно як самим розробникам, щоб упевнитися в готовності продукту, так і замовникам, щоб бачити за що вони заплатили. Якщо програмне забезпечення успішно проходить тестування, воно починає впроваджуватися, якщо ж ні – відправляється на доопрацювання. Найчастіше, тестування проводять окремі люди.

Мануальне тестування – це пряма взаємодія людини з додатком. У його процесі можна отримати зворотній зв'язок про продукт, що неможливо, якщо використовувати автоматизоване тестування.

Крім того, тестувальник зможе скласти свій відгук і рекомендації щодо поліпшення ПО, а проведене тестування буде свідченням порівняння очікуваного і отриманого результатів.

Ще одна особливість мануального тестування полягає у зворотному зв'язку щодо дизайну користувальницького інтерфейсу. Тільки реальна людина зможе звернути увагу на нюанси в колірних тонах або несиметричному розташуванні полів або кнопок.

Під час завантаження додатку висвітлюється SplashScreen (рис.4.1), котрий з’являється одразу при початку завантаження додатку та символізує завантаження додатку, що позитивно впливає на користувача.


Рис. 4.1 Splash Screen при завантаженні додатку

Після повного завантаження додатку відкривається головне меню (рис.4.2), яке має приємні анімації наводки на кнопки.

На цьому етапі від користувача не вимагається ніяких дій окрім вибору необхідного звіту, або ж читання опису додатку доступного по знайомій гарячій клавіші «F1».

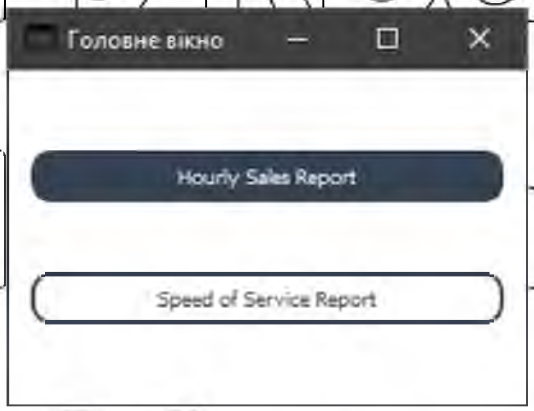


Рис. 4.2 Головне меню з активною підсвіткою обраної клавіші

Після вибору потрібного пункту меню – проходить коректне перемикання між вікнами програми, без якихось «глюків».

Гарячі клавіші інтуїтивно зрозумілі: для повернення назад завжди використовується клавіша ESC, хрестик вікна або ж окремо виведена клавіша; в кожному вікні можна викликати довідку за допомогою гарячої клавіші F1

Перейшовши у вікно звіту від користувача, потребується обрати дату за яку він хоче завантажити звіт, або ж просто оновити дані, якщо треба дізнатися актуальні (рис.4.3).


Рис. 4.3 Активне вікно зі звітом

Оновивши дані за допомогою однойменної кнопки користувач одразу через декілька секунд отримає очікуваний результат не дивлячись на те, що кількість записів за добу може досягати 10 тисяч. Завантажений звіт представлено на рис.4.4.


Рис. 4.4 Завантажений звіт

Під час проведення мануального тестування було виявлено, що додаток є user friendly (англ. зрозумілий, дружелюбний для користувача), і є інтуїтивно зрозумілий. Користувацька частина інтерфейсу не є перевантаженою незрозумілими функціями.

Під час проведення мануального тестування також була оброблена необхідна кількість ресурсів комп'ютера для функціонування даного програмного забезпечення.

Характеристики комп'ютера на якому проводилися тестування:

- Процесор: Intel Core i5-7300HQ;
- Оперативна память: 8 Гб;

- Пропускна спроможність WiFi модулю: 1 Гбіт/с

Кількість необхідного ресурсу комп'ютера:

- При фоновій роботі:
- Оперативна пам'ять: 11.2 МБ;
- Навантаження на центральний процесор та відео чіп: близько 0%;
- Навантаження на дискову систему: 0 МБ/с.
- При завантаженні даних:
- Оперативна пам'ять: 11.4 МБ;
- Навантаження на мережу: близько 0,1 Мбіт/с.
- Навантаження на центральний процесор та відео чіп: близько 2-4%;
- При активному перегляді даних:
- Навантаження на центральний процесор: максимальна помітка 5%;

Таким чином було визначено, що програма вийшла не вимогливою до апаратних ресурсів і може функціонувати на системах з слабким апаратним забезпеченням.

## 4.2. Можливості розвитку додатку

Після запуску питання розробки додатку не закривається раз і назавжди: вам доведеться стежити за ринком, підлаштовуватися під нього і вкладати гроші в подальший розвиток.

Мінімальне втручання-це запуск оновлень і усунення помилок: такі заходи дозволяють усунути недоліки, упущені при тестуванні, і підготувати додаток під актуальні вимоги ринку.

Також важливо займатися аналітикою: аналіз аудиторії та її дій допоможе не витрачати енергію даремно і вкладатися в розвиток тільки найбільш затребуваних функціональних можливостей.

Ось декілька прикладів розвитку додатку:

– Додавання більшої кількості звітів
– Додавання більш гнучких налаштувань звітів
– Додавання штучного інтелекту для аналізу даних та більш точного прогнозування
– Збільшення кількості тем для додатку
– Внесення до додатку повноцінної локалізації для різних країн світу, та їх гнучке налаштування
– Розробка консолідованого сховища даних для менеджменту і аналітики даних по всім торговим точкам

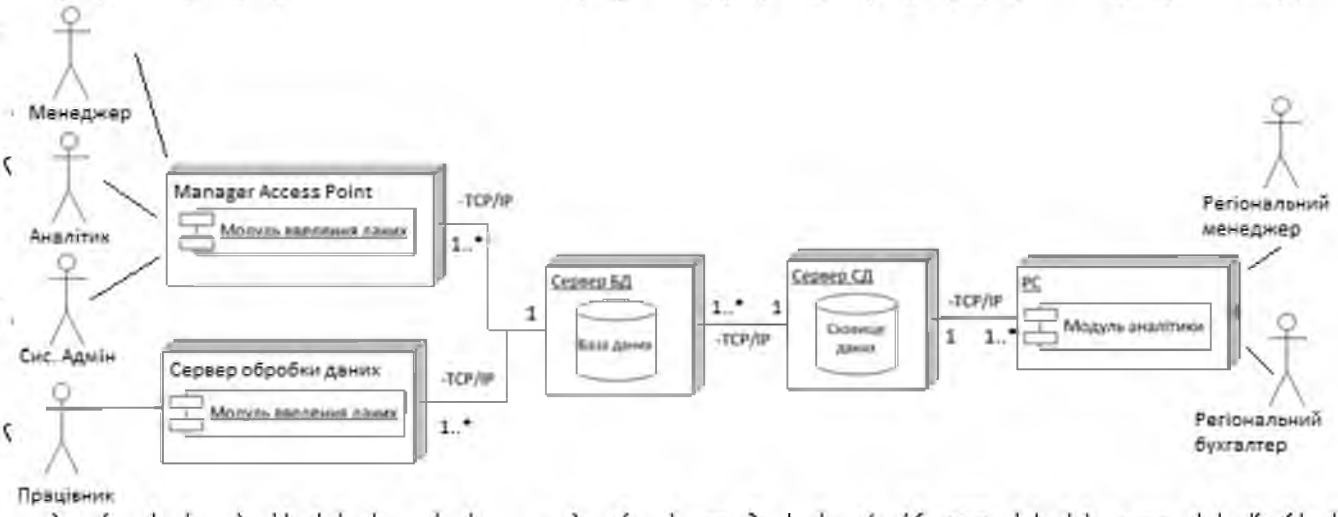


Рис. 4.5 Топологія системи зі сховищем даних

Кожен приклад так чи інакше впливає на кінцевий продукт і розширює аудиторію якій буде цікавий цей продукт. Коротко, про деякі з них.

Можливість кастомізації додатку реалізована в багатьох інших системах, це дозволяє користувачам налаштувати додаток під себе, обрати свою індивідуальну кольорову схему, а також можливо створити свою власну, та за необхідністю поділитися нею з іншими.

Локалізація – дуже цінна річ, котра дозволить додатку існувати не тільки на рідній землі, а й процвітати за її кордонами.

# ВИСНОВОК



У результаті виконання роботи було спроектовано та розроблено інформаційно-аналітичну систему для перегляду даних про продажі з торгової точки.

У роботі вирішено наступні задачі:

1. Проведено аналіз та опис предметної області програмного продукту. Виділено основні бізнес–процеси підприємства «Q-Store», його ціннісні пропозиції, цілі, споживчі сегменти, функції, ключові ресурси, витрати та джерела доходів, що допомогло у розробці специфікації вимог до системи.

2. Спроектовано систему, розроблено специфічні та функціональні вимоги до системи, проведено опис діаграм використання системи за типами користувачів за допомогою мови UML. Обґрунтовано актуальність технологій розробки, фреймворків та архітектурних рішень, що забезпечують якісність та продуктивність роботи системи, спроектовано діаграму класів.

3. Описано кроки реалізації, наведено код системи з коментарями до нього, описано основні архітектурні рішення клієнтської частини проекту.

4. У результаті тестування розробленої системи було встановлено, що вона відповідає вимогам які були створені до неї.

Отримана система автоматизує та прискорює перегляд даних. Крім цього, у подальшу розробку покладено модель, за якою власники системи можуть розвивати систему у потрібному напрямку.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сомервилл, Иан. Инженерия программного обеспеченияю. Москва, 2017. 143 с.

2. Брукс, Ф. Мифический человекомесяц или как создаются программные системы. Москва, 2006. 45 с.

3. Диаграмма прецедентов. Википедия: веб-сайт. URL: https://ru.wikipedia.org/wiki/Диаграмма_прецедентов (дата звернення: 01.09.2021).

4. Орлов С.А. Технологии разработки программного обеспечения. Питер, 2012. 36 с.

5. Почему интуитивно понятный интерфейс важен для веб-дизайна и разработки программного обеспечения. Цифровые технологии: веб-сайт. URL: https://it-wo.ru/it-tehnologii/pochemu-intuitivno-ponyatnyj-interfejs-nastolko-vazhen-dlya-veb-dizajna-i-razrabotki-programmnogo-obespecheniya (дата звернення: 15.09.2021).

6. В.В. Войтенко. C/C++ : теорія та практика / В.В. Войтенко, А.В.Морозов – Житомир, 2003.

7. Документація про Сигнали і слоти. Qt Documentation. веб-сайт. URL: https://doc.qt.io/qt-5/signalsandslots.html (дата звернення 08.04.2020)

8. Інформація про Qt. Вікіпедія: веб-сайт. URL: https://uk.wikipedia.org/wiki/Qt (дата звернення 01.10.2021)

9. Сигналы и слоты в Qt. Хабр: веб-сайт. URL: https://habr.com/ru/post/50812/

10. Довідник по Transact-SQL: веб-сайт. URL: https://docs.microsoft.com/ru-ru/sql/t-sql/language-reference?view=sql-server-ver15

11. Пасічник В.В. Організація баз даних та знань / Пасічник В.В., Резніченко В.А. – К.: Видавнича група BHV, 2006.

12. Qt5.10. Профессиональное программирование на C++ / Макс Шлее - СПб, BHV-СПб, май 2018

13. Qt. Профессиональное программирование. Разработка кроссплатформенных приложений на C++ / Марк Саммерфилд - СПб, Символ-Плюс, май 2011

14. Python и анализ данных / Маккинни У. - ДМК Пресс, 2020

15. Работа с данными в любой сфере / Кирилл Еременко - ООО «Альпина Паблишер», 2019

16. Аналитическая культура / Карл Андерсон - ООО «Манн, Иванов и Фербер», 2017

17. Куєтовська С. В. Методологія системного підходу та наукових досліджень: Курс лекцій. – Тернопіль: Економічна думка, 2005. – 124 с.

18. Проектування інформаційних систем [Електронний ресурс] – Режим доступу до ресурсу: https://uadoc.zavantag.com/text/1719/index-9.html.

19. Уніфікована мова моделювання UML [Електронний ресурс] – Режим доступу до ресурсу: http://www.znannya.org/?view=uml.

20. Застосування UML (Частина 2). Діаграма послідовності [Електронний ресурс] – Режим доступу до ресурсу: http://www.dut.edu.ua/ua/news-1-626-7897-zastosuvannya-uml-chastina-2-diagrama-poslidovnosti---sequence-diagram_kafedra-kompyuternih-nauk-ta-informaciynih-tehnologiy.

21. UML - Activity Diagrams [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/uml/uml_activity_diagram.htm.

22. What is Package Diagram [Електронний ресурс] – Режим доступу до ресурсу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-package-diagram/.