Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: **Інформаційна система управління мережею кав'ярень**

Виконав: студент IV курсу, групи СТ-41
спеціальності 126 Інформаційні системи та технології
(шифр і назва спеціальності)

(підпис) Тернівська А.П.
(прізвище та ініціали)

Керівник (підпис) Никитюк В.В.
(прізвище та ініціали)

Нормоконтроль (підпис) Шимчук Г.В.
(прізвище та ініціали)

Завідувач кафедри (підпис) Боднарчук І.О.
(прізвище та ініціали)

Рецензент (підпис) Цуприк Г.Б.
(прізвище та ініціали)

Тернопіль
2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«___» _____ 202_ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 126 Інформаційні системи та технології
(шифр і назва спеціальності)

Студенту Тернівська Анна Петрівна
(прізвище, ім'я, по батькові)

1. Тема роботи Інформаційна система управління мережею кав'ярень

Керівник роботи Никитюк Вячеслав Вячеславович кандидат технічних наук, доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « *29* » *квітня* 2024 року № *4/7-471*

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи Наукові публікації, електронні ресурси, підручники , посібники

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. Розділ 1. Аналіз предметної області проектування інформаційної системи. 1.1. Теоретичні основи ведення кавового бізнесу. 1.2. Порівняльний аналіз відомих рішень. 1.3. Реалізація архітектури програмної системи. 1.4. Проектування структури бази даних.Розділ 2. Програмна реалізація системи. 2.1. Вибір засобів розробки програмної системи. 2.2. Вибір системи управління базами даних. 2.3. Особливості програмної реалізації. Розділ 3. Тестування системи управління мережею закладів громадського харчування громадського харчування. 3.1. Тестування інормаційної системи. 3.2. Розрахунок обсягу даних та навантаження на програмну систему. 3.3. Інструкція користувача. 4. Безпека життєдіяльності основи охорони праці. Висновки. Перелік джерел. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Тема. 2. Мета роботи. 3-4. Аналіз відомих рішень.5. Спрощене представлення архітектури Системи. 6. Багаторівнева архітектура клієнт-сервер. 7. Схема роботи технології JDBC. 8. ER діаграма бази даних. 9. Середовище розробки. 10. Програмна реалізація . 11. Діаграма компонентів модуля «замовлень». 12. Тестування системи. 13. Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	доцент кафедри МТ Сенчишин Віктор Степанович		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	07.05.2025	*Виконано*
2.	Підбір джерел про інформаційну систему управління	08.05.2025-15.05.2025	*Виконано*
	мережею кав'ярень		
3.	Опрацювання джерел по темі кваліфікаційної роботи	16.05.2025-22.05.2025	*Виконано*
4.	Виконання дослідження щодо інформаційної системи		
	управління мережею кав'ярень		
5.	Оформлення розділу «Аналіз предметної області	23.05.2025-01.06.2025	*Виконано*
	проектування інформаційної системи.»		
6.	Оформлення розділу «Програмна реалізація системи»	23.05.2025-01.06.2025	*Виконано*
6.	Оформлення розділу «Тестування системи управління		
	мережею закладів громадського харчування	23.05.2025-01.06.2025	*Виконано*
	громадського харчування.»		
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	02.06.2025-05.06.2025	*Виконано*
8.	Виконання завдання до підрозділу «Основи охорони	02.06.2025-05.06.2025	*Виконано*
	праці»		
9.	Оформлення кваліфікаційної роботи	06.06.2025-10.06.2025	*Виконано*
10.	Нормоконтроль	11.06.2025-13.06.2025	*Виконано*
11.	Перевірка на плагіат	16.06.2025	*Виконано*
12.	Попередній захист кваліфікаційної роботи	18.06.2025	*Виконано*
13.	Захист кваліфікаційної роботи		

Студент _____ Тернівська А.П.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Никитюк В.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Інформаційна система управління мережею кав’ярень // Кваліфікаційна робота освітнього рівня «Бакалавр» // Тернівська Анна Петрівна // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп’ютерно-інформаційних систем і програмної інженерії, кафедра комп’ютерних наук, група СТ-41 // Тернопіль, 2024 // С. 62, рис. – 34, табл. – 3 , додат. – 2, бібліогр. – 32.

Ключові слова: інформаційна система, програмна система, кав’ярня, автоматизація, мобільний за стосунок, проектування систем.

Кваліфікаційна робота присвячена розробці програмної системи управління мережею закладів громадського харчування. В першому розділі кваліфікаційної роботи здійснено аналіз предметної області дослідження, проаналізовано відомі програмні системи, визначено їх переваги та недоліки, а також описано архітектуру програмної системи, здійснено аналіз та проектування структури бази даних.

В другому розділі кваліфікаційної роботи проведено аналіз засобів розробки програмної системи, описано основні засоби реалізації бази даних та основні проблемні моменти програмної реалізації.

В третьому розділі кваліфікаційної роботи реалізовано інтерфейс системи, який є інтуїтивно зрозумілим, одноманітним для легшого та комфортнішого сприйняття. Описано процес тестування та описано основні особливості роботи.

Об'єктом дослідження є процеси створення програмної системи управління мережею закладів громадського харчування.

Предметом дослідження є методи та програмні засоби створення програмної системи управління мережею закладів громадського харчування.

ANNOTATION

Information System for Managing aNetwork of Coffee Shops // Qualification work of the educational level "Bachelor" // Ternivska Anna Petrivna // Ternopil National Technical University named after Ivan Pulyu, Faculty of Computer Information Systems and Software Engineering, Department of Computer Sciences, ST group -41 // Ternopil, 2024 // P. 62, fig. – 34, tab. – 3, add. – 2, bibliography - 32.

Keywords: information system, software system, coffee shop, automation, mobile for relationship, system design.

The qualification work is devoted to the development of a software system for managing a network of public catering establishments. In the first section of the qualification work, the subject area of the study was analyzed, known software systems were analyzed, their advantages and disadvantages were determined, and the architecture of the software system was described, and the database structure was analyzed and designed.

In the second section of the qualification work, an analysis of software system development tools was carried out, the main means of database implementation and the main problematic points of software implementation were described.

In the third section of the qualification work, the interface of the system is implemented, which is intuitive and uniform for easier and more comfortable perception. The testing process is described and the main features of the work are described.

The object of the study is the process of creating a software system for managing a network of public catering establishments.

The subject of the research is methods and software tools for creating a software system for managing a network of public catering establishments.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі активно розвиваються різні інформаційні технології. Однією з предметних областей, де постійно застосовуються подібні технології, області, де потрібно вести облік. Особливо складною є проблема організації обліку на територіально-розподілених виробничих об'єктах, де потрібно інтеграція різних інформаційних систем, розподілених у просторі. Як приклад такого об'єкта можна навести організацію громадського харчування. Мережа кавових закладів – вид діяльності, який відноситься до ведення бізнесу, що вимагає обліку придбаного харчового матеріалу та обліку продажів. Даний облік ускладнений великою кількістю географічно розподілених об'єктів: самих закладів, їх постачальників і працівників, які використовуються ними облікових систем.

Об'єктом дослідження є процеси створення програмної системи управління мережею закладів громадського харчування.

Предметом дослідження є методи та програмні засоби створення програмної системи управління мережею закладів громадського харчування.

Мета та задачі дослідження. Метою кваліфікаційної роботи є розробка програмної системи управління мережею закладів громадського харчування.

Для вирішення завдань обліку часто потрібно використовувати ту чи іншу розподілену базу даних, керовану системою управління базами даних (СУБД). В базі даних можна, можливо зберігати всю необхідну інформацію. Це можуть бути найменування постачальників продуктів, адреси, телефони, фотографії продукції в базах даних можна вести облік платежів за придбану продукцію. Для цього у базі даних існують таблиці. Безпосередньо заповнювати таблиці достатньо незручно. Особливо для кінцевого користувача, який далекий від адміністрування СУБД.

Звичайно, безпосередньо в СУБД теж можна, можливо створювати форми для введення даних, запити та звіти. Однак, від кінцевого користувача це буде

вимагати більше глибоких знань основ роботи з персональним комп'ютером. Другим мінусом такого підходу буде те, що у кінцевого користувача повинна бути встановлена СУБД у рамках якої буде функціонувати база даних. Це накладає додаткове навантаження на бюджет юридичного особи, а також на технічні знання користувача. Тому найкращою буде база даних, керувати якою буде сторонній додаток, який може бути написано для вирішення завдань конкретного підприємства.

Практичне значення одержаних результатів. Особливістю даної розробки є використання сучасних інформаційних технологій при проектуванні, реалізації та тестуванні програмної системи управління мережею закладів громадського харчування на прикладі кав’ярень.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

1.1 Теоретичні основи ведення кавового бізнесу

Сучасна економічна система представляє з себе тісно інтегровані форми бізнесу, які існують та залежать від існування інших видів бізнесу. Більшість з них тісно пов'язані з інформаційними технологіями в тій чи іншій ступені.

Не винятком є і малі види бізнесу. Хтось із малого бізнесу віддає свої переваги таким програмним продуктам з ведення обліку як 1С Підприємство, хтось іншим системам. А деякі фірми замовляють індивідуальні програми, які найбільш точно охоплюють потреби ведення обліку.

Під час ведення кавового бізнесу необхідно вести облік товарів, продуктів, облік постачальників товарів. Дані товари використовуються для подальшого приготування продукції. Такими напівфабрикатами можуть виступати цукор, кава в зернах, молоко, вершки та інші. При змішуванні їх утворюється кінцевий продукт, який продається користувачеві.

Весь цей механізм необхідно враховувати, щоб полегшити працю та виключити по можливості помилки, пов'язані з людським фактором.

Для цього призначені комп'ютерні програми, пов'язані з обліком та звітністю. Вони допомагають тримати у полі зору відомості про всіх постачальників, товарів, всіх продані постачальниками товари за місяць, тиждень, день.

При закритті каси слід також підраховувати всі операції, які зробив оператор-барист.

Також необхідно вести облік по всій мережі кавових закладів. Таким чином, кавовий бізнес – це складний механізм, де потрібно розраховувати всі нюанси щодо взаємодії з постачальниками та клієнтами.

У цьому допомагає програмне забезпечення.

Мережа кавових закладів – вид діяльності, який відноситься до ведення бізнесу, що вимагає обліку придбаного харчового матеріалу та обліку продажів. Даний облік ускладнений великим кількість географічно розподілених об'єктів: самих закладів, їх постачальників і працівників, які використовуються ними облікових систем. Тому для вирішення зазначеною проблеми доцільно побудова розподіленою інформаційної системи на базі архітектури « клієнт - сервер».

1.2 Порівняльний аналіз відомих рішень

Розглянемо основні програми, які є аналогами до розроблюваної системи та визначимо їх основні переваги та недоліки.

Fusion POS (рисунок 1.1.) - чудовий помічник в оптимізації процесів для ресторану та кафе. Режим «Кафе» включає обслуговування по залах, столах, з можливістю перенесення замовлення, його об'єднання або поділу; режим «Фастфуду» підійде для обслуговування на замовлення — без вибору зали та столу, де касовий апарат – там Fusion POS.

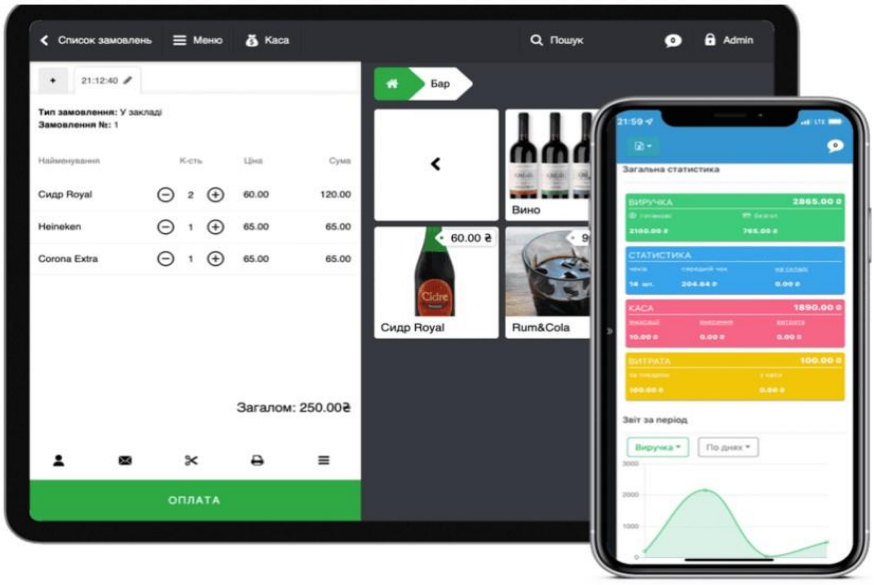


Рисунок 1.1. – Система Fusion POS

POS-система Poster - програма автоматизації та складського обліку закладів HoReCa та громадського харчування (рисунок 1.2). Швидка та надійна онлайн-каса на планшеті iPad або Android. Одна POS-система вирішує всі питання: онлайн-каса, склад, фінанси, аналітика, CRM.

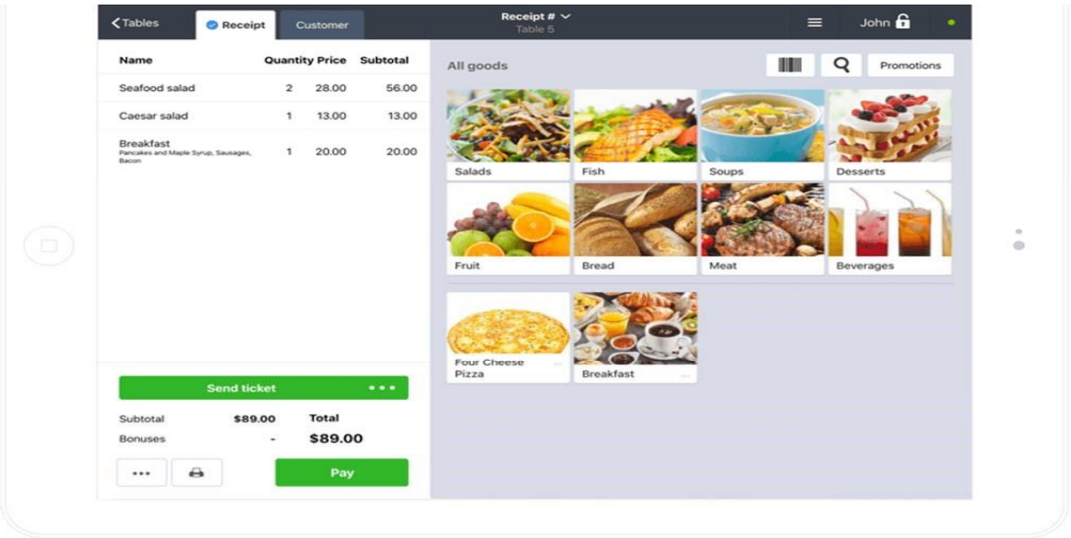


Рисунок 1.2. – POS-система Poster

GBS.Market. Проста в освоєнні програма для автоматизації кафе. Широкий набір можливостей та невисока вартість. Підтримка торговельного обладнання. Робоче місце офіціанта, управління замовленнями, калькуляція та виробництво, взаємодія з кухнею (рисунок 1.3).

Програма автоматизації торгівлі GBS.Market омає потенціал, який включає все для вирішення облікових завдань у малому та середньому бізнесі: організовує технологічний порядок у продажі, веде складський облік, видає детальну статистику щодо функціонування вашого бізнесу.

Товари та категорії. Поділ асортименту за категоріями та підкатегоріями. Товари групуються вами за певним принципом, при цьому програма дозволяє створювати необмежену кількість категорій та підкатегорій.

Створюється окрема картка для кожного товару.

Фотографії для товару у будь-якій кількості. Підтримуються формати JPG, PNG, BMP.

5 додаткових полів у картці товару для характеристик, які ви можете змінювати та заповнювати за потреби.

5 полів для ціни на кожен товар, щоб можна було вказати роздрібні, оптові, дрібнооптові ціни тощо.

Рисунок 1.3. –Система GBS.Market

Створення асортиментних рядів, тобто коли товар має один штрих-код, але, наприклад, різні розміри, колір та інше. Ця опція рятує від пересортування!

Формування складових товарів (наборів) та комплектів (калькуляція), наприклад, у кафе кава може бути складовим товаром, який складається зі стаканчика, самого напою, молока, палички, цукру тощо. При цьому продається набір або комплект, а списання здійснюється товарів зі складу.

Пошук за всім асортиментом. Можна використовувати фільтри за штрихкодом, найменуваннями, кількістю, ціною та іншими характеристикам товарів.

Синхронізація з EXCEL. Приймання і додавання товарів можна завантажувати не тільки вручну, але й з документів EXCEL.

Дії з товарами

Облік надходження товару до складу з можливістю позначки постачальника.

Облік продажів готівкою, банківською карткою, бонусами (балами). При цьому можлива комбінація з будь-яких типів оплати. Наприклад, бонуси + готівка, банківська картка + готівка, готівка + банківська картка + бонуси і так далі.

Облік продажів зі знижкою, у борг або на виплат, якщо ви таке практикуєте.

Списання товарів із зазначенням причини.

Повернення вже проданих товарів із зазначенням причини. При цьому можна повернути весь перелік за чеком або лише окрему позицію з нього.

Бронювання або резервування товарів. Опція працює з клієнтською базою, резерв або бронювання прив'язується до певного клієнта із зазначенням терміну дії резерву.

Переміщення асортименту між магазинами мережі.

Журнал історії дій з товаром. Ведеться з моменту занесення товару у базу, формує окрему вкладку в картці самого товару.

Робота з постачальниками. У програмі GBS.Market можна вести аналітичний облік щодо роботи з постачальниками товарів:

Програма дозволяє здійснювати облік за товарами під повну або часткову їх реалізацію від постачальника.

Є можливість формувати накладні взаєморозрахунків у зручній табличній формі.

У GBS.Market вказуються всі заборгованості при розрахунках з постачальниками, якщо вони існують.

Є вибірка даних щодо кожного постачальника окремо за зазначений період. Період ви можете обрати будь-який, що вас цікавить.

Для кожного постачальника є окреме поле для коментаря. У ньому ви відзначаєте нюанси роботи з тим чи іншим постачальником, наприклад, що він завжди затримує постачання, тому дозамовлення краще робити заздалегідь.

Усі дані, що формує GBS.Market щодо постачальників, можна експортувати у формат EXCEL для друкування чи надсилання як звіту постачальникові.

PalmaBox Horeca - сучасна хмарна програма автоматизації закладів харчування та торгівлі (рисунок 1.4).

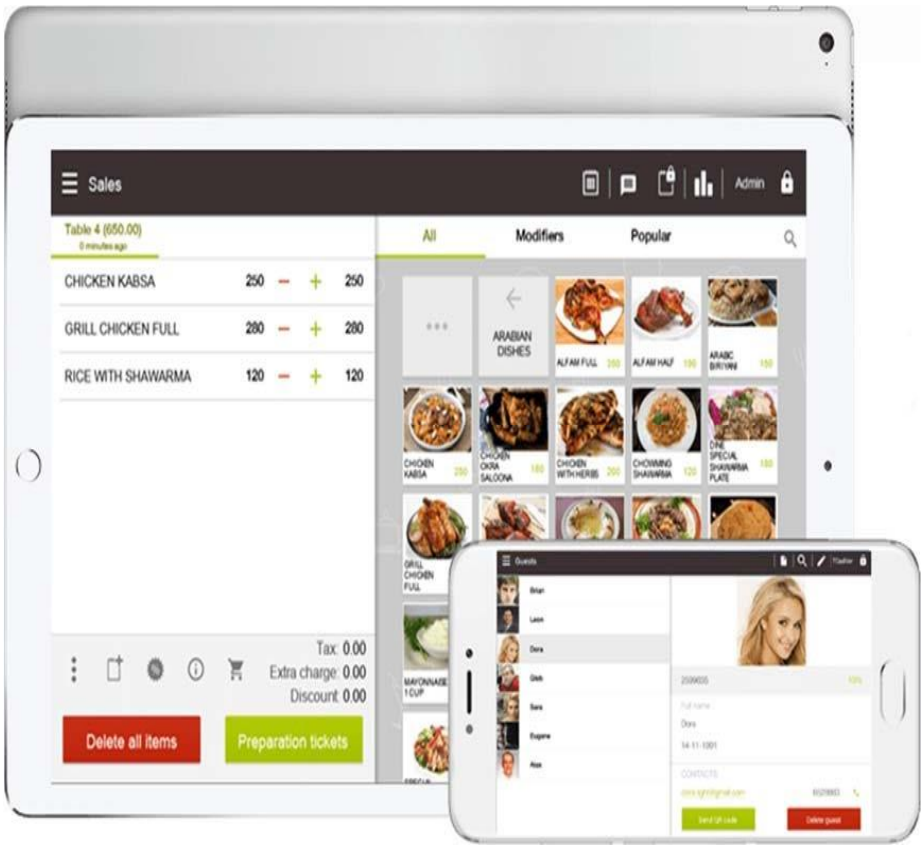


Рисунок 1.4. – Система PalmaBox Horeca

Прайс без обмежень – страви, товари, послуги. Управління замовленнями – від створення до оплати. Продажі в залі, при доставці, виїзній торгівлі. Обмін даними з кухнею, баром та складом. Карта столів. Виведення замовлення на екрани кухні та залу, на чек. Всі види оплат, інтеграція з касою та автоматичні звіти керуючому.

Основним недоліком перелічених вище систем є їх висока ціна та складність розгортання для невеликихї кав’ярень.

1.3 Реалізація архітектури програмної системи

У рорботі буде розглядатися мережа кавових закладів, тобто мережа територіально-розподілених виробничих об’єктів, які використовують різні інформаційні системи. Для реалізації зазначеної мережі необхідно застосувати архітектуру "клієнт-сервер". Використання подібної архітектури можна, можливо обґрунтувати необхідністю здійснення централізації, для ведення єдиного обліку продажу та залишків продукції, планування постачання. Така система можна представити як дворівневу систему, що містить компоненти: клієнт та база даних, де:

«Клієнт»: користувач системи (офіціант, барист, менеджер, бухгалтер, касир тощо), що працює у визначеній точці (кав'ярні) через клієнта (модуль облікової програми, що розглядається далі).

"Сервер/База даних": розташована на окремому сервері, розміщеному в головному офісі (дата центрі), яка є єдиною для всіх "клієнтів", тобто містить єдине меню, постачальників тощо.

Застосування віддаленої бази даних в аналізованій системі необхідно для спрощення виконання агрегованих операцій, таких як ведення статистики, облік закупівель та витрати сировини.

Побудова інформаційних систем є досить складним процесом. Системи, що працюють в економічних предметних областях, що обробляють бізнеспроцеси, або провідні облік, будуються на основі технологій баз даних. Незважаючи на поступовийе розвиток технології NoSQL, все ж таки в даний час час значно переважає технологія побудови реляційних баз даних. У світі побудови реляційних баз даних існує багато форматів даних. Обробляються дані, так званими СУБД, системами управління баз даних. Так ось, глобально прийнято ділити СУБД на локальні та промислові.

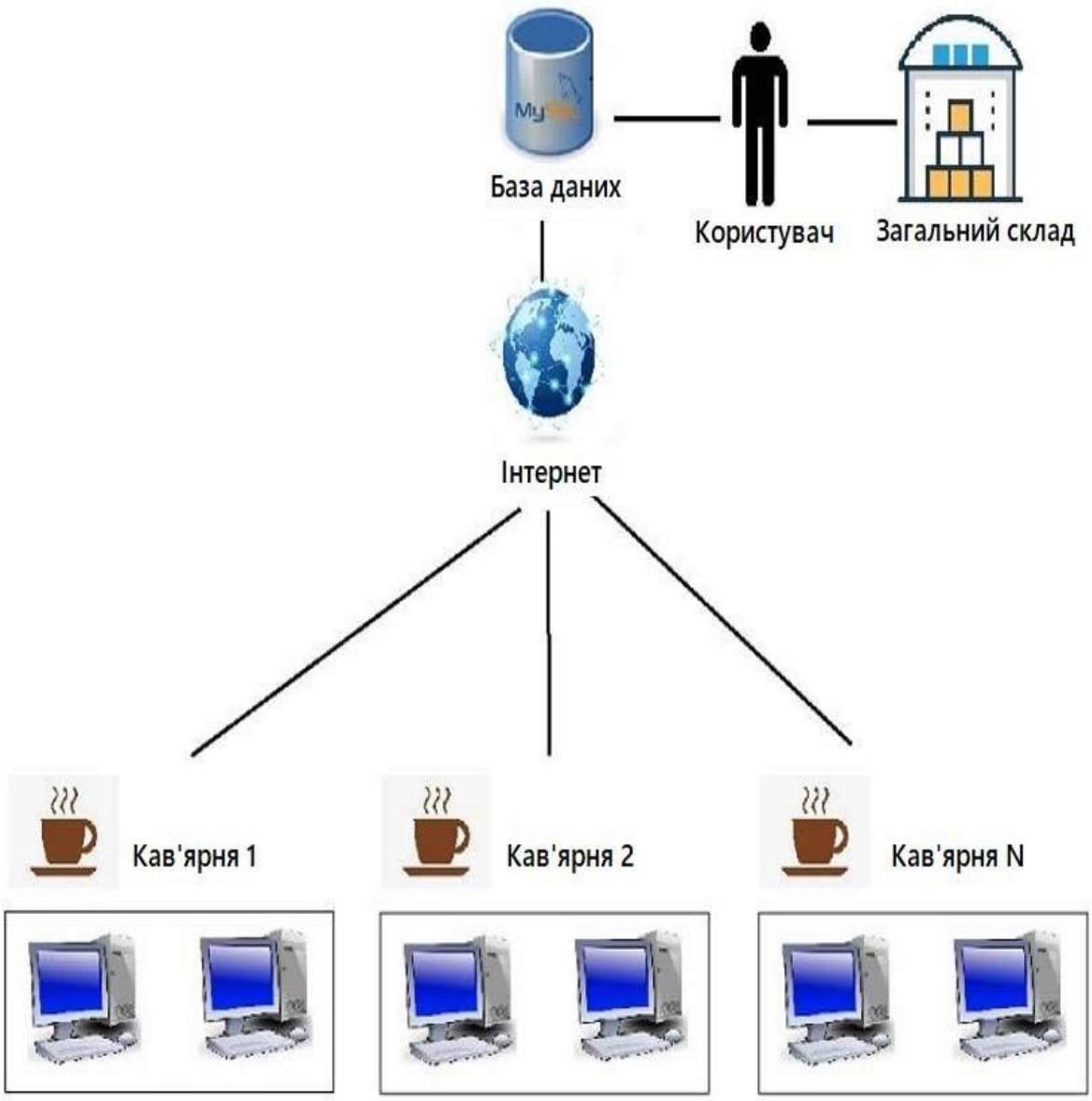


Рисунок 1.5 – Спрощене представлення архітектури системи

Локальні СУБД підприємства для обробки невеликих завдань у межах одного підприємства. Наприклад, це може бути нарахування заробітної плати, облік товарів на складі. Та й взагалі абсолютно будь-яке завдання, яке вимагає обліку може бути вирішено із застосуванням технологій локальних СУБД. У локальних СУБД є два суттєві обмеження. Перше обмеження пов'язано з максимально допустимим числом полів у таблиці, числом записів у таблиці, розміром таблиці і так далі. Наприклад у Visual FoxPro максимально допустима кількість записів у таблиці не може перевищувати 1 мільярд записів, а максимальний розмір файлу таблиці не може перевищувати 2 гігабайти. Таким

чином, у локальних базах даних є певні обмеження. Для маленького та середнього за величиною підприємства досягти 1 мільярда записів у таблиці дуже важко, тому подібні обмеження не суттєві. До того ж, можна робити архівні копії, перезаписуючи частину старих записів у вигляді архіву в окрему базу даних.

Істотні обмеження виявляються у кількості користувачів, які одночасно працюють із СУБД. Їх кількість, як правило не може перевищувати одного, двох десятків. При добре працюючій мережі, кількість користувачів зростає до кількох десятків. Але все ж таки, коли користувачів занадто багато, система починає працювати повільно. І виною тому є технологія файл-серверної системи, які використовують локальні СУБД.

Наступним етапом у розвитку стала клієнт-серверна технологія побудови систем, що працюють на основі баз даних. Таким чином, існують дві основні моделі у побудові бази даних, звані архітектурою СУБД: файл-сервер, що підходить для проектування невеликих систем та клієнт-сервер, призначені для систем корпоративного рівня.

Системи файл-серверного типу працюють в такий спосіб. Коли програма посилає запит базі даних, то в результаті програмі передаються всі дані та обробка цих даних, наприклад масову заміну одного значення на інше, веде клієнтська програма. При цьому на мережу створюється велике навантаження. Коли комп'ютерів, що звертаються до однієї і тієї ж бази даних стає занадто багато, мережа починає давати збої у роботі.

При клієнт-серверному підході справа іде інакше. Програма-клієнт посилає дані на сервер у вигляді запиту. Сервер обробляє запит клієнта та посилає йому лише необхідний результат, а не всі дані. Таким чином, мережевий трафік залишається не завантаженим.

Як правило, бази даних, що відносяться до клієнт-серверних технологій розташовуються на окремих комп'ютерах мережі, які називаються серверами мережі. Програма (СУБД), яка обробляє цю базу даних застосовує мову SQL,

тому такі програми в рамках клієнт-серверної технології прийнято називати SQL-серверами.

Клієнтський додаток звертається, раніше всього до SQL-серверу, тобто, до СУБД, що у свою чергу, обробляє базу даних. Таким чином, досягається централізація в управлінні безпекою доступу до даних. Наприклад, тригери і збережені процедури, написані мовою SQL і які у базі даних (керовані при цьому SQL-сервером), будуть діяти всім клієнтам.

Кількість клієнтів при використанні технології клієнт-сервера практично не обмежено. Прикладом може служити будь-який великий інтернет-ресурс. Доступ, наприклад, до порталу https://www.facebook.com/ може досягати багато тисяч користувачів одночасно, що працюють у системі. Їх кількість може бути обмежено тільки фізичної продуктивністю кластера серверів, на якому розташована СУБД.

Локальну базу даних, наприклад, Access, те саме можна розташувати на окремому мережевому комп'ютері, а клієнти будуть розташовані на різних машинах мережі, але це не зробить цю систему клієнт-серверною. Вона як і раніше буде системою з файл-серверною архітектурою мережі.

Крім практично необмеженої кількості користувачів інформації у системі клієнт-сервера, у неї є ряд інших переваг у порівнянні з файл-серверною технологією:

1) Достатньо сильна централізація управління даними, що жорстко позначається на безпеці та цілісності даних.

2) Центральне сховище файлів.

3) Сервера спільно використовують наявне програмне та технічне забезпечення.

4) Гнучка керованість при дуже великій кількості користувачів.

Але у цієї системи є і свої недоліки:

1) Дороге технічне забезпечення. Насамперед всього це дорогі сервери, які повинні обробляти величезну кількість інформації.

2) Дорогі ліцензії на серверні операційні системи.

3) Потрібен адміністратор для виконання масових операцій над даними.

Клієнт-серверні системи, загалом своєму випадку діляться на дворівневі та багаторівневі системи.

При багаторівневій організації архітектури клієнт-сервер між рівнем клієнтів та рівнем сервером додається проміжний рівень – рівень сервера програм (рисунок 1.6).

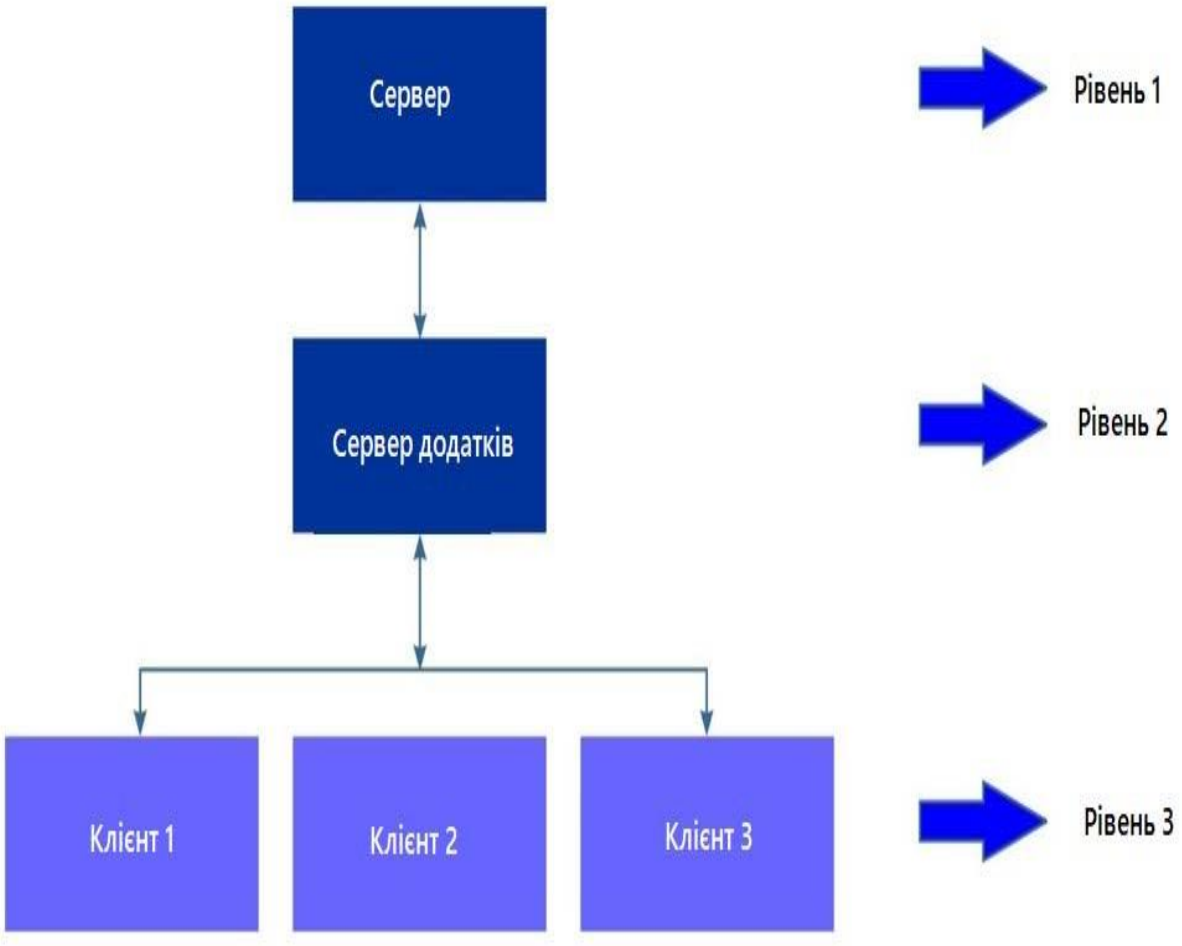


Рисунок 1.6. – Багаторівнева архітектура клієнт-сервер

При такій архітектурі лише на рівні додатків зберігається СУБД, а база даних зберігається окремо, на сервері – першому рівні. Розглянута

багаторівнева архітектура дозволяє більше гнучко розподіляти функції системи та навантаження між компонентами ПЗ, а також може знизити вимоги до апаратних ресурсів робочих місць клієнтів.

Після обробки дані пересилаються назад, у дельта-пакетах. У цих пакетах міститься інформація до внесення змін до запису та після внесення. Перед фізичними змінами у базі даних на сервері бази даних, сервер додатків здійснює аналіз коректності цих змін. Наприклад, якщо запис намагається змінити перший клієнт, а запис у цей час уже був змінено другим клієнтом, то зроблені зміни першого клієнта будуть відкинуті сервером додатків.

Сервера додатків створюються на основі модулів даних:

- Remote Data Module – для серверів DCOM, TCP/IP та OLEnterprise;
- Transactional Data Module – для сервера MTS;
- WebSnap Data Module – для сервера Web;
- SOAP Server Data Module – для серверів SOAP.

Побудова програми "тонкого" клієнта відрізняється від додатка "товстого" клієнта. Для "тонкого" клієнта слід організувати взаємодія між сервером додатків та "тонким" клієнтом. Забезпечити взаємодія щодо обміну даними між клієнтом та сервером.

Таким чином, побудова конкретного програмного комплексу з технології клієнт-сервер залежить від того, є чи створювана система двох або три-рівнева, а також від реалізації завдань для конкретної предметний області.

Для проектування системи була використана технологія доступу до даних Java Database. Доступ до даних у JDBC (рисунок 1.7) здійснюється за допомогою використання драйвером доступу до даних.

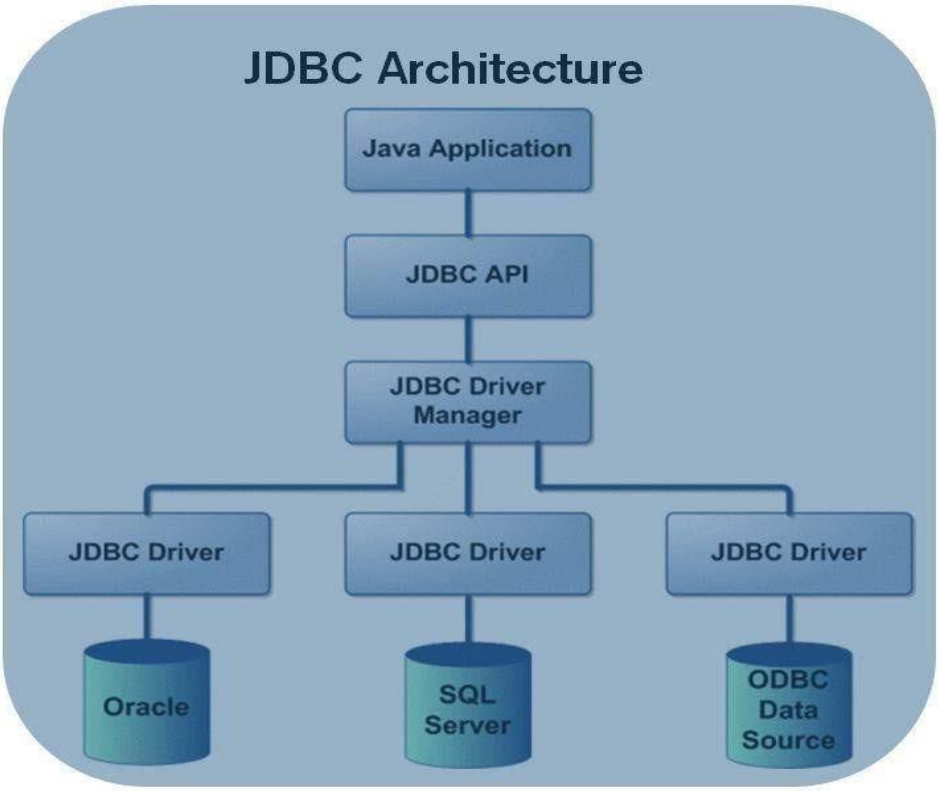


Рисунок 1.7. – Схема роботи технології JDBC

Для доступу до баз даних MySQL застосовуються свої драйвера, версія яких залежить від версії MySQL. У цьому проекті застосовується java.sql.Connection.

За допомогою технології JDBC можна підключитися до фізичної таблиці бази даних, отримати її за допомогою SQL запиту набір необхідних даних, виконати необхідні маніпуляції з даними та здійснити фізичне збереження з набору до таблиці. Також, технологія JDBC дозволяє здійснювати редагування бази даних без формування набору даних. JDBC була розроблено головним чином для реалізації архітектури "Клієнт-сервер".

Так, у реляційному способі доступу для отримання даних з тієї чи іншої таблиці бази даних використовується запит Select. Таким чином, застосовуючи даний запит можна, можливо сформувати вибірку, в яку потраплять не всі дані таблиці, а лише необхідні. Під цим мається на увазі, що крім необхідних

записів, так само можна обмежитися лише необхідними полями. А дані обробляє сервер.

За допомогою команди Select легко створювати багатотабличні запити, вибірка яких буде формувати об'єкт Recordset, який буде джерелом даних для створення користувацьких форм.

1.4 Проектування структури бази даних

Перед початком проектування БД її необхідно візуалізувати, тобто представити її структуру, таблиці, рядки та зв'язки між ними.

Проектування баз даних - процес створення візуальної схеми бази даних та визначення внутрішньої будови що входять до неї елементів та необхідних обмежень цілісності.

Логічне проектування БД - це процедура створення візуальної схеми бази даних на основі конкретної формальної моделі даних, наприклад, реляційної моделі даних. Для реляційної моделі даних будується даталогічна модель - набір схем зв'язків і відношень, зазвичай із зазначенням первинних ключів, а також «зв'язків» між відношеннями, що представляють собою зовнішні ключі.

Перетворення початкової концептуальної моделі, як правило, здійснюється за формальними правилами. Цей етап може бути у значній ступень автоматизований за допомогою спеціальних засобів візуального та фізичного проектування БД.

На етапі логічного проектування враховується специфіка конкретної моделі даних, але може не враховуватися специфіка конкретної СУБД. Оскільки у всіх об'єктів у мережі кав'ярень однакові постачальники, меню, ціни, продажі та загальна база співробітників, доцільно створити одну єдину базу, розміщену на сервері, до якої будуть звертатися "клієнти".

«Клієнт» це користувач системи (офіціант, барист, менеджер, бухгалтер, касир і т.д.), який працює у визначеній точці (кав'ярні) через клієнта (модуль

облікової програми, що розглядається далі). Клієнти розподілені, а сама база – централізована.

Ця БД містить 11 сутностей. Розглянемо основні з них і зв'язок між ним.

Сутність постачальники містить 10 атрибутів. Організація, повне найменування, телефон 1, телефон 2, адреса, email, ІПН, КПП, примітка. У цій сутності містяться основні відомості про постачальників (для цілей обліку). Між найменуванням організації та сутності «Постачальники» та сутності «Надходження » є зв'язок для однозначного ідентифікації постачальника.

Сутність «Вступ» містить 7 атрибутів: найменування товару, постачальника, кількість, ціна закупівлі, номер накладної, дата надходження, примітка.

Сутність «Співробітники» містить 5 атрибутів для цілей кадрового обліку. Дана сутність пов'язана з сутністю логіни та паролі.

Сутність «Напівфабрикати» містить 3 атрибути, найменування пов'язано з сутністю «Витрати». У Сутності витрата ведеться облік продажу з метою контролю витрат напівфабрикатів та дозамовлення їх у віддаленому складі.

Всі зв'язки в БД є зв'язками один до багатьох. Зв'язки між сутностями є ідентифікуючими, оскільки вторинний ключ потрапляє до атрибутів первинного ключа іншої сутності.

Проектування бази даних для цього проекту здійснювалося в MySQL Workbench 5.5. Робота з цим графічним клієнтом починається з формування локальною моделі бази даних. Це не фізична база даних, а віртуальна схема з полями та зв'язками між полями, а також іншими елементами бази даних.

Логічно дані представлені у вигляді об'єктів. До основних елементів структури БД відносяться об'єкти.

Структура таблиць БД складається з наступних основних елементів. Таблиці Логіни та паролі (LogPass) та Співробітники (Spiv) містять відомості про логінb, паролі та статус (активний чи ні) облікового запису та його ролі. При цьому за результатами тестування прийнято рішення з метою безпеки

хешувати паролі алгоритмом md5(), а пароль генерувати генератором псевдовипадкових чисел rand(). Для забезпечення сумісності, це буде реалізовано у наступних версіях програми, з додаванням додаткової ознаки "тип пароля": відкритий або хешований. Ця база ведеться централізовано, на віддаленому сервері та заповнюється керівництвом організації з головного офісу. Підставою для внесення відомостей є наказ з кадрів.

Таблиця відомості про надходження товарів від постачальників на віддалений склад, та містить найменування продуктів, їх артикули, відомості про постачальників для цілей бухобліку.

Таблиця Відгуки (Vidg) містить відгуки, які переносяться з паперових носіїв, таких як книга відгуків та пропозицій, або усних зауважень, зафіксованих документально. База ведеться для цілей ведення претензійної роботи.

Таблиця Витрата (Vytr) ведеться локально і служить для цілей обліку залишків сировини на локальному складі та оформлення замовлень на постачання на віддалений склад.

Таблиця Тип продажів (TipProd) містить відомості для цілей податкового обліку та містить номер касового чеке та тип продажу. Дані передаються до центральної БД після закінчення зміни для генерації статистики та звітів з продажу.

База повинна відповідати реляційній моделі даних, для цього необхідно побудувати ER діаграму, наведену на рисунку 1.8.

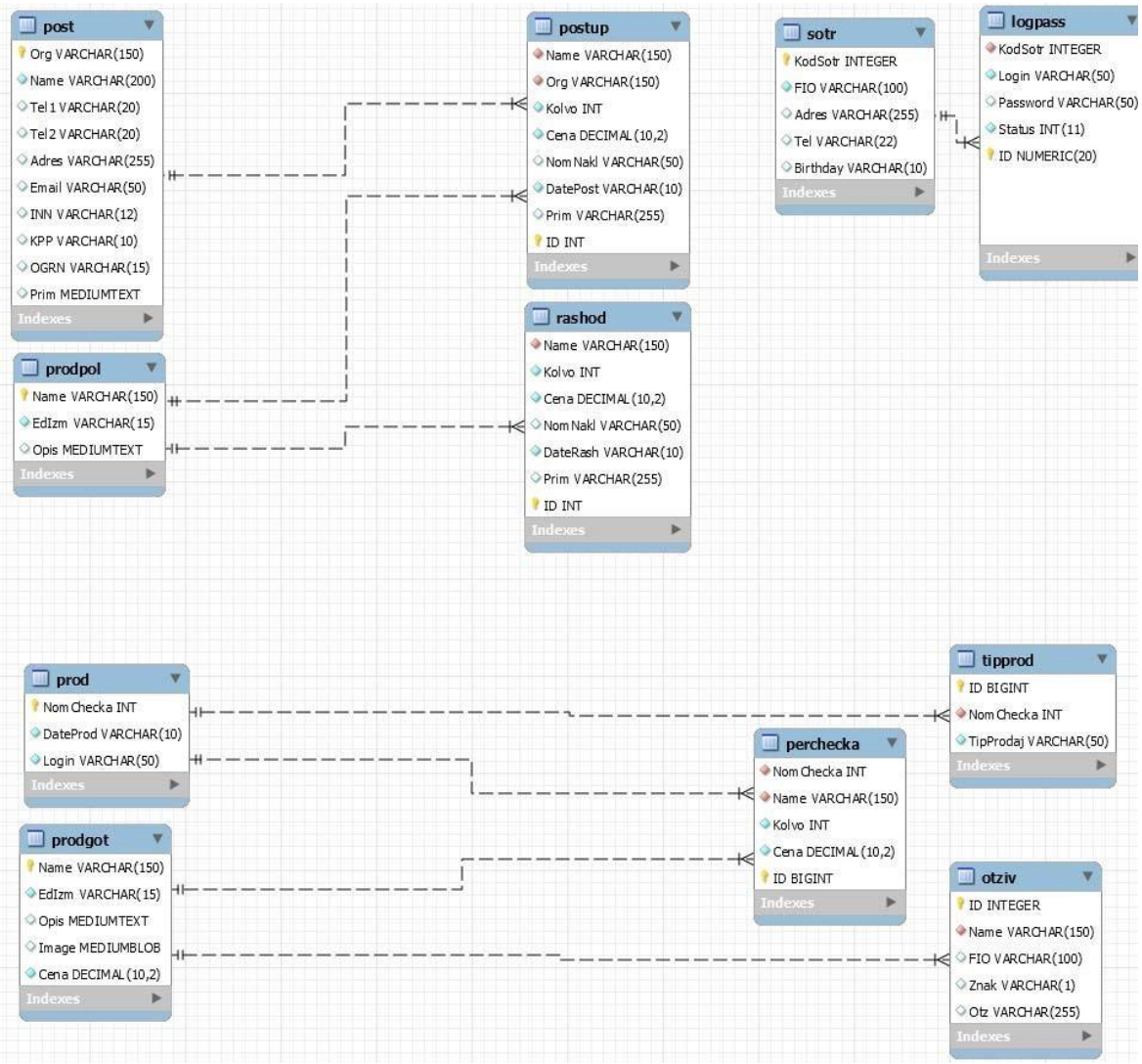


Рисунок 1.8. – ER діаграма бази даних

Діаграма дозволяє перевірити відповідність бази реляційної моделі, її логічної цілісності та правильності побудови зв'язків.

Після побудови моделі бази даних необхідно сформувати скрипт мовою SQL. Цей скрипт можна зберегти у файл. Сенс у цій операції в тому, щоб не набирати команди вручну. Графічне проектування у значній ступені економить час і зменшує ймовірність помилок.

Наступним етапом буде формування фізичної бази даних на підставі створеного скрипта, фрагмент якого представлений на рисунку 1.9.

```
USE `database`;

CREATE  TABLE IF NOT EXISTS `database`.`logpass` (
  `Login` VARCHAR(50) NOT NULL ,
  `Password` VARCHAR(50) NULL DEFAULT NULL ,
  `Status` INT(11) NOT NULL DEFAULT '0' ,
  PRIMARY KEY (`Login`) ,
  UNIQUE INDEX `Login_UNIQUE` (`Login` ASC) )
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;


CREATE  TABLE IF NOT EXISTS `database`.`prodgot` (
  `Name` VARCHAR(150) NOT NULL ,
  `EdIzm` VARCHAR(15) NOT NULL ,
  `Opis` MEDIUMTEXT NULL DEFAULT NULL ,
  `Image` MEDIUMBLOB NULL DEFAULT NULL ,
  `Cena` DECIMAL(10,2) NOT NULL ,
  PRIMARY KEY (`Name`) ,
  UNIQUE INDEX `Name_UNIQUE` (`Name` ASC) )
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

CREATE  TABLE IF NOT EXISTS `database`.`prod` (
  `NomChecka` INT NOT NULL AUTO_INCREMENT ,
  `DateProd` VARCHAR(10) NOT NULL ,
  `Login` VARCHAR(50) NOT NULL ,
  PRIMARY KEY (`NomChecka`) )
ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

```

Рисунок 1.9. – Фрагмент скрипта імпорту структури БД

Цей скрипт містить SQL-запити на перевірку існування таблиці та її створення за відсутності. Наступним етапом є створення полів таблиць та вказівка їх типів даних - текстових (string), числових (int) тощо, а також їх гранична довжина та вказівка на допустимість відсутності значення.

Надалі вибирається первинний ключ, наприклад порядковий номер запису в таблиці. На завершення вибирається тип таблиць та її кодування. Фрагмент скрипту імпорту початкової структури БД наведено у додатку А.

1.5 Висновки до першого розділу

Здійснено аналіз предметної області роботи, проаналізовано відомі програмні системи, визначено їх переваги та недоліки, а також описано архітектуру програмної системи, здійснено аналіз та проектування структури бази даних.

РОЗДІЛ 2. ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

2.1 Вибір засобів розробки програмної системи

В даний час час для розробки ПЗ існує достатньо багато різних засобів та середовищ. Це і C++, C#, JAVA і багато інші.

У кожній з поширених мов є свої переваги та недоліки, а також специфічна сфера застосування. Але тепер усі мови достатньо розвинені та стандартизовані, що вибір тієї чи іншої мови аргументується більше уподобаннями чи навичками конкретного програміста, ніж поставленими завданнями. Так само вибір мови аргументується спільної розробкою. Коли програміст влаштовується на певну роботу, де ведеться командна розробка ПЗ, то він змушений писати програмний код тією мовою, якою працює решта команди.

Для реалізації проекту була обрано середа розробки Java, саме Java SE 12 (рисунок 2.1) середеовище розробки IntelliJ IDEA Community Edition 2019.2.3 x64 (рисунок 2.2).

	ТИП ДАНИХ	ДІАПАЗОН ДОПУСТИМИХ ЗНАЧЕНЬ	ОБ'ЄМ ЗАЙМАНОЇ ПАМ'ЯТІ
цілочисельні	byte	от -128 до 127	1 байт
	short	от -32768 до 32767	2 байта
	int	от -2147483648 до 2147483647	4 байта
	long	от -9223372036854775808 до 9223372036854775807	8 байт
з плаваючою точкою	float	от -3.4E+38 до 3.4E+38	4 байта
	double	от -1.7E+308 до 1.7E+308	8 байт
символи	char	от 0 до 65536	2 байта
логічні	boolean	true или false	Для зберігання значення цього типу досить 1 біта, але в реальності пам'ять такими порціями не виділяється, тому змінні цього типу можуть бути по-різному упаковані віртуальною машиною.

Рисунок 2.1. –Типи даних у Java

JAVA SE 12 - це сучасна популярний мова програмування з гнучкими можливостями. JAVA базується на концепціях сучасного об'єктно-орієнтованого програмування:

- поліморфізм; - успадкування;
- інкапсулювання.

Поліморфізм – це можливість, за якої об'єкти, маючи однакову специфікацію здатні мати різну реалізацію.

Успадкування представляє собою технологію, коли дочірній об'єкт успадковує атрибути об'єкта батька.

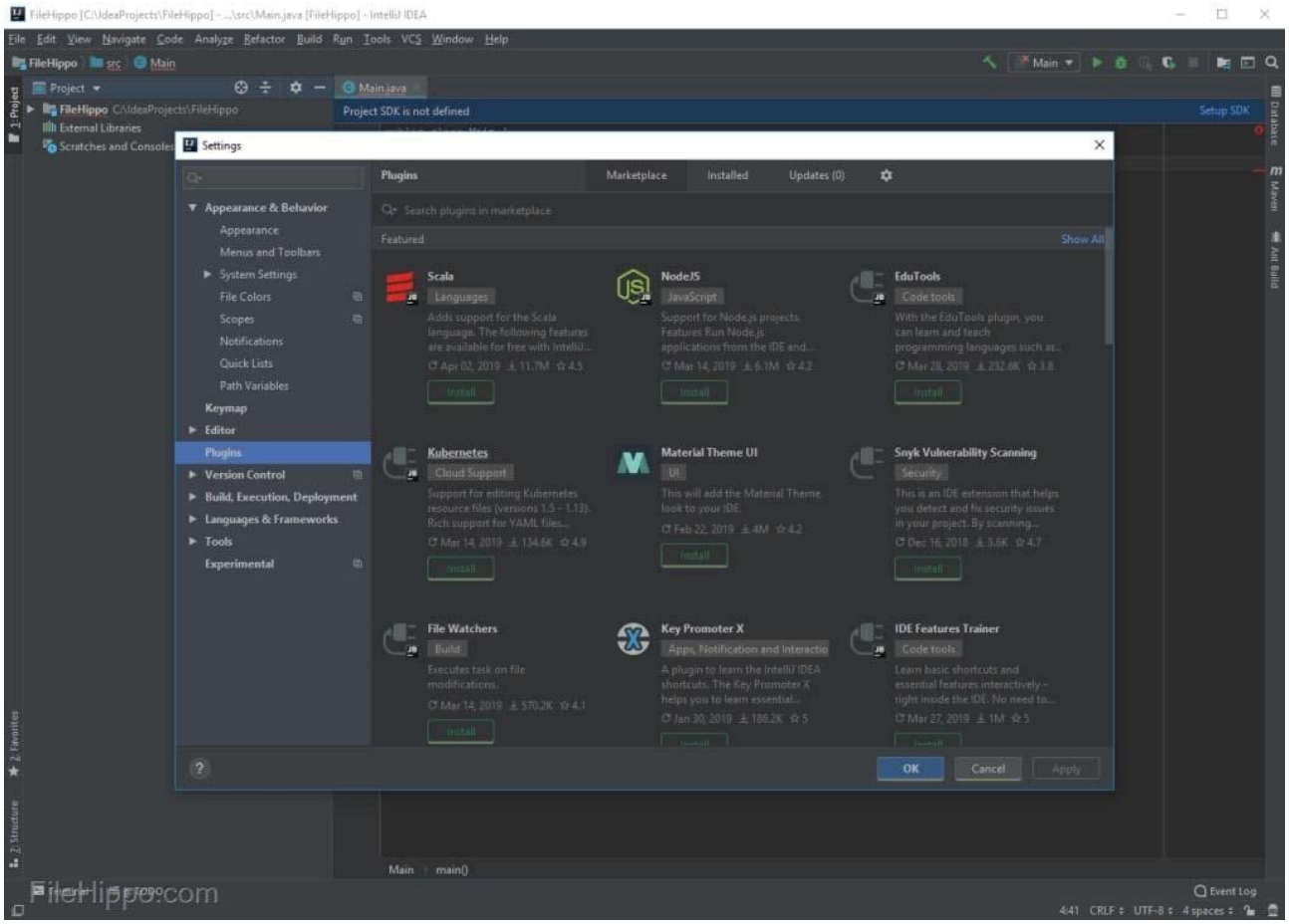


Рисунок 2.2. – Середовище розробки IntelliJ IDEA Community Edition 2019

Інкапсулювання – це об'єднання даних та функцій в єдиному компоненті.

JAVA 12 був розроблений компанією Sun. В даний час час JAVA належить компанії Oracle. Варто відзначити, що JAVA - це багатоплатформова мова розробки. Сучасні версії Java дозволяють писати програми для Андроїд.

Для роботи з базою даних JAVA представляє з себе численні технології. Це технологія JDBC та багато інші.

Для реалізації цього проекту була обрано технологія JDBC. Це «конектор» до бази даних.

Таким чином, JAVA 12 - це одна з самих потужних мов розробки прикладного програмного забезпечення, у тому числі й для роботи з базами даних у розподілених системах.

2.2 Вибір системи управління базами даних

В даний час час існує величезне число різноманітних засобів розробки ПЗ, багато з яких за функціоналом дуже схожі один на одного.

Для вирішення певної проблеми призначена та або інга мова програмування або середовище розробки. Зараз можна зробити висновок, що вибір середовища розробки вже залежить більше не від завдання, яке доведеться вирішити, а все більше від переваги програмістом тієї чи іншої мови. Це пов'язано з дедалі більшою універсальністю та розвитком систем розробки.

Облік у розподіленій інформаційної системі мережі кав'ярень зручно здійснювати за допомогою СУБД (Система управління базами даних) та прикладної мови програмування.

СУБД представляє собою систему, яка призначена для управління наявною базою даних. Серед СУБД існує багато рішень. Слід згадати, що раніше всього, СУБД діляться на настільні та промислові.

Настільні СУБД призначені для організації роботи з базами даних у рамках дрібних, середніх та великих підприємств.

Промислові ж, СУБД використовуються для організації роботи з базами даних, до яких звертаються мільйони користувачів. Наприклад, той же сайт державної статистики, який щодня відвідують тисячі користувачів всією країни. Дані цього, як і будь-якого іншого сайту зберігаються у базі даних, за форматом що належать до промислових СУБД.

Недоліком настільних СУБД є те, що зі збільшенням числа записів і зростання користувачів система починає повільно працювати. Для вирішення цією проблеми застосовується кешування результатів запитів та буферизація. Наприклад, система зберігання даних InnoDB для СУБД MySQL має можливість вказати розмір буфера, який буде зберігається у ОЗП. Якщо розмір ОЗП більший обміну бази, вся БД поміститься в оперативній пам'яті і буде працювати на порядок швидше.

І, якщо зі збільшенням обсягу записів (даних) у базі даних таких СУБД можна вирішити питання, шляхом архівування старих, хронологічно перших записів та чищення бази даних, то в міру збільшення числа користувачів і навантаження на СУБД питання залишається актуальним. Саме питання, що стосується навантаження і покликана вирішити промислова СУБД.

Серед настільних СУБД слід виділити такі як dBase та Visual dBase, Microsoft Access, Microsoft FoxPro та Visual FoxPro, Paradox.

Серед промислових СУБД можна виділити Oracle та Microsoft SQL Server, MySQL. Для здійснення даного проекту була обрана СУБД MySQL Server 5.5. Яка представляє з з себе реляційну систему управління базами даних. Може встановлюватися в операційну систему як служба, що є дуже зручно.

Класичним підходом до управління базою даних MySQL, включаючи створення самої бази даних, створення таблиць та зв'язків між ними, а також наповнення таблиць та інші операції здійснюється за допомогою клієнтського додатку у вигляді командного вікна, схожого на звичайне командне вікно Windows.

В даний час існує достатньо багато клієнтських програм, що виконують адміністрування баз даних MySQL у графічному режимі, так, як це здійснюється в Access. Звичайно, механізм візуального формування тут дуже відрізняється.

Основними програмами, що дозволяють здійснювати графічне проектування бази даних MySQL є PHPMyAdmin, MySQL Workbench, dbForge Studio for MySQL і багато інші програми.

Для здійснення даного проекту була обрано програма адміністрування MySQL Workbench 5.5 (рисунок 2.3).

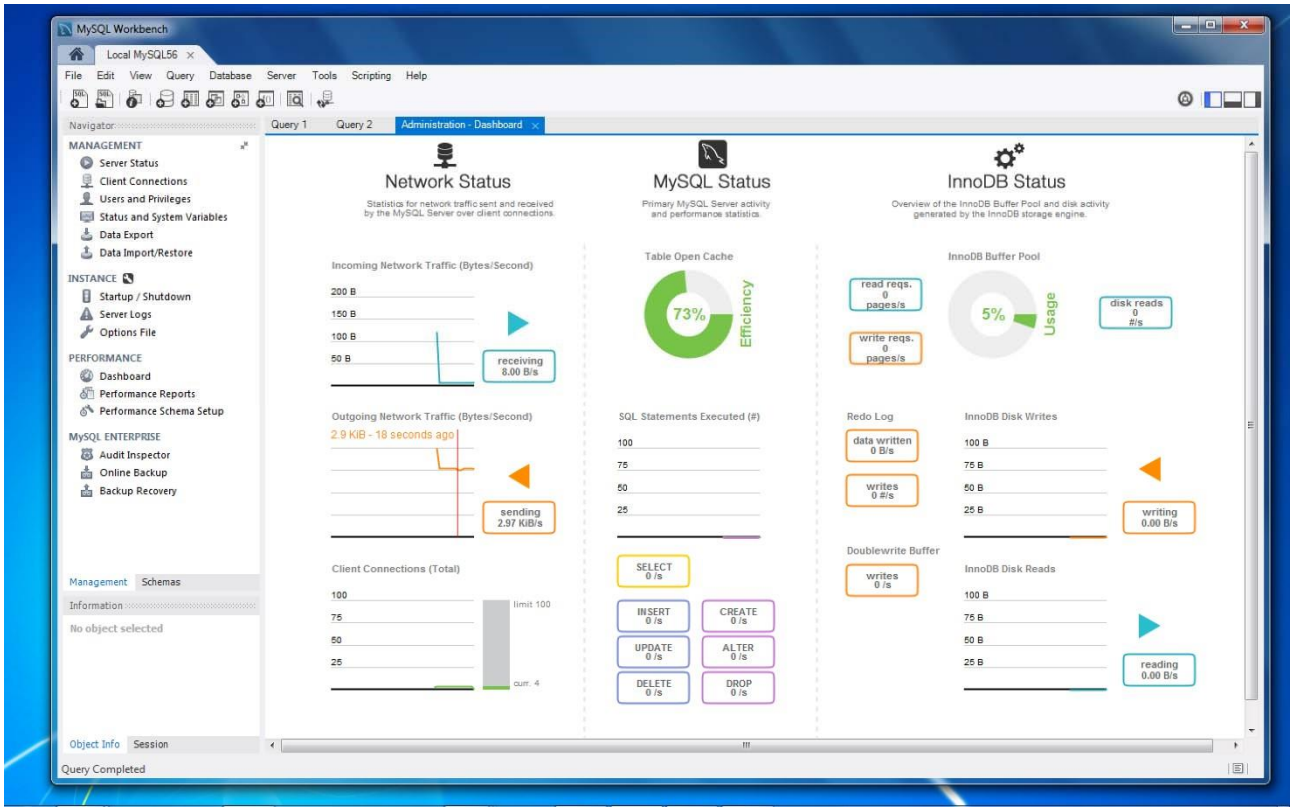


Рисунок 2.3. – Середовище розробки MySQL Workbench 5.5

Основою роботи СУБД MySQL є стандартизована мова управління даними SQL. В операційну систему встановлюється сервер, який представляє з себе програмне забезпечення. Цей сервер і є безпосередньо СУБД, якому посилаються адміністратором команди SQL, і здійснюється управління базами даних MySQL через клієнтський додаток.

Система управління базами даних MySQL характеризується високою швидкістю роботи та високої відмовостійкістю.

В даний час час підтримку та розвиток MySQL здійснює компанія Oracle. Вона ж супроводжує і клієнтські програми для адміністрування MySQL Workbench.

Відмінною особливістю MySQL від MS SQL Server є те, що останній в даний час найбільш орієнтований працювати з .Net мовами, тоді як MySQL працює зі багатьма іншими мовами програмування.

Основною відмінністю для більшості розробників є все ж таки різниця в синтаксисах самої мови SQL. Нижче наводяться основні відмінність трьох перерахованих промислових СУБД (рисунок 2.4).

Oracle	MySQL	SQL Server
GUI, SQL	SQL	GUI, SQL
C, C++, Java, Ruby, Objective C	C, C++, Java, Ruby, Objective C	Java, Ruby, Python, VB, .Net, PHP
Windows, Linux, Solaris, HP-UX, OS X, z/OS, AIX	Windows, Linux, OS X, FreeBSD, Solaris	Windows

Рисунок 2.4. – Порівняльна таблиця промислових СУБД

У порівнянні з MS SQL Server та ORACLE, MySQL призначений більше для проектів середньої величини. Мабуть самою популярною предметною областю для використання MySQL є сайтобудування.

Це пов'язано раніше всього з відкритістю коду та безкоштовним поширенням MySQL. Тим не менш, надійність і модність MySQL дозволяє будувати як корпоративні додатки, так і програми великого рівня.

Саме обсяги і надійність СУБД MySQL, а також можливість подальшого масштабування та забезпечили її вибір для реалізації поставленою в роботі завдання.

2.3 Особливості програмної реалізації

Насамперед перед написанням програмного коду необхідно побудувати діаграму компонентів, яка описує особливості фізичного та логічного представлення проектованої інформаційної системи. Діаграма компонентів програми дозволяє визначити внутрішню архітектуру додатку та будову інформаційної системи, встановивши залежності та зв'язки між програмними компонентами. У багатьох середовищах розробки найменування модуля чи компонента програми відповідає файлу, що підключається. Основними графічними елементами при побудові діаграми компонентів є компоненти, інтерфейси та залежності між ними.

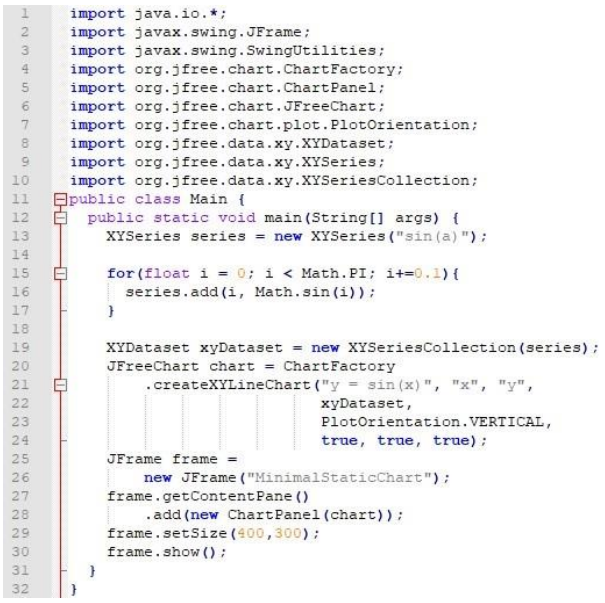
Як середовище розробки ПЗ обрано платформу IntelliJ IDEA Community Edition 2019.2.3 x64.

Рисунок 2.5. – Спрощене представлення архітектури системи

На самому початку визначено необхідний набір основних підключаємих модулів, таких як модулі для з'єднання з БД, модулі для побудови графічного інтерфейс (GUI).

Наступним етапом йде визначення списку необхідних спеціалізованих бібліотек. До таких бібліотек відносяться бібліотеки для побудови графіків та

звітів, для спеціалізованого взаємодії по API з операторами фіскальних даних та платіжними системами. На рисунку 2.6 представлений фрагмент вихідного коду, що відповідає за ініціалізацію користувацького інтерфейсу додатку.



```
import java.io.*;
import javax.swing.JFrame;
import javax.swing.SwingUtilities;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.xy.XYDataset;
import org.jfree.data.xy.XYSeries;
import org.jfree.data.xy.XYSeriesCollection;
public class Main {
  public static void main(String[] args) {
    XYSeries series = new XYSeries("sin(a)");

    for(float i = 0; i < Math.PI; i+=0.1){
      series.add(i, Math.sin(i));
    }

    XYDataset xyDataset = new XYSeriesCollection(series);
    JFreeChart chart = ChartFactory
        .createXYLineChart("y = sin(x)", "x", "y",
                           xyDataset,
                           PlotOrientation.VERTICAL,
                           true, true, true);
    JFrame frame =
        new JFrame("MinimalStaticChart");
    frame.getContentPane()
        .add(new ChartPanel(chart));
    frame.setSize(400,300);
    frame.show();
  }
}
```

Рисунок 2.6. – Фрагмент модуля, який ініціює створення користувацького інтерфейсу програми

Цей модуль імпортує необхідні для роботи модулі та бібліотеки, та створює вікно певного розміру. Його графічні елементи підвантажуються з файлів, а текстові значення з змінних.

Першим модулем розробляється «замовленння» (рисунок 2.7), який містить такі сутності та їх вміст, як номер кав'ярні, номер столика, статус броні, статус замовлення, склад замовлення, чек. Наприклад, для роботи модуля прийому замовлення необхідно створити масив (array), в якому ключем буде порядковий номер у замовленні, а значеннями – найменування товарної позиції, внутрішній id товару.

На поданому вище рисунку візуалізується діаграма компонентів модуля прийому замовлень. У ній встановлено чіткий зв'язок між рестораном Restaraunt та певним столиком Table у ньому, статусом резервації is_occuped, а також обслуговуючому його офіціанта, складі замовлення Order.

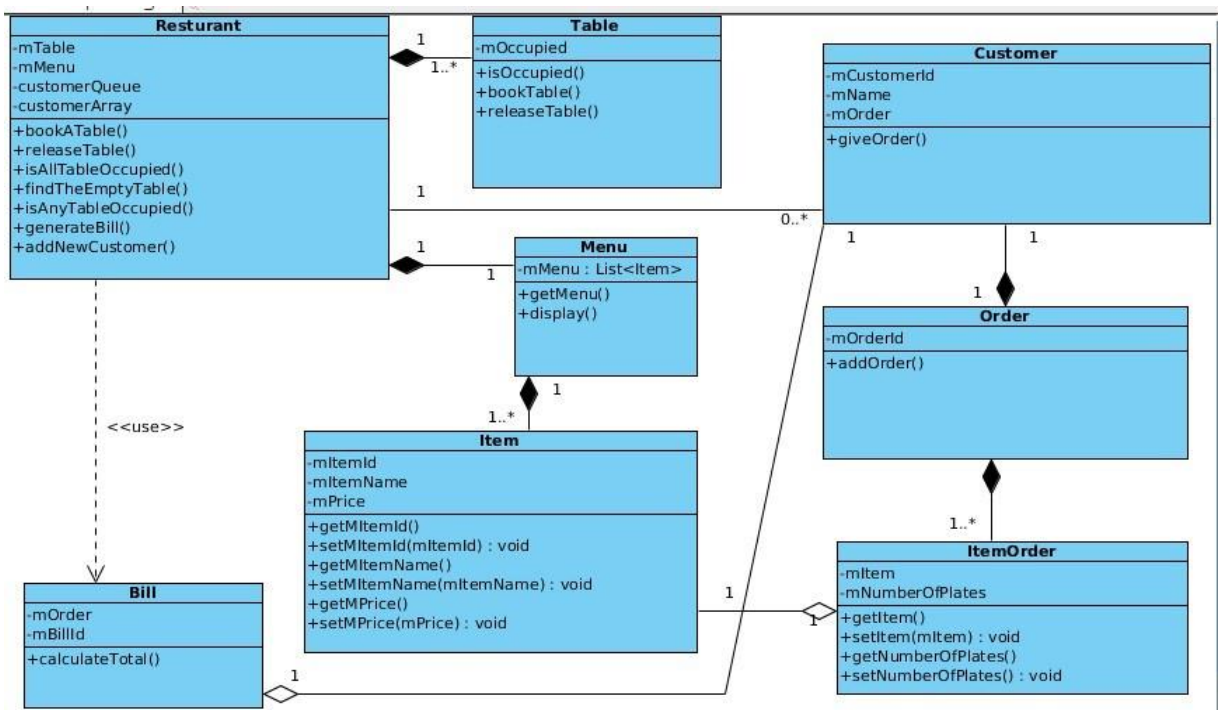


Рисунок 2.7. – Діаграма компонентів модуля «замовлень»

Цей модуль формує масив (array), за підсумками якого формується SQLзапит на додавання в БД нового запису про замовлення, його склад, тривалість виконання, а також відрахування з залишків сировини певної кількості найменувань.

Наступним модулем іде модуль перевірки залишків на віддаленому складі при їх наближенні до вичерпання. При оформленні замовлення відбувається перевірка наявності необхідного сировини на локальному складі кав'ярні. За його наявності замовлення передається у виробництво. Після виробництва, замовлення переходить до кінцевого споживача та ведеться його облік.

За відсутності у достатньому кількості необхідної сировини на локальному складі, замовлення не підтверджується. Після цього формується автоматична заявка на постачання з віддаленого складу.

```
1  package tgu_disser;
2  public class Customer{
3  private int mCustomerId;
4  private Order mOrder;
5  private String mCustomerName;
6  public Customer(){
7  mCustomerId = 0;
8  mOrder = new Order(mCustomerId);
9  }
10 public Customer(int customerId){
11 mCustomerId = customerId;
12 mOrder = new Order(customerId);
13 }
14 public int getCustomerId() {
15 return mCustomerId;
16 }
17 public void setCustomerId(int mCustomerId) {
18 this.mCustomerId = mCustomerId;
19 }
20 public void giveOrder(Item item, int numberOfPlates){
21
22 ItemOrder newItemOrder = new ItemOrder(item,  numberOfPlate
23 mOrder.addOrder(newItemOrder);
24 }
25 public void iAmDone(){
26 }
27 public Order getOrder() {
28 return mOrder;
29 }
30 public void setOrder(Order mOrder) {
31 this.mOrder = mOrder;
32 }
33 public String getCustomerName() {
34 return mCustomerName;
35 }
36 public void setCustomerName(String mCustomerName) {
37 this.mCustomerName = mCustomerName;
38 }
39 }
```

Рисунок 2.8. – Фрагмент програмного коду модуля «замовлень»

Після отримання заявки на віддалений склад, відбувається перевірка наявності сировини на ньому. За її наявності, воно поставляється до певної кав'ярні. За відсутності оформляється замовлення у постачальників.

Аналогічно система веде себе при наближенні залишків продукції до кінця. У цьому випадку всі необхідні дії щодо оформлення замовлень виробляються заздалегідь, без відмови у прийомі замовлення у клієнта.

На поданому рисунку діаграми класів (рисунок 2.9) описана архітектура класу на замовлення на віддаленому складі. Перевіряються залишки на віддаленому складі і за їх відсутності викликається клас замовлення у постачальників. При позитивному залишку, формується масив замовлення Order, який містить відомості про необхідні для постачання компонентів Item та

їх кількості total_price, а також відомості про конкретний підрозділ, приймаючий сировину.

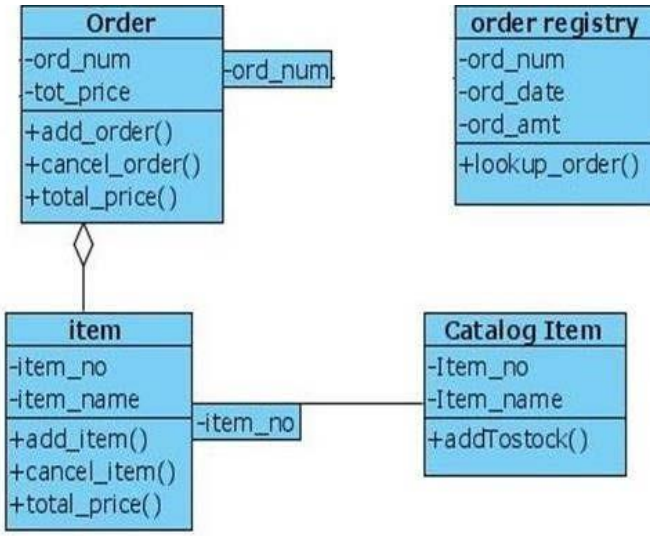


Рисунок 2.9. – Діаграма компонентів модуля «перевірка на локальному/віддаленому складі»

Отриманий масив перетворюється на запит і відправляється в базу, до якої відбувається реєстрація замовлення. За підсумками обробки клас order_registry формує запит до БД на присвоєння статусу виконаного.

```
1   public void setOrder(Order mOrder) {
2   this.mOrder = mOrder;
3   }
4   public float calculateTotal(){
5   float retValue = 0;
6   Iterator<ItemOrder> it = mCustomer.getOrder().getItemOrder().iterator();
7   while (it.hasNext() == true){
8   ItemOrder element = it.next();
9   retValue+= (element.getItem().getItemPrice())* (element.getNumberOfPlates());
10  }
11  return retValue;
12  }
13  public int getBillId() {
14  return mBillId;
15  }
16  public void setBillId(int mBillId) {
17  this.mBillId = mBillId;
18  }
19  }
20
21
22
```

Рисунок 2.10. – Фрагмент програмного коду модуля «перевірка залишків»

Дані з масиву вирушають запитом до БД для перевірки на наявність залишків зазначених товарів. У випадку позитивної відповіді БД, створюється новий запис у таблиці замовлень.

```
1   SELECT * from products
2   WHERE id=(id)
3   AND ostatok >= (к)
4
5   INSERT INTO Sales
6   VALUES (1,'Capucchino',1,1);
```

Рисунок 2.10. – Фрагмент запиту до бази даних на мові SQL

Отриманий відповідь від бази обробляється клієнтом, після чого формується діалогове вікно. Після отримання результату виконання запиту в користувацький інтерфейс виводиться результат його виконання у вигляді тексту. Текст зберігається у змінних типу String. Фрагменти вихідного коду основних модулів програми винесені у додаток Б.

2.4 Висновки до другого розділу

Зроблено аналіз засобів розробки програмної системи, описано основні засоби реалізації бази даних та основні проблемні моменти програмної реалізації.

РОЗДІЛ 3. ТЕСТУВАННЯ СИСТЕМИ УПРАВЛІННЯ МЕРЕЖЕЮ ЗАКЛАДІВ ГРОМАДСЬКОГО ХАРЧУВАННЯ

3.1 Тестування інформаційної системи

При тестуванні програмного продукту підтверджується робота бізнес правил, встановлених під час проектування програми та бази даних, а саме, здійснюється каскадне оновлення та видалення записів у таблицях бази даних, спроба додавання некоректних значень, реакція на помилки тощо.

Для проведення тестування інформаційної системи мережі громадського харчування розроблено програма та методика, представлені нижче.

Об'єктом тестування є програмне забезпечення (ПЗ) інформаційної системи мережі громадського харчування, а саме клієнтський модуль, що виконується на базі віртуальною машини Java та бази даних Mysql. Пред'явлене для тестування ПЗ повинно бути представлено у складі, достатньому для проведення повнофункціонального тестування відповідно до передбачених програмою та методикою тестування.

Метою тестування інформаційної системи мережі громадського харчування є:

- перевірка ПЗ розподіленої інформаційної системи мережі громадського харчування на відповідність затвердженому проекту розробки та впровадження системи;
- перевірка працездатності ПЗ інформаційної системи мережі громадського харчування та виявлення можливих помилок;
- перевірка якості інтерфейсу користувача ПЗ інформаційної системи мережі громадського харчування;
- перевірка якості інформаційного обміну між окремими модулями розподіленої інформаційної системи мережі громадського харчування;
- перевірка інформаційної безпеки.

Тестування складається з двох етапів – автоматичного та ручного.

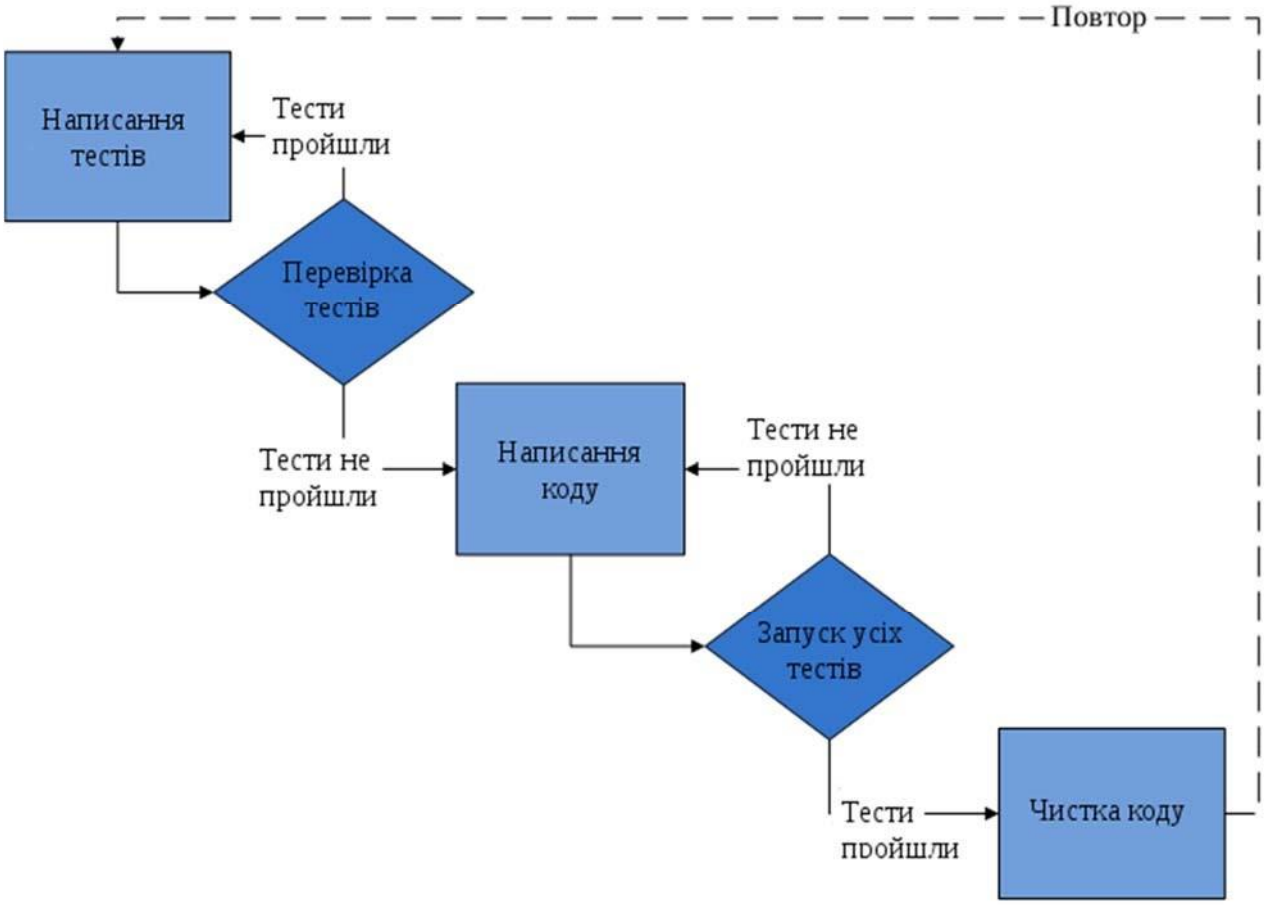


Рисунок 3.1. – Загальний алгоритм тестування

Автоматизоване тестування ПЗ - процес тестування, при якому ключові функції та кроки тесту як запуск, ініціалізація, виконання, аналіз та видача результату, виробляються автоматично спеціальним ПЗ за допомогою інструментів для автоматизованого тестування. Тестування інформаційної системи мережі громадського харчування проводиться за методом автоматизованого тестування з використанням програми jUnit.

Автоматизоване тестування засноване на використанні спеціального ПЗ для контролю виконання тестів та порівняння фактичних отриманих результатів із прогнозованими результатами. У процесі автоматизованого тестування використовуються спеціальний додаток - менеджер тестування.

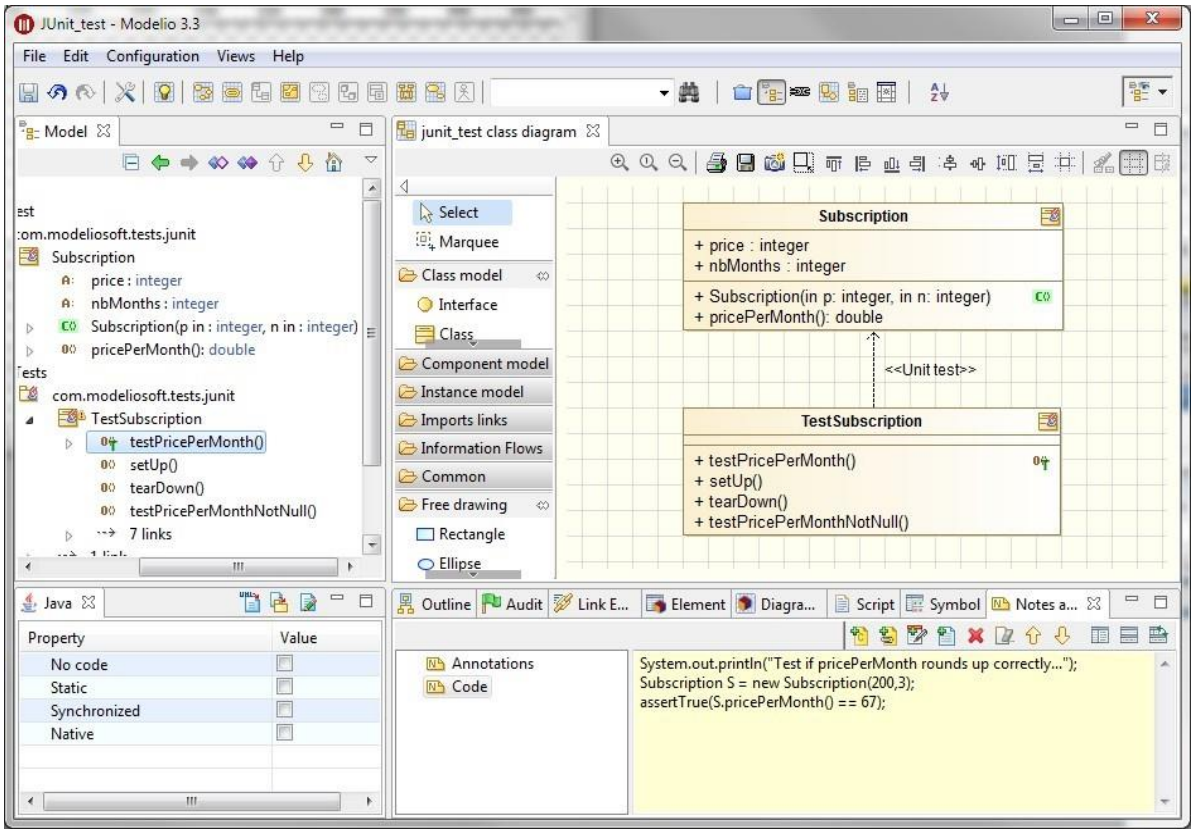


Рисунок 3.2. – Програма автоматичного тестування

За результатами автоматичного тестування були виявлено та виправлено деякі помилки, пов'язані з недостатньою фільтрацією даних при запитах до бази, таких як передача тексту (String) у полі, тип якого число (int) і т.д., що могло б призвести до падіння програми або її некоректним результатам роботи.

Навантажувальний тестування (load testing) оцінити поведінку системи при зростаючій навантаженні та великій кількості користувачів. Метою навантажувального тестування є визначення максимального навантаження, яке може витримати система.

Під час навантажувального тестування можуть здійснюватися операції, які дозволяють більш точно виміряти продуктивність та визначити «вузьке місце» системи:

- витрачається час за певної інтенсивності виконання цих операцій;
- визначення кількості клієнтів одночасно працюючих з додатком;

- визначення меж прийнятної продуктивності та швидкодії при збільшенні навантаження.

У ролі навантаження може виступати кількість користувачів (клієнтів), а також кількість операцій на сервері (рисунок 3.3).

Рисунок 3.3. – Динаміка подачі навантаження

За результатами діаграми видно, що навантаження на процесор та час відгуку системи задовольняє вимогам продуктивності при розрахунковому навантаженні 30 користувачів.

Ручне тестування проводиться через користувацький інтерфейс ПЗ. У ході ручного тестування здійснюється каскадне оновлення та видалення записів у таблицях бази даних, спроба додавання некоректних значень, реакція на помилки як поділ на нуль, спроба вступу літерних значень та поля, призначені тільки для чисел (integer) і навпаки, перевіряється робота всіх модулів програми та виконується порівняння їх результату роботи з очікуваним.

За результатами тестування складений зведений протокол, поданий у таблиці 3.1. До неї включено опис проведеного тесту, результат проведення тесту, а також за наявності – рекомендації щодо модифікації ПЗ. Умовою

допуску програми до експлуатації є тільки повне проходження тестів без критичних зауважень.

Таблиця 3.1 – Зведений протокол тестування ПЗ

№	Опис	Результат	Рекомендації
1	Перевірка на відповідність затвердженому проекту розробки та впровадження системи	відповідає	
2	Автоматична перевірка працездатності ПЗ та виявлення можливих помилок	відповідає	
3	Ручна перевірка працездатності ПЗ та виявлення можливих помилок	відповідає	
4	Перевірка якості інтерфейсу користувача ПЗ	відповідає	
5	Перевірка якості інформаційного обміну між окремими модулями ПЗ	відповідає	
6	Перевірка інформаційної безпеки	Істотних ризиків не виявлено	Використати хешування паролів (md5) у базі Використати зашифроване з'єднання (або VPN) для з'єднання з БД по відкритих каналах

Висновок за результатами тестування. В результаті проектування було розроблено додаток, діяльність якого у значній ступені полегшує роботу бариста, а також мінімізує помилки, що виникають під час роботи. Впровадження у мережу кав'ярень інформаційної системи призвело до зменшення витрат і витрат, спричинених необхідністю ручного перенесення даних з однієї інформаційної системи в іншу.

Створення розподіленою бази даних дозволило виконувати агреговані запити одночасно для всієї мережі, отримувати статистику в режимі реального часу.

Перехід на одну програму виключив необхідність виконання операцій імпорту-експорту в ручному режимі, з таких систем як інтернетбанк, системи бухгалтерської та податкової звітності, бази даних і т.д., що скоротило кількість помилок та витрати на ручну працю.

3.2 Розрахунок обсягу даних та навантаження на програмну систему

Під час розробки розподіленої інформаційної системи постає питання, що вимагають оцінки апаратних ресурсів – кількості серверів (віртуальних машин) та їх апаратні характеристики.

У цьому дослідженні пропускається дослідження апаратної платформи «клієнтів» та розраховуються тільки необхідні характеристики "сервера".

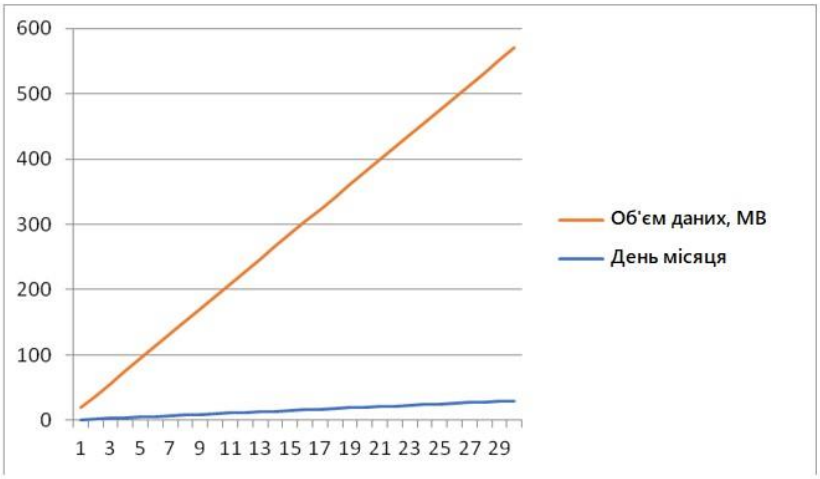


Рисунок 3.4. – Розрахунок місячного приросту обсягу даних для цілей вибору жорсткого диска

Базові показники:

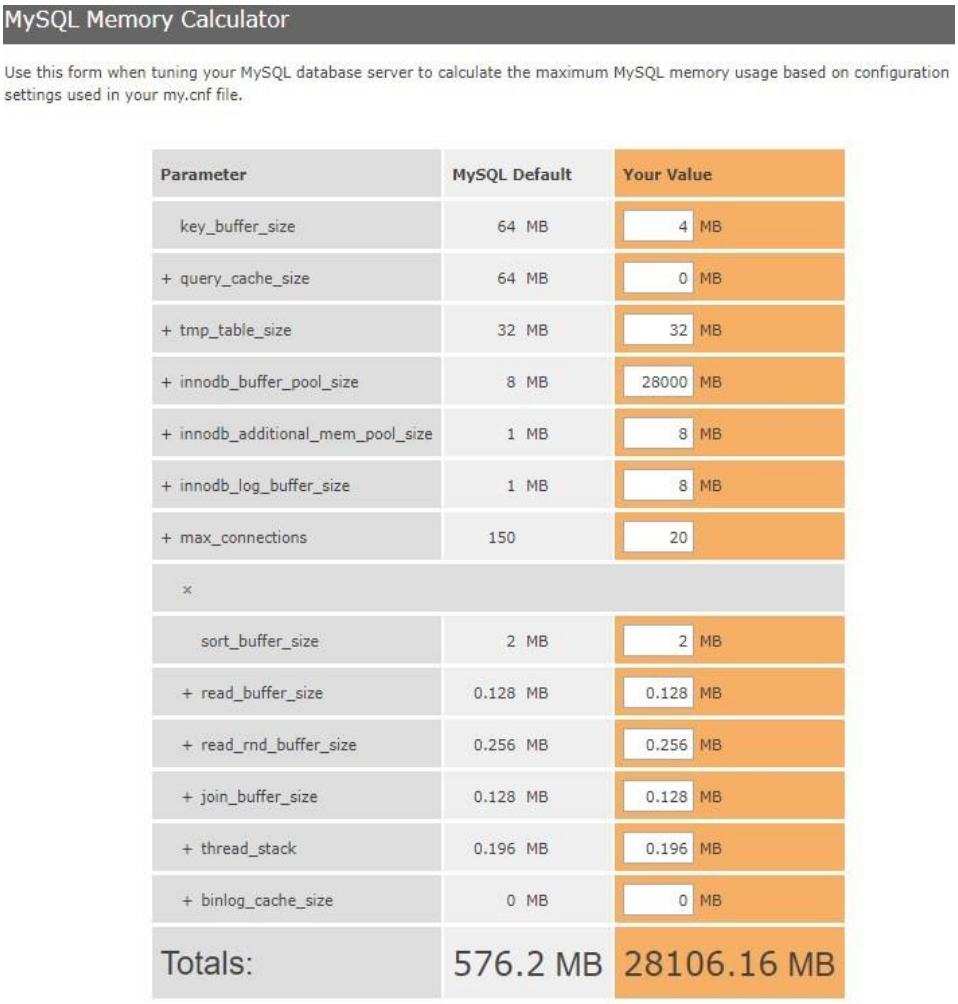
- кількість торгових точок: 11;
- умовна кількість клієнтів: 30-50;
- кількість одночасно працюючих клієнтів: <10 (пік за даними PhpMyadmin);

- звернень за хвилину: 37;
- середній об’єм даних, що генеруються за день: 18 МБ.

Кількість клієнтів, обсяг генерованих даних - змінна величина регульована при розрахунку/моделюванні.

Враховуючи місячний приріст обсягом трохи більше 600 МБ, при доборі апаратної конфігурації сервера можна вибрати жорсткий диск (SSD) самої мінімальної цінової категорії, оскільки його переповнення з даної динамікою трапиться пізніше, ніж пройде його термін експлуатації і він вийде з ладу через механічний знос.

На рисунку 3.5 виконаний розрахунок обсягу БД у пам'яті для сервера з 32 ГБ ОЗП.



MySQL Memory Calculator

Use this form when tuning your MySQL database server to calculate the maximum MySQL memory usage based on configuration settings used in your my.cnf file.

Parameter	MySQL Default	Your Value
key_buffer_size	64 MB	4 MB
+ query_cache_size	64 MB	0 MB
+ tmp_table_size	32 MB	32 MB
+ innodb_buffer_pool_size	8 MB	28000 MB
+ innodb_additional_mem_pool_size	1 MB	8 MB
+ innodb_log_buffer_size	1 MB	8 MB
+ max_connections	150	20
×		
sort_buffer_size	2 MB	2 MB
+ read_buffer_size	0.128 MB	0.128 MB
+ read_rnd_buffer_size	0.256 MB	0.256 MB
+ join_buffer_size	0.128 MB	0.128 MB
+ thread_stack	0.196 MB	0.196 MB
+ binlog_cache_size	0 MB	0 MB
Totals:	576.2 MB	28106.16 MB

Рисунок 3.5. – Розрахунок необхідної кількості оперативною пам'яті

При цьому розмір буфера InnoDB встановлений у 28 ГБ відповідно до рекомендації виділення до 80% ОЗП на сервері, який виконує тільки роль сервера БД і не містить інших додатків, що створюють навантаження. Виділення такого обсягу пам'яті дозволить кешувати та розмістити всю БД в ОЗУ, що багаторазово прискорить роботу з базою, у тому числі виконання складних запитів, типу COUNT(*). За відсутності кешування бази, під час виконання подібного запиту, наприклад, для підрахунку залишків, високе навантаження може паралізувати можливість виконання інших запитів, обробка яких потрібно в оперативному режимі, наприклад, на оформлення замовлень.

Узагальнюючи вищенаведені розрахунки, як апаратної платформи для сервера БД рекомендовано вибір сервера економ класу, оскільки враховуючи малу кількість одночасно працюючих клієнтів та запитів відсутні високі прогнозовані навантаження.

3.3 Інструкція користувача

Якщо користувач увійшов як адміністратор, він отримує форму для входу (рисунок 3.6) та проходження авторизації. Якщо пройдено, то він може увійти та отримати опцію адміністратора. Тоді він може додавати співробітника, а також видаляти. Він також може оновити свою інформацію (рисунок 3.7).

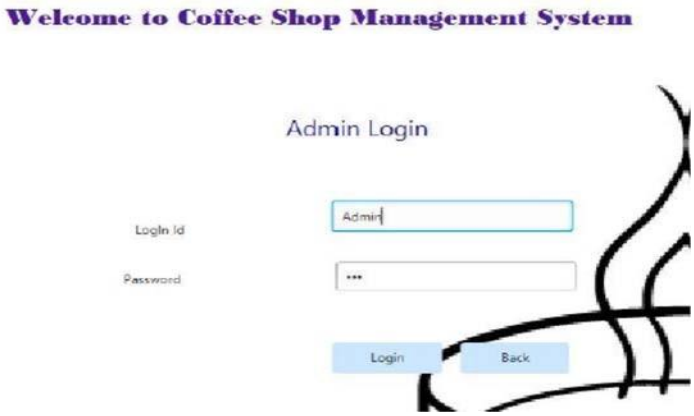


Рисунок 3.6. – Форма для входу в систему

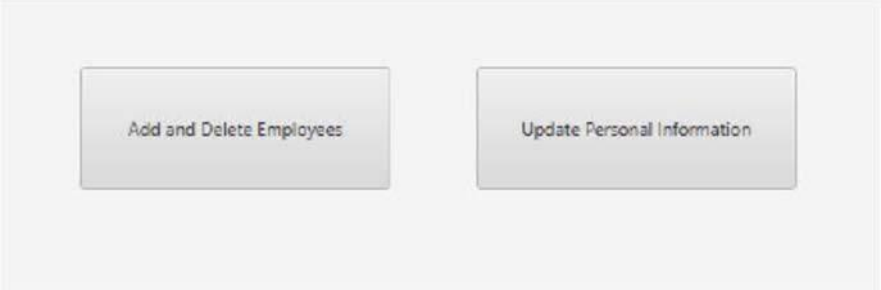


Рисунок 3.7. – Форма для оновлення інформації про працівників

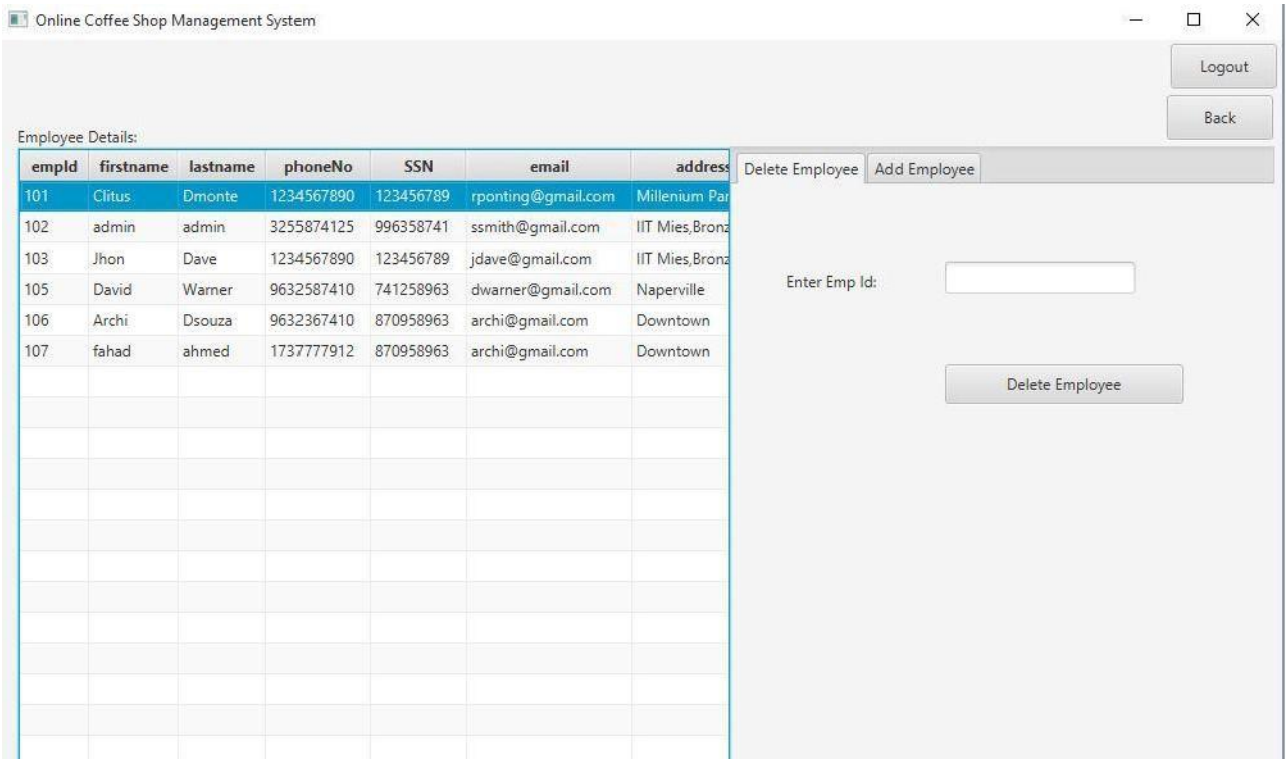


Рисунок 3.8. –. Форма для редагування інформації про працівників

Якщо користувач увійшов як клієнт, він має два варіанти реєстрації та входу. Якщо ця авторизація в порядку, то отримати дозвіл на вхід і отримати вітальне повідомлення від кав'ярні. Потім він також може оновити свій профіль (оновити та видалити). Якщо він може натиснути оновити опцію, щоб оновити свій профіль (рисунок 3.9), тоді він може отримати замовлення та оплатити, а потім підтвердити (рисунок 3.10).

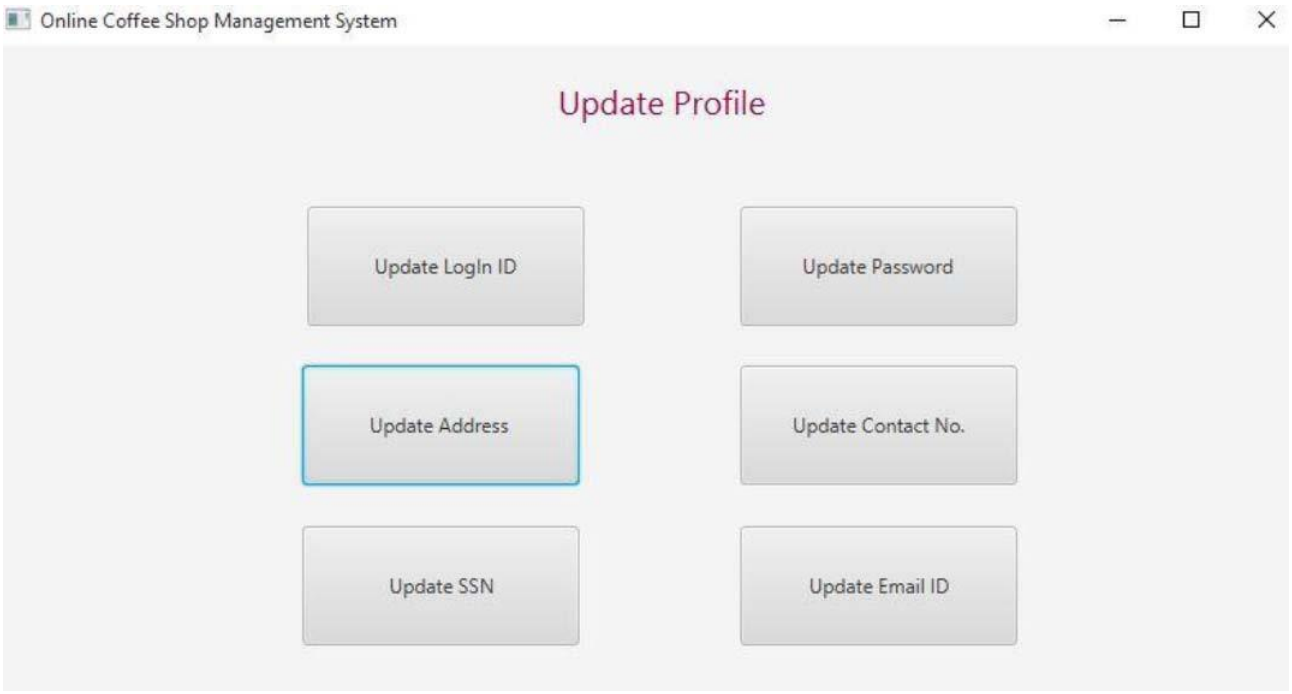


Рисунок 3.9. – Форма для оновлення інформації про клієнтів

На рисунку 3.11 представлено форму для підтвердження інформації про замовлення.

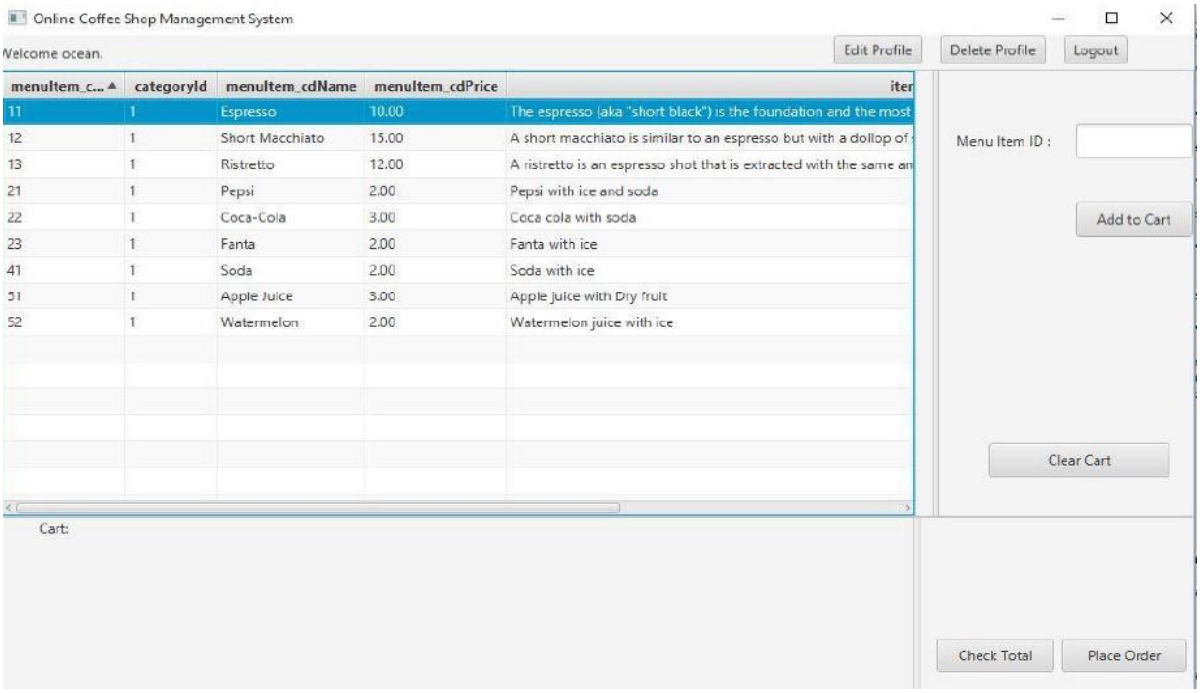


Рисунок 3.10. – Форма для оновлення інформації про замовлення

Рисунок 3.11. – Форма для підтвердження замовлення

Якщо користувачв увійшов як адміністратор, з’явиться вікно з отриманими трьома варіантами: можливість перегляду усіх замовлення за день і перевірки суми готівки для продажу, тоді він може додати елемент або видалити, і оновити товар. Нарешті він також може оновити свою інформацію.

Рисунок 3.12. – Форма для перегляду всіх замовлень

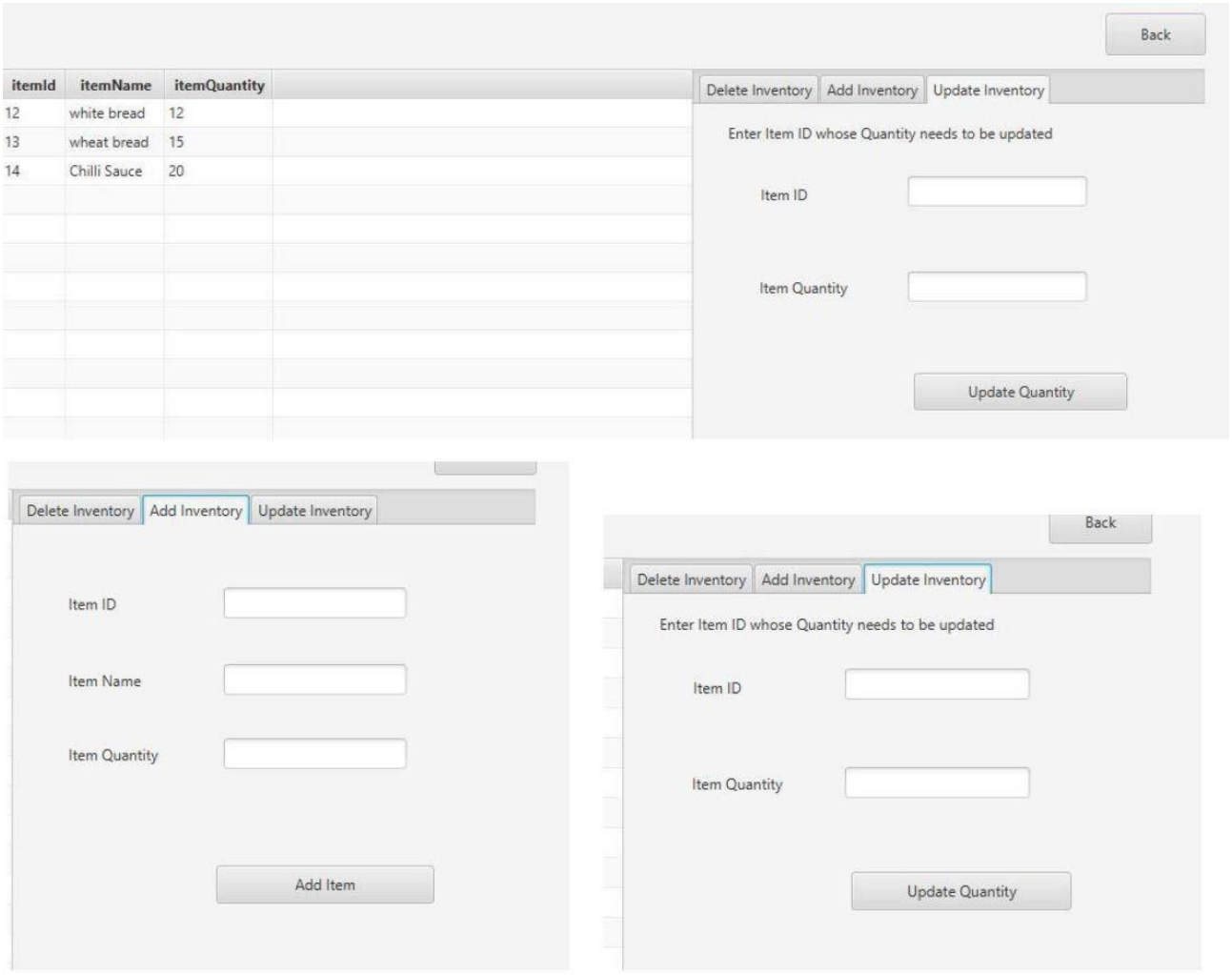


Рисунок 3.13. – Форми для оновлення замовлень

Рисунок 3.14. – Форма для оновлення профілів та основної інформації

Якщо користувач увійшов як кур'єр, він отримає два варіанти. Він може перевірити доставку товару клієнту та оновити свою інформацію (рисунок 3.15).

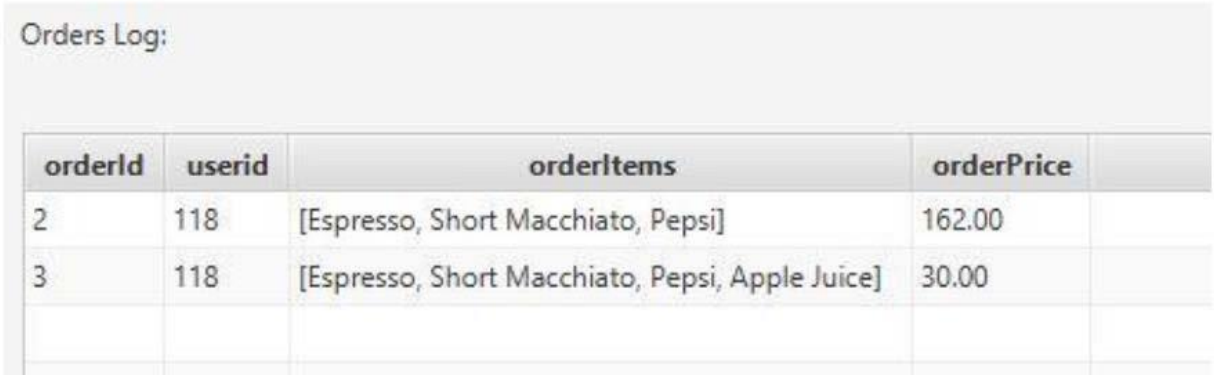


Рисунок 3.15. – Форма для перегляду доставки

3.4 Висновки до третього розділу

Інтерфейс системи є інтуїтивно зрозумілим, одноманітним для легшого та комфортнішого сприйняття. Дослідженуваний додаток може працювати тільки на стаціонарному ПК, за наявності встановленої віртуальної машини Java.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Порядок надання домедичної допомоги постраждалим при раптовій зупинці серця

Домедична допомога у разі зупинки серця є надзвичайно важливим заходом, адже за 5-6 хвилин без кровопостачання кора головного мозку починає зазнавати на собі дію процесів, що мають незворотній характер.

Навички першої допомоги – критична необхідність для кожної людини, особливо в умовах війни. Сьогодні це без перебільшень може врятувати життя.

Послідовність дій при наданні домедичної допомоги дорослим при раптовій зупинці кровообігу за відсутності автоматичного зовнішнього дефібрилятора:

1) перед наданням допомоги необхідно переконатися у відсутності небезпеки та за її відсутності перейти до наступного кроку;

2) визначити наявність свідомості – обережно потрясти дорослого за плече та голосно звернутися до нього, наприклад, «З Вами все гаразд? Вам потрібна допомога?»

3) якщо дорослий реагує:

а) залишити його у попередньому положенні, якщо йому нічого не загрожує;

б) з’ясувати характер події, що сталася;

в) здійснити виклик екстреної медичної допомоги та дотримуватись вказівок диспетчера прийому виклику;

г) за необхідності надати дорослому зручного положення;

ґ) забезпечити нагляд за дорослим до приїзду бригади екстреної (швидкої) медичної допомоги;

4) якщо дорослий не реагує:

а) звернутися до осіб, які поряд, за допомогою. Якщо випадкових свідків декілька, слід звертатися до конкретної особи;

б) якщо дорослий лежить на животі, повернути його на спину та відновити прохідність дихальних шляхів.

Якщо стан дорослого пов'язаний з отриманням травми, наприклад падіння з висоти, вважати, що у нього є травма в шийному відділі хребта. В такому випадку слід максимально обмежити рухи в шийному відділі хребта;

в) відновити прохідність дихальних шляхів, визначити наявність дихання за допомогою прийому: «чути, бачити, відчувати». Наявність дихання визначати до 10 секунд. Якщо виникли сумніви чи є дихання, або воно ненормальне, вважати, що дихання відсутнє;

5) якщо дорослий дихає нормально, при відсутності свідомості слід

перевести його у стабільне бокове положення, здійснити виклик екстреної медичної допомоги та перевіряти кожні 3-5 хвилин дихання до моменту приїзду бригади екстреної (швидкої) медичної допомоги. Якщо є настороженість щодо травми потрібно уникати повороту в стабільне бокове положення, а забезпечити прохідність дихальних шляхів методом висування нижньої щелепи до приїзду бригади екстреної (швидкої) медичної допомоги;

6) якщо дихання відсутнє:

а) здійснити виклик екстреної медичної допомоги та дотримуватись вказівок диспетчера прийому виклику. Якщо є інші випадкові свідки, слід сказати їм здійснити виклик екстреної медичної допомоги та негайно перейти до наступного кроку. При наявності гучного зв'язку на телефоні слід здійснювати виклик екстреної медичної допомоги та одночасно проводити серцево-легеневу реанімацію;

б) розпочати проведення серцево-легеневої реанімації: виконати 30 натискань на середину грудної клітки глибиною не менше 5 см (не більше 6 см), з частотою 100 натискань (не більше 120) за хвилину; виконати 2 вдихи з використанням маски-клапану, дихальної маски тощо. При відсутності захисних

засобів можна не виконувати штучне дихання, а проводити тільки натискання на грудну клітку. Виконання двох штучних вдихів повинно тривати не більше 5 секунд; після двох вдихів продовжити натискання на грудну клітку відповідно до наведених рекомендацій у цьому підпункті; не слід переривати натиснення на грудну клітку дорослому більше ніж на 10 секунд;

7) змінювати особу, що проводить натискання нагрудну клітку, кожні 2 хвилини.

8) припинити проведення серцево-легеневої реанімації до прибуття бригадиї екстреної (швидкої) медичної допомоги за наступних умов:

- при появі у дорослого явних ознак життя; відновлення самостійного нормального дихання, координованої рухової активності, відкривання очей;
- виникненні загрози життю рятівнику та/або дорослому;
- неможливості проведення серцево-легеневої реанімації внаслідок значного фізичного виснаження.

Послідовність дій при наданні домедичної допомоги дорослим при раптовій

зупинці кровообігу та наявності автоматичного зовнішнього дефібрилятора:

1) перед наданням допомоги переконатися у відсутності небезпеки та за її відсутності перейти до наступного кроку;

2) визначити наявність свідомості – обережно потрясти дорослого за плече та голосно звернутися до нього, наприклад, «З Вами все гаразд? Вам потрібна допомога?»;

3) якщо дорослий не реагує:

а) звернутися до осіб, які поряд, за допомогою. Якщо випадкових свідків декілька, слід звертатися до конкретної особи;

б) якщо дорослий лежить на животі, повернути його на спину та відновити прохідність дихальних шляхів. Якщо стан дорослого пов’язаний з отриманням травми, наприклад падіння з висоти, вважати, що у нього є травма в

шийному відділі хребта. В такому випадку слід максимально обмежити рухи в шийному відділі хребта;

в) відновити прохідність дихальних шляхів, визначити наявність дихання за допомогою прийому: «чути, бачити, відчувати». Наявність дихання визначати до 10 секунд. Якщо виникли сумніви, чи є дихання, або воно ненормальне, вважати, що дихання відсутнє;

г) здійснити виклик екстреної медичної допомоги та дотримуватись вказівок диспетчера прийому виклику. Якщо є інші випадкові свідки, слід сказати їм здійснити виклик екстреної медичної допомоги та негайно перейти до наступного кроку. При наявності гучного зв'язку на телефоні слід здійснювати виклик екстреної медичної допомоги та одночасно проводити серцево-легеневу реанімацію. Якщо рятувальник один, і не відомо, що автоматичний зовнішній дефібрилятор знаходиться у безпосередній близькості від місця події, не слід залишати дорослого та шукати автоматичний зовнішній дефібрилятор – в такому випадку слід проводити серцево-легеневу реанімацію як передбачено у пункті 3 цього Порядку;

ґ) розпочати проведення серцево-легеневої реанімації: виконати 30 натискань на середину грудної клітки глибиною не менше 5 см (не більше 6 см), з частотою 100 натискань (не більше 120) за хвилину; виконати 2 вдихи рекомендовано з використанням маски-клапану, дихальної маски тощо. При відсутності захисних засобів можна не виконувати штучне дихання, а проводити тільки натискання на грудну клітку. Виконання двох штучних вдихів повинно тривати не більше 5 секунд; після двох вдихів продовжити натискання на грудну клітку відповідно до наведених рекомендацій у цьому підпункті; не слід переривати натискання на грудну клітку дорослому більше ніж на 10 секунд;

4) змінювати особу, що проводить натискання на грудну клітку, кожні 2 хвилини. У випадку якщо особа, яка проводить натискання на грудну клітку, відчуває виснаження, заміну слід виконати раніше, ніж через 2 хвилини;

5) як тільки автоматичний зовнішній дефібрилятор наявний на місці події, слід негайно: увімкнути пристрій та чітко дотримуватись голосових вказівок; якщо осіб які надають допомогу декілька, вмикання автоматичного зовнішнього дефібрилятора та приклеювання електродів до грудної клітки дорослого слід одночасно з проведенням компресій на грудну клітку; у випадку необхідності проведення дефібриляції, прослідкувати, щоб ніхто не торкався дорослого; після проведення дефібриляції слід негайно розпочати натискання на грудну клітку;

6) припинити проведення серцево-легеневої реанімації до прибуття бригади екстреної (швидкої) медичної допомоги за наступних умов: при появі у дорослого явних ознак життя; відновлення самостійного дихання, координованої рухової активності; виникненні загрози життю рятівнику та/або дорослому.

У випадку появи явних ознак життя до приїзду бригади екстреної (швидкої) медичної допомоги, електроди від автоматичного зовнішнього дефібрилятора слід залишити на грудній клітці дорослого.

4.2 Планування робіт щодо охорони праці

У плануванні робіт з охорони праці як вихідні дані використовують:

- план економічного і соціального розвитку підприємства;
- план організаційно-технічних, технологічних та інших заходів, направлених на підвищення ефективності, пошук резервів підприємства;
- матеріали паспортизації цехів, дільниць, робочих місць на відповідність їх вимогам охорони праці; а також атестації робочих місць за умовами праці;
- комплексні заходи щодо досягнення встановлених нормативів безпеки, гігієни праці і виробничої санітарії;

- матеріали розслідувань нещасних випадків, професійних захворювань та аварій;
- приписи органів державного нагляду, служб охорони праці;
- рекомендації комісії з питань охорони праці підприємства; пропозиції уповноважених трудового колективу з питань охорони праці;
- результати адміністративно-громадського контролю стану охорони праці;
- постанови профспілкового комітету, рішення зборів (конференцій) трудового колективу з питань охорони праці;
- відповідні накази адміністрації, документи вищестоящих господарських і профспілкових органів.

Важливе значення у системі планування робіт щодо забезпечення безпеки праці на підприємствах має розробка розділу «Охорона праці» колективного договору (угоди, трудового договору), що укладається між власником і трудовим колективом, за рішенням якого його інтереси може представляти також і профспілка.

Одним з найважливіших розділів колективного договору є інженерно-технічні заходи задля досягнення нормативів безпеки, гігієни праці і виробничого середовища, підвищення наявного рівня охорони праці. До цільових заходів належать, наприклад, розробка, виготовлення і установка нових, ефективніших інженерно-технічних засобів охорони праці, огорож, засобів сигналізації і контролю, запобіжних пристроїв тощо.

Заходи з охорони праці мають бути забезпечені проектно-кошторисно конструкторською та іншою технічною документацією, фінансуванням і матеріальними ресурсами. Згідно зі статтею 19 закону України «Про охорону праці», фінансування охорони праці на підприємстві здійснюється власником.

На підприємствах, в галузях і на державному рівні створюються фонди охорони праці. Такі ж фонди створюються органами місцевого і регіонального самоврядування для потреб регіону. В державний, галузеві та регіональні фонди

охорони праці направляються, разом із коштами державного і місцевого бюджетів, відрахування підприємств (в розмірі не менше 0,5% від об'єму реалізації продукції, виконаних робіт, наданих послуг, а для бюджетних підприємств – не менше 0,2% від фонду оплати праці) та іншими надходженнями, кошти, отримані від застосування органами державного нагляду штрафних санкцій до підприємств, а також кошти від штрафування фізичних осіб, винних у порушеннях вимог охорони паці.

На підприємствах фінансування заходів з охорони праці може здійснюватись, зазвичай за рахунок:

- виробничих витрат (матеріальні витрати на удосконалення технології та організації виробництва, підтримку основних виробничих фондів у робочому стані, утримання засобів колективного захисту та ін.);
- амортизації основних фондів;
- капітальних вкладень;
- банківського кредиту;
- кошторису затрат бюджетних організацій і установ; · фонду охорони праці.

Засоби фонду охорони праці підприємства повинні використовуватися тільки на виконання комплексних заходів, спрямованих на досягнення встановлених нормативів з охорони праці, подальше підвищення рівня охорони праці в умовах діючого виробництва.

До цих заходів, які є основою для розробки колективного договору, належать:

- атестація робочих місць на відповідність їх вимогам нормативних актів про охорону праці;
- навчання працівників з питань охорони праці;
- забезпечення працівників спецодягом, спецвзуттям та іншими засобами індивідуального захисту;

- проведення обов’язкових медичних оглядів працівників; · проведення експертизи технічного стану будівель і споруд;
- приведення рівня шуму, вібрації, ультразвуку, іонізуючих та інших шкідливих випромінювань на робочих місцях у відповідність вимогам чинних нормативних актів;
- розробка, виготовлення і монтаж нових або реконструкція існуючих вентиляційних систем, аспіраційних, порохо- і газоуловлювальних пристроїв для приведення у відповідність вимогам чинних нормативних актів про охорону праці та їх паспортизація тощо.

Першочерговість виконання заходів встановлюється на основі атестації робочих місць на відповідність їх вимогам нормативних актів з охорони праці, аналізу причин нещасних випадків на виробництві і професійних захворювань з урахуванням їх економічної і соціальної значимості.

Засоби фонду охорони праці підприємства не повинні витрачатися на ремонтні та інші роботи, пов’язані з підтриманням в належному технічному стані основних фондів (включаючи інженерно-технічні засоби безпеки, засоби колективного захисту працівників), на надання пільг і компенсацій працівникам, на природоохоронні заходи.

Засоби галузевих і регіональних фондів охорони праці підприємствам направляються згідно з затвердженими кошторисами витрат на фінансування заходів, передбачених відповідно галузевими і регіональними програмами покращення стану безпеки, гігієни праці та виробничої санітарії.

До заходів, які здійснюються за рахунок коштів галузевого фонду охорони праці, належать, наприклад:

- створення і впровадження автоматизованих систем попередження аварій, пожеж, нещасних випадків на виробництві, а також систем управління охороною праці;
- розробка і виконання програм метрологічного, кадрового і організаційного забезпечення атестації робочих місць;

- нормативно-правове забезпечення охорони праці;
- проведення і впровадження у виробництво науково-дослідних, проектноконструкторських робіт з безпеки і гігієни праці.

За рахунок регіонального фонду охорони праці виконуються, наприклад, такі заходи :

- приведення у відповідність з вимогами нормативних актів приміщень для зберігання отруйних, пожежо- і вибухонебезпечних речовин;
- утилізація небезпечних і шкідливих відходів виробництва;
- забезпечення безпечної експлуатації об’єктів з постійним перебуванням великої кількості людей (магазинів, навчальних закладів та ін.).

Керівництво державним фондом охорони праці здійснює Державний комітету країни з нагляду за охороною праці (Держгірпромнагляд). Кошти цього фонду відповідно до кошторису, затвердженого Держгірпромнаглядом за узгодженням з Мінфіном, направляються на фінансування заходів комплексного вирішення завдань з охорони праці, передбачених національною програмою покращення стану безпеки, гігієни праці і виробничого середовища, проведення науководослідних і проектно-конструкторських робіт в межах цієї програми, сприяння становленню і розвитку спеціалізованих підприємств і виробництв, науковотехнічних центрів і експертних груп та здійснення цих заходів.

Таким чином, на Україні створена і функціонує єдина система планування і фінансування заходів з охорони праці: на підприємствах і в галузях, на регіональному і державному рівнях.

4.3 Висновки до четвертого розділу

В даному розділі кваліфікаційної роботи розкрито питання надання домедичної допомоги при раптовій зупинці серця, систематизовано чітку послідовність дій за рахунок швидкого розпізнавання стану, виклику екстрених

служб, проведення СЛР та використання дефібрилятора (за наявності). Це критично важливі знання для збереження життя.

Планування робіт з охорони праці повинно забезпечувати створення безпечних умов праці, передбачати запобіжні заходи, навчання персоналу, контроль ризиків та відповідність нормативним актам. Системний підхід зменшує кількість травм і нещасних випадків на виробництві.

ВИСНОВКИ

Перед розробкою було проведено дослідження предметної області, виконані електронні діаграми бізнес-процесів, а так само виконані моделі інформаційно-логічної моделі бази даних.

Підсумковим етапом розробки було проведення автоматичного та ручного тестування роботи додатку на предмет некоректного введення даних користувачем у додаток та коректну роботу бізнес-правил з каскадного видалення та каскадного оновлення даних у таблицях бази даних.

Можна, можливо відзначити, що програма вийшла універсальною і може функціонувати у будь-якій подібній організації громадського харчування з будьяким асортиментом продукції.

Для розробки клієнтського модуля був обраний об'єктно-орієнтована мова високого рівня Java, що містить у собі можливості обробки бази даних, які можна порівняти з можливостями СУБД. А широкий асортимент методів та моделей доступу до даних дозволив спроектувати програми максимально комфортним способом.

Додаток використовує концепцію JDBC для доступу до даних. Це абстрактна модель для доступу до локальних та віддалених даних. Її застосування дає можливість із відносною легкістю поширювати створений програмний продукт на інші персональні комп'ютери, оскільки немає необхідності замислюватися про те, що б на комп'ютері користувача було встановлено драйвер доступу до даних.

Рамками даного проекту та обмежений час не дозволили виконати інтеграцію створеного програми з іншими інформаційними системами, але такі можливості цілком можливо реалізувати за допомогою API.

Інтерфейс програмною оболонки представляє з себе стандартний GUI-інтерфейс у стилі багатодокументного додатку, як головна форма виступає форма запуску, а дочірні відкриваються у її рамках.

Створена в ході роботи інформаційна система призначена для встановлення на настільні ПК під керуванням ОС Windows 10. В силу кросплатформенності платформи Java, можлива експлуатація зазначеної ІС та інших операційних системах в режимі обмеженої функціональності.

У системі можна, можливо досить легко реєструвати нові продажі, а також проводити редагування введеною інформації до бази даних.

Програма має зручні інструменти фільтрації даних. Дані можна, можливо фільтрувати по частковому збігу, що дозволяє користувачу, не знаючи, наприклад, точного найменування, ввести частину назви та система видасть йому результат пошуку. Відібрані дані можна, у разі необхідності редагувати.

ПЕРЕЛІК ДЖЕРЕЛ

1. Web Architecture 101 [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/storyblocks-engineering/web-architecture-101-a3224e126947.

2. Static Vs Dynamic Websites [Електронний ресурс] – Режим доступу до ресурсу: https://www.wix.com/blog/2021/11/static-vs-dynamic-website/.

3. A Guide to Different Types of Website Structures [Електронний ресурс] – Режим доступу до ресурсу: https://xd.adobe.com/ideas/process/informationarchitecture/different-types-of-website-structures.

4. Лаврищева К.М. Програмна інженерія / Лаврищева К.М. –К: Академперіодика, 2008. – 319с.

5. Харченко О.Г. Інструментальний засіб порівняльного оцінювання і багатокритеріального вибору архітектури програмних систем / О.Г.Харченко, І.О.Боднарчук, І.Е.Райчев, І.О.Галай // Інженерія програмного забезпечення. – 2015. –№1(21). – С. 10–24.

6. Харченко, О., Яцишин, В., & Боднарчук, І. (2013). Експертна система проектування архітектури програмного забезпечення. Комп’ютерні Технології Друкарства, (29), 10–26.

7. Матійчук, Л., Готович, В., & Бонар, В. (2025). Порівняння ефективності методів некерованого машинного навчання для виявлення аномалій в obd2 даних. Measuring and computing devices in technological processes, (1), 407-414.

8. Харченко О.Г. Службовий твір Комп’ютерна програма “Архітектор програмних систем” / О.Г.Харченко, І.Е.Райчев, О.А.Щербак, Б.С.Павленко, І.О.Боднарчук // Свідоцтво про реєстрацію авторського права на твір №59631. Видане державною службою інтелектуальної власності України 13.05.2015, м.Київ, НАУ.

9. Боднарчук, І., Харченко, О., Хоміцький, Б., & Шимчук, Г. (2019). Проектування архітектури програмних систем в проектах з гнучкими методами управління. Матеріали XXI Наукової Конференції Тернопільського Національного Технічного Університету Імені Івана Пулюя, 46–48.

10. Харченко, О. Г., Галай, І. О., Бондарчук, І. О., & Яцишин, В. В. (2010). Проектування архітектури WEB-застосування на основі моделі якості проектування. Інженерія Програмного Забезпечення, 4(4), 26.

11. Кириченко А. В. Динамічні сайти на HTML, CSS, JavaScript та Bootstrap. Практика, практика та лише практика. / А. В. Кириченко, Є. В. Дубовик.Наука та техніка, 2018. – 272 с.

12. Матійчук Л., Готович В., Бонар В. Порівняння ефективності методів некерованого машинного навчання для виявлення аномалій в OBD2 даних. Вимірювальна та обчислювальна техніка в технологічних процесах. Хмельницький національний університет. (1), 2025. С. 407–414.

13. Коноваленко І. В. Платформа .NET та мова програмування C# 8.0 : навч. посіб. / І. В. Коноваленко, П. О. Марущак. – Тернопіль : В. А. Паляниця, 2020 – 320 с.

14. Комп'ютерні мережі: [навчальний посібник] / А. Г. Микитишин, М. М. Митник, П. Д. Стухляк, В. В. Пасічник.– Львів: «Магнолія 2006», 2013– 256с.

15. Микитишин А. Г., Митник М. М., Стухляк П. Д., Пасічник В. В. Комп’ютерні мережі. Книга 1 [навчальний посібник]. Львів : «Магнолія 2006», 2013. 256 с.

16. Моргунов Е. П. PostgreSQL. Основи мови SQL: [навч. посіб.] [Електронний ресурс]

17. Придатко О. В., Бурак Н. Є., Дзень В. Є., Куninець М. С. Адаптивна інформаційно-довідкова система "UniBell" як складова частина проєкту "Smartуніверситет". Науковий вісник НЛТУ України. 2020, т. 30, № 5. С. 105–113.

18. Литвин В.А. Принципи створення сучасних веб-сайтів. – Одеса: Астропринт, 2019 р. с. 234.

19. Мороз Ю.Б. Захист інформації в комп'ютерних системах та мережах. – Київ: Наукова думка, 2021 р. с. 289.

20. Google Developers. Progressive Web Apps. [Електронний ресурс] – Режим доступу до ресурсу: https://developers.google.com/web/progressive-web-apps.Strutynska, I. (2019). Метрики цифрової трансформації бізнесу: світові та вітчизняні реалії. Галицький економічний вісник, 61(6), 30–45.

21. Бойко І.В., М.Р. Петрик, Г.Б. Цуприк. Інформаційні технології видобутку даних (Data mining, високопродуктивні обчислення у складних системах): навчальний посібник. Тернопіль: : ТНТУ 2020 – 62 с.

22. Strutynska, I. V. (2018). Informatsiini tekhnolohii orhanizatsii biznesu–imperatyv innovatsiinoho rozvytku biznes-struktur [Information technologies of business organization an imperative of innovative development of business structures]. Galician Economic Journal, 55(2), 40–49.

23. Конспект лекцій з курсу «Програмування для мобільних пристроїв» для здобувачів освітнього ступеня «бакалавр», спеціальності 126 «Інформаційні системи та технології», денної форми навчання. Укладачі: Готович В.А., Михайлович Т. В. ТНТУ, 2020 р.

24. Системи баз даних та знань. Книга 1.Організація баз даних та знань: [навч. посіб.] / А. Ю. Берко, О. М. Верес, В. В. Пасічник. – Львів: «Магнолія2006», 2019. – 584 с.

25. Якість та тестування ПЗ. URL: https://dl.tntu.edu.ua/index.php.

26. Firebase Test Lab [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/docs/test-lab.

27. What is automation testing? URL: https://www.globalapptesting.com/blog/what-is-automation-testing.

28. Кодекс цивільного захисту України [Електронний ресурс] – Режим доступу: https://zakon.rada.gov.ua/laws/show/5403-17#Text

29. Про затвердження Положення про підсистему реагування на надзвичайні ситуації, проведення аварійно-рятувальних та інших невідкладних робіт єдиної державної системи цивільного захисту : наказ Міністерства внутрішніх справ України від 04.05.2016 №356.

30. Про затвердження Положення про штаб з ліквідації наслідків надзвичайної ситуації та видів оперативно-технічної і звітної документації штабу з ліквідації наслідків надзвичайної ситуації : наказ Міністерства внутрішніх справ України від 26.12.2014 №1406.

31. Про організацію роботи штабу з ліквідації наслідків надзвичайної ситуації та забезпечення його готовності : наказ Державної служби України з надзвичайних ситуацій від 16.03.2015 №149.

32. Рекомендації для роботодавців щодо організації виконання робіт підвищеної небезпеки під час воєнних (бойових) дій [Електронний ресурс] – Режим доступу: https://dsp.gov.ua/faq/rekomendatsii-dlia-robotodavtsiv-shchodo-orhanizatsiivykonannia-robit-pidvyshchenoi-nebezpeky-pid-chas-voiennykh-boiovykh-dii/

ДОДАТКИ

Додаток А

Лістинг скрипта імпорту структури бази даних

```
USE ` database` ;
-- ----------------------------------------------- ---
-- -- Table ` database`.`logpass` _ _ _
-- ----------------------------------------------- -----
CREATE TABLE IF NOT EXISTS ` database` . `logpass` (
` Login ` VARCHAR(50) NOT NULL ,
` Password` VARCHAR(50) NULL DEFAULT NULL ,
` Status ` INT(11) NOT NULL DEFAULT '0' ,
PRIMARY KEY (` Login` ),
UNIQUE INDEX ` Login_UNIQUE` (` Login`ASC ) )
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
-- ----------------------------------------------- ----
- -- Table ` database` . ` prodgot`
-- ----------------------------------------------- -----
CREATE TABLE IF NOT EXISTS ` database` . ` prodgot` (
` Name` VARCHAR(150) NOT NULL ,
` EdIzm ` VARCHAR(15) NOT NULL ,
` Opis` MEDIUMTEXT NULL DEFAULT NULL,
` Image` MEDIUMBLOB NULL DEFAULT NULL,
` Cena` DECIMAL(10,2) NOT NULL,
PRIMARY KEY (` Name` ),
UNIQUE INDEX ` Name_UNIQUE` (` Name` ASC) )
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
-- ----------------------------------------------- --
--- - Table ` database`.`prod` _ _ _
-- ----------------------------------------------- -----
CREATE TABLE IF NOT EXISTS ` database` . ` prod` (
` NomChecka` INT NOT NULL AUTO_INCREMENT,
` DateProd ` VARCHAR(10) NOT NULL ,
` Login ` VARCHAR(50) NOT NULL ,
PRIMARY KEY (` NomChecka` ))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;



-- ----------------------------------------------- ----
- -- Table ` database` . ` perchecka`
-- ----------------------------------------------- -----
CREATE TABLE IF NOT EXISTS ` database` . `perchecka` (
` NomChecka` INT NOT NULL,
```

```
` Name` VARCHAR(150) NOT
NULL , ` Kolvo` INT NOT NULL,
` Cena` DECIMAL(10,2) NOT NULL,
`ID` INT NOT NULL AUTO_INCREMENT ,
PRIMARY KEY (`ID`) ,
INDEX ` ProdGotName_idx` (` Name` ASC),
INDEX ` ProdNomCheka_idx` (` NomChecka`ASC ),
CONSTRAINT ` ProdGotName`
FOREIGN KEY ( ' Name ')
REFERENCES ` database` . ` prodgot` (` Name` )
ON DELETE CASCADE
ON UPDATE CASCADE,
CONSTRAINT ` ProdNomCheска`
FOREIGN KEY (` NomChecka` )
REFERENCES ` database` . ` prod` (` NomChecka` )
ON DELETE CASCADE
ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

CREATE TABLE IF NOT EXISTS ` database` . ` post` (
` Org` VARCHAR(150) NOT NULL,
` Name` VARCHAR(200) NOT NULL ,
`Tel1` VARCHAR(20) NULL DEFAULT NULL ,
`Tel2` VARCHAR(20) NULL DEFAULT NULL ,
` Adres ` VARCHAR(255) NULL DEFAULT NULL ,
` Email ` VARCHAR(50) NULL DEFAULT NULL ,
`INN` VARCHAR(12) NULL DEFAULT NULL ,
`KPP` VARCHAR(10) NULL DEFAULT NULL ,
`OGRN` VARCHAR(15) NULL DEFAULT NULL ,
` Prim` MEDIUMTEXT NULL DEFAULT NULL,
PRIMARY KEY (` Org` ))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

CREATE TABLE IF NOT EXISTS ` database` . `prodpol` (
` Name` VARCHAR(150) NOT NULL ,
` EdIzm ` VARCHAR(15) NOT NULL ,
` Opis` MEDIUMTEXT NULL DEFAULT NULL,
PRIMARY KEY (` Name` ))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
-------------------------------------------------- ---
---- Table ` database ` . _
-------------------------------------------------- -----
CREATE TABLE IF NOT EXISTS ` database` .
` Name` VARCHAR(150) NOT NULL ,
```

```
` Org` VARCHAR(150) NOT NULL,
` Kolvo` INT NOT NULL,
` Cena` DECIMAL(10,2) NOT NULL,
` NomNakl` VARCHAR(50) NULL DEFAULT NULL,
` DatePost ` VARCHAR(10) NOT NULL ,
` Prim` VARCHAR(255) NULL DEFAULT NULL,
`ID` INT NOT NULL AUTO_INCREMENT ,
PRIMARY KEY (`ID`) ,
INDEX ` ProdPolName_idx` (` Name` ASC),
INDEX ` PostOrg_idx` (` Org`ASC ),
CONSTRAINT ` ProdPolName`
FOREIGN KEY ( ' Name ')
REFERENCES ` database` . ` prodpol` (` Name` )
ON DELETE CASCADE
ON UPDATE CASCADE,
CONSTRAINT ` PostOrg`
FOREIGN KEY (` Org` )
REFERENCES ` database` . ` post` (` Org` )
ON DELETE CASCADE
ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;

USE ` database` ;

SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Додаток Б

Лістинг програмного коду клієнтського модуля

```
package t_kava ;

// Підключення до бази даних на зовнішньому сервері
_ class ConnectToDB {
       public static final String
DB_URL =
"jdbc:ip:3336"; private static final String user =
""; private static final String password = "";

    public static void main ( String [] args
) { try {
            Class.forName ( DB_Driver );
            Connection                connection                =
DriverManager.getConnection (DB_URL);
// System.out.println (" З'єднання з СУБД виконано .");
  boolean is_ok = true ;

} catch ( ClassNotFoundException e) {
            e.printStackTrace ();  //  обробка  помилки
Class.forName
            System.out.println ("JDBC драйвер для СУБД не
знайдено !");
} catch ( SQLException e) {
            e.printStackTrace ();  //  обробка  помилок
DriverManager.getConnection
            System.out.println (" Помилка SQL!");
}
}
}

// Клас ініціалізації користувальницького інтерфейсу
public class MainFrame extends JFrame {
      public MainFrame () {         setBounds (800,
600, 800, 600); // Розміри вікна
}
    public static void main ( String [] args ) {

        SwingUtilities.invokeLater ( new Runnable () {
public void run () {
                MainFrame mf = new MainFrame ();
mf.setDefaultCloseOperation (EXIT_ON_CLOSE); mf.setVisible
( true );
```

```
}
});
}
}
// Клас прийому замовлення та приміщення інгредієнтів у масив
public class MainOrder {
    public static void main ( String [] args ) {

        Map < String ,  Integer >      sostavzakaza   =       new
LinkedHashMap < String , Integer >(); getStorageData (
sostavzakaza );
        Map < String ,  Integer >      OrderMap  =            new
LinkedHashMap < String , Integer >(); if (
sostavzakaza.isEmpty ()) {
            System.out.println    (    Texts.sostavpust    );
return ;
}

printkomponents (
Texts.AVAILABLE_komponentS , sostavzakaza );
showHowTo ();
        Scanner sc = new Scanner (System.in);
while ( sc.hasNext ()) {
            String value = sc.nextLine
(); if ("q". equalsIgnoreCase ( value ))
{                 System.exit (0);
}
            if ("DONE". equalsIgnoreCase (
value )) { break ; }
            processInput ( value , sostavzakaza , OrderMap
);
               if (!
OrderMap.isEmpty ()) {
                 System.out.println ( Texts.REMOVE );
} }         printkomponents ( Texts.YOUR_Order_END ,
OrderMap );
}
          private static void showHowTo () {
System.out.println (
Texts.TEXT_CHOOSE ); System.out.println ( Texts.TEXT_EXIT
);
        System.out.println ( Texts.TEXT_FINISH );
}

       private static void processInput ( String
value ,
```

```
            Map < String , Integer > sostavzakaza , Map <
String ,
Integer > OrderMap ) {
        String [] splittedValue = value.split (""); if
( splittedValue.length == 2) {             int NumIng
= stringToInt ( splittedValue [0]); int Count =
stringToInt ( splittedValue [1]);
                    if ( NumIng <= 0 | | NumIng >
sostavzakaza.size ())
{

System.out.println (
Texts.komponent_NUMBER_LIMIT ); showHowTo ();
return ;
}               processOrderMap ( NumIng ,
komponentCount ,
sostavzakaza , OrderMap );
} else {
            System.out.println ( Texts.SELECTION_ERROR );
showHowTo ();
}
}
    private static void processOrderMap ( int NumIng , int
komponentCount , Map < String , Integer > sostavzakaza ,
Map < String ,
Integer > OrderMap ) {
        StringBuilder key = new StringBuilder ();
if ( komponentaCount > 0 && iskomponentPresent ( NumIng
,
komponentCount , sostavzakaza , key )) {
if ( OrderMap.containsKey ( key.toString ())) { int
amount = OrderMap.get ( key.toString ()); amount + =
komponent Count ;
                OrderMap.put ( key.toString (), amount );
} else { OrderMap.put ( key.toString (), komponent Count );
}
              if     (     sostavzakaza.containsKey     (
key.toString  ()))  {  int  amount  =  sostavzakaza.get  (
key.toString ()); amount -= komponentCount ;
                sostavzakaza.put ( key.toString (), amount
);
}
            printkomponents ( Texts.YOUR_Order , OrderMap
); showHowTo ();
            printkomponents ( Texts.AVAILABLE_komponents ,
sostavzakaza );
```

```
} else if ( komponentaCount < 0 &&
iskomponentPresentInOrder ( NumIng , komponentCount ,
OrderMap
, key )) {             int value = OrderMap.get (
key.toString ());
value += komponentCount ;
            OrderMap.put ( key.toString (), value );

            value = sostavzakaza.get ( key.toString ());
value -= komponentCount ;
            sostavzakaza.put ( key.toString (), value );
            printkomponents ( Texts.YOUR_Order , OrderMap
            );
            showHowTo ();             printkomponents
( Texts.AVAILABLE_komponents ,
sostavzakaza
); } else {

System.out.println ( Texts.komponent_UNAVAILABLE_AMOUNT );
showHowTo ();
}
} private static boolean icomponentPresentInOrder ( int
NumIng
, int komponentCount , Map < String , Integer > OrderMap ,
StringBuilder key ) { int i = 1; for ( Entry < String ,
Integer
> entry : OrderMap.entrySet ())
{
            if (i == NumIng ) {
komponentCount = -1 * komponent count ;
                if              ( komponentaCount              <=
OrderMap.get ( entry.getKey ())) {
key.append ( entry.getKey ());
return
true ; } }
i++; }
        return false ;
} private static boolean iskomponentPresent ( int NumIng ,
int komponentCount , Map < String , Integer > sostavzakaza
, StringBuilder key ) { int i = 1; for  ( Entry < String ,
 Integer >     entry  : sostavzakaza.entrySet ()) { if (i
== NumIng ) {                 if ( komponentaCount > 0 &&
komponentCount <=
entry.getValue ()) {
                    key.append ( entry.getKey ());
```

```
return
true ; } }
i++; }
        return false ;
}
    private static int stringToInt ( String str )
{ try {            return
Integer.parseInt ( str );
} catch ( Exception e) { return
            Texts.ERROR ;
}
}
    private static void printkomponents ( String prefix ,
            Map < String , Integer > komponent Map ) {
        System.out.println ( prefix ); int i = 1; for  (
Entry
< String ,  Integer >          entry  : komponent
Map.entrySet ()) {
            System.out.println (i+"." + entry.getKey () +
":"
+ entry.getValue ()); i++;
}
}


      private static class Texts { private static
String
sostavpust = "На локальному складі немає достатнього кількості
";        private static String AVAILABLE_komponentS = "
Доступні
складові виходячи з залишків сировини :"; private static
String TEXT_CHOOSE = " Виберіть інгредієнт та його
кількість ";         private static String TEXT_EXIT = "
Підтвердіть
завершення зміни ";
        private static String TEXT_FINISH = "Завершіть
вибір
компонентів ";
        private static String SELECTION_ERROR = " Помилка
виконання команди ";
        private static String komponent_NUMBER_LIMIT = " Ви
вказали неправильно внутрішній номер продукту "; private
static final String komponent_UNAVAILABLE_AMOUNT  = "
Вибране кількість сировини не доступно.";        private
static String YOUR_Order = "\n\n\ nСклад
замовлення :"; private static String YOUR_Order_END =
"\n\n\Замовлення складається з :";
private static final String REMOVE = "У вас є
```

```
можливість прибрати додані інгредієнти .";
private
static final int ERROR = -1;
}
}
```