МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) Автоматизації і комп'ютерних систем
Кафедра інформаційних систем

«До захисту в ЕК»	**«До захисту допущено»**
Директор інституту (декан факультету)	Завідувач кафедри
_____ Андрій Форсюк	_____ Сергій Чумаченко
(підпис) (ім'я та прізвище)	(підпис) (ім'я та прізвище)
«___» _____ 2022 р.	«___» _____ 2022 р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

зі спеціальності 122 «Комп'ютерні науки»
(код та назва спеціальності)
освітньо-професійної програми _____
на тему: Створення інформаційної підтримки діяльності інтернет-магазину з продажу продукції ПрАТ "Оболонь"

Виконав: здобувач 4 курсу, групи КН-4-3СК
Васильченко Ілля Борисович _____
(прізвище, ім'я, по батькові повністю) (підпис)

Керівник Ліманська Наталія Володимирівна _____
(прізвище, ім'я, по батькові повністю) (підпис)

Консультанти_____ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

Рецензент Володимир Овчарук _____
(ім'я та прізвище) (підпис)

Я як здобувач(ка) Національного університету харчових технологій розумію і підтримую політику університету з академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволеної допомоги під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Здобувач_____
(підпис)

Київ - 2022р.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) Автоматизації і комп'ютерних систем
Кафедра Інформаційних систем
Освітній ступінь бакалавр
Спеціальність 122 «Комп'ютерні науки»
(код і назва)
Освітньо-професійна програма «Комп'ютерні науки»
(назва)

ЗАТВЕРДЖУЮ
Завідувач
кафедри Інформаційних систем

_____Чумаченко С.М.
"____" _____2022року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

Васильченко Іллі Борисовича
(прізвище, ім'я, по батькові)

1. Тема роботи: Створення інформаційної підтримки діяльності інтернет-магазину з продажу продукції ПрАТ "Оболонь"
керівник роботи Ліманська Наталія Володимирівна, ст.викл. _____,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від "31 " 03 2022 року №163 -кс
2. Строк подання здобувачем роботи 06 червня 2022 р
3. Вихідні дані до роботи *дані_про ПрАТ "Оболонь" , дані про товару які продаються через інтернет-крамницю ПрАТ «Оболонь», документи та супровідна документація для функціонування інтернет-крамниці*
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Системний аналіз діяльності підприємства ПрАТ «Оболонь», опис комплексу задач автоматизації
5. Перелік графічного матеріалу
1. Організаційна структура підприємства
2. Контекстна діаграма, функціональної модель, діаграми декомпозиції
3. Моделі даних
4. Приклади таблиць в середовищі PostgreSQL
5. Макети UI-дизайну
6. Приклади інтерфейсу системи

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
I	Ліманська Наталія Володимирівна, ст.викл.		
II	Ліманська Наталія Володимирівна, ст.викл.		

7. Дата видачі завдання___31 травня 2022 року_____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів виконання кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Системний аналіз інтернет-крамниці ПрАТ «Оболонь»	**20.02.2022 – 10.03.2022**	Виконано
2	Розробка моделі даних та проектування бази даних	**27.03.2022 – 06.04.2022**	Виконано
3	Визначення та реалізація функції ІС	**07.04.2022 – 20.05.2022**	Виконано
4	Оформлення пояснювальної записки	**25.05.2022 – 30.05.2022**	Виконано
5	Розробка презентації	**30.05.2022 – 31.05.2022**	Виконано
6			
7			
8			
9			

Здобувач _____ __Васильченко І.Б.____.
(підпис) (прізвище та ініціали)

Керівник роботи _____ ____Ліманська Н.В._____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Головною метою даної бакалаврської роботи є розробка система для управління інтернет-крамницею підприємства ПрАТ «Оболонь».

Сама робота розділена на чотири етапи, які описані в двох розділах цієї бакалаврської роботи.

Об'єктом дослідження в кваліфікаційній роботі є інтернет-магазин підприємства ПрАТ «Оболонь».

Бакалаврська робота містить 86 сторінок, 4 таблиць, 84 рисунка, 7 додатків і 12 літературних джерел.

КЛЮЧОВІ СЛОВА: ІНФОРМАЦІЙНА СИСТЕМА, ГНУЧКІСТЬ, АДАПТИВНІСТЬ, PYTHON.

ANNOTATION

The main purpose of this bachelor's work is to develop a system for managing the online store of PJSC "Obolon".

The work itself is divided into four stages, which are described in two sections of this bachelor's thesis.

The object of research in the qualification work is the online store of PJSC "Obolon".

The bachelor's work contains 86 pages, 4 tables, 84 figures, 7 appendices and 12 references.

KEY WORDS: INFORMATION SYSTEM, FLEXIBILITY, ADAPTABILITY, PYTHON.

Зміст

ВСТУП

Інтернет-крамниця – це електронний ресурс, який реалізує прямий продаж товару клієнту. На відміну від звичайних магазинів, розміщення інформації щодо товарів відбувається прямо на сайті магазину, як і сама угода на покупку товару між клієнтом та продавцем.

Є безліч інформаційних систем, які забезпечують управління інтернет-крамницями, але вони мають лише певні можливості та функції, а для адаптації під бізнес-модель підприємства спроможні лише системи розроблені спеціально під них.

Гибка система – це можливість відносної гибкості функцій та модулів в випадку необхідності трансформації системи.

В кінцевому вигляді повинна бути ефективна системи зі зрозумілим та зручним інтерфейсом.

РОЗДІЛ 1.СИСТЕМНИЙ АНАЛІЗ ДІЯЛЬНОСТІ ПІДПРИЄМСТВА ПРАТ «ОБОЛОНЬ»

1.1 Загальна характеристика підприємства ПрАТ «Оболонь»

В рамках проходження практики було досліджено ПрАТ «Оболонь» — корпорацію з виробництва пива, безалкогольних та слабоалкогольних напоїв, мінеральної води.

Головний завод корпорації Оболонь , має проектну потужність 14 млн дал пива на місяць і є найбільшою пивоварнею в Європі. За останні півтора роки на заводі встановлена унікальна варильна система, яка здійснює 12 варок на день по 740 гектолітрів холодного сусла. Особливістю цього нового варочного порядку німецької фірми Ziemann є те, що він дозволяє економити витрати на теплову енергію під час виробництва сусла. А енергоємність є одним із пріоритетних завдань, які стоять перед інженерами компанії.

Також серед останніх оновлень заводу є лінія розливу у скляні пляшки потужністю 50 тис. пляшок на годину. Лінію змонтували у рекордні терміни. Один із предметів гордості – це унікальна установка, яка проводить рекуперацію пивних дріжджів. Ця технологія дозволяє налагодити практично безвідходне виробництво продукції.

Гарантією високої якості продукції є найсучасніші потужні фільтраційні установки, виготовлені також німецькою фірмою. Потужність одного з фільтрів становить 950 гектолітрів на годину. Фільтраційна система “Оболоні” не має аналогів не лише в Україні, а й у Європі. Введення сучасних технологій дало змогу вивести на ринок унікальний продукт "Оболонь Живе". Це дало новий поштовх до покращення процесу пивоваріння. І першою в цьому стала броварня "Оболонь".

Основним експортером українського пива залишається компанія "Оболонь". Незважаючи на всі складнощі, український пивоварний гігант, як і раніше, є лідером галузі.

Пивоварна корпорація «Оболонь», незважаючи на скорочення обсягів виробництва (у сезон виробничі лінії компанії працювали лише наполовину

своїх можливостей), лишається найбільшим постачальником українського пива на зовнішній ринок. За підсумками першого півріччя 2021 року її частка в експорті цього напою склала 49% у натуральному вираженні. Основними ринками збуту для компанії є Молдова та Білорусь. Хоча поставки здійснюються у більш ніж 50 країн, у тому числі Азії та Балтії, а також до Японії та Китаю. Компанія також розвиває сегмент мінеральної води.

1.2 Організаційна структура сільськогосподарського підприємства ПрАТ «Оболонь»

1.2.1 Загальна схема організаційної структури

Підприємство засновано в 1980 році, за 42 роки воно вдосконалює не тільки продукт, який виготовляє, а й свою організаційну роботу, тому на поточний момент воно має чітко виражену організаційну структуру.

Генеральний директор на підприємстві – це головний керуючий, якого обирає рада директорів корпорації. Він має підлеглих, які в свою чергу є також керівниками більше меншого складу персоналу.

Фінансовий директор має підлеглих, які є також керівниками невеликих відділів. По його підлеглих належать головний бухгалтер, фінансовий інспектор, керівник відділу стратегічного розвитку, керівник юридичного відділу, керівник маркетингового відділу та керівник відділу збуту.

Загальний вигляд організаційної структури відображено на рис.1.1.

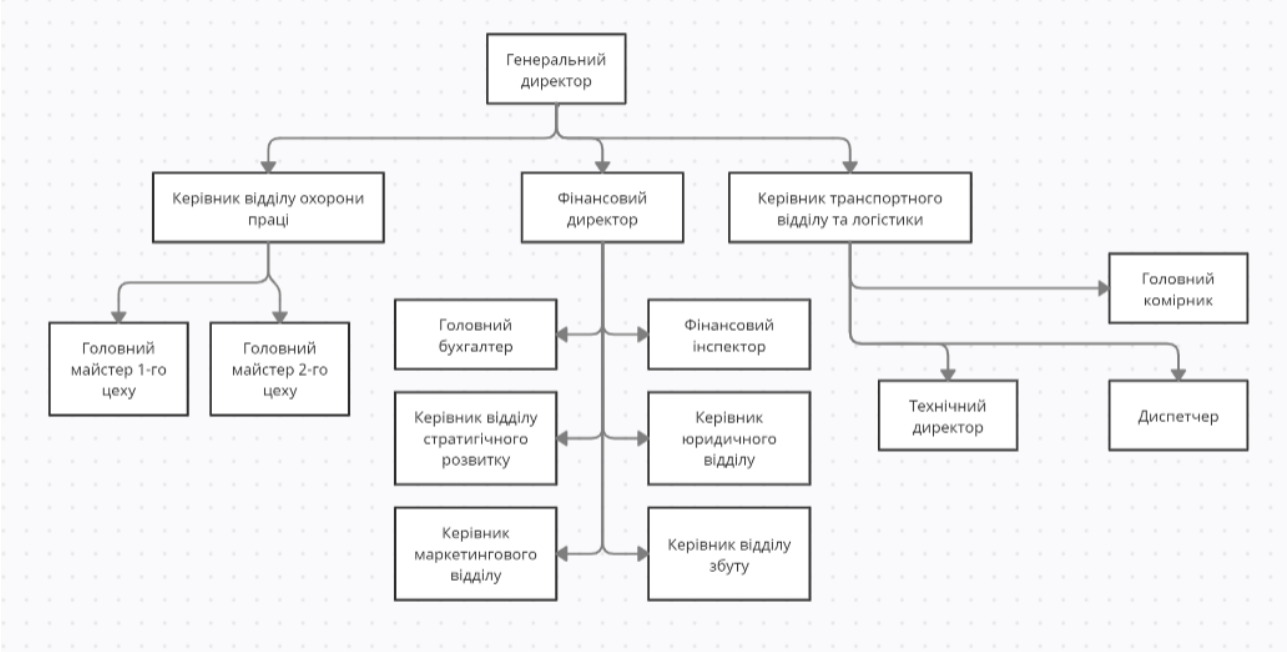


Рис.1.1. Організаційна структура підприємства

Кожний відділ та підрозділ підприємства має свої обов'язки, задачі та несе відповідальність в межах своєї компетентності.

1.2.2 Схема організаційної структури відділу збуту ПрАТ «Оболонь»

Схема організаційної структури відділу збуту ПрАТ «Оболонь» зображена на Рис.1.2.2.1.

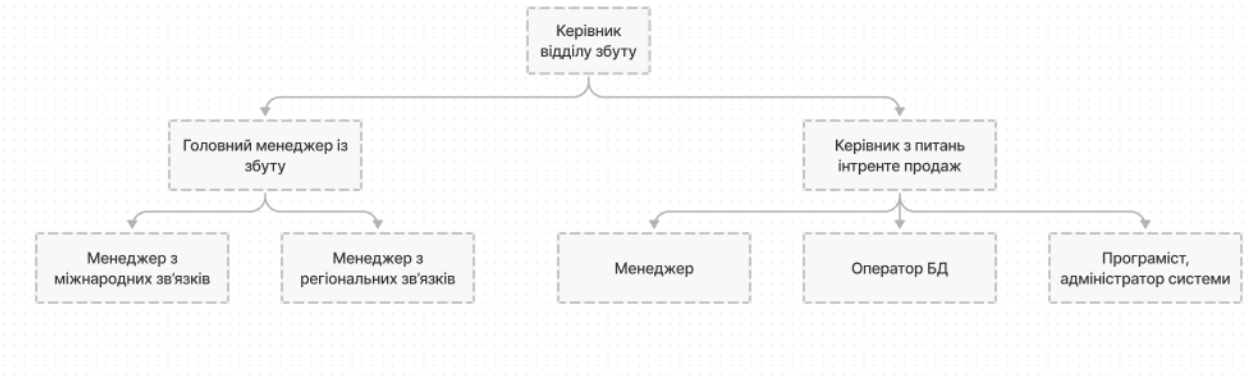


Рис.1.2.2.1. Структура відділу збуту

Функції та задачі відділу збуту відображені в таблиці 1.2.2.1.

Таблиця 1.2.2.1. Функції та задачі відділу збуту

№	Задачі	Функції
1	Організація збуту продукції.	Участь у підготовці прогнозів, проектів перспективних і поточних планів виробництва і реалізації продукції, у проведенні маркетингових досліджень по вивченню попиту на продукцію підприємства, перспектив розвитку ринків збуту. Узгодження умов постачань. Складання планів постачань і їхнє узгодження з планами виробництва з метою забезпечення здачі готової продукції виробничими підрозділами в терміни. Виконання плану реалізації продукції. Постійне відстеження ринку своєї продукції, покупка зразків продукції, вироблених підприємствами-конкурентами, порівняння якості і попиту на цю продукцію з власною продукцією і при необхідності доведення якості до необхідних рівнів. Підготовка пропозицій по зменшенню обсягів виробництва продукції з обмеженим попитом і зняттю з виробництва продукції, що користується низьким попитом. Підготовка і проведення заходів щодо формування збутової продукції.

Таблиця 1.2.2.1. Функції та задачі відділу збуту (Продовження)

2	Планування постачань продукції відповідно до укладених договорів.	Формування пропозицій по корегуванню цін на пропоновану продукцію залежно від обсягів продажів. Організація оптової торгівлі. Уживання заходів по забезпеченню своєчасної оплати поставленої продукції.
3	Виконання планів постачань у терміни і в обсязі відповідно до замовлень і укладених угод.	Вивчення вимог покупців до якості, упакування й асортименту продукції: - проведення опитувань споживачів; - організація спеціальної телефонної служби по консультаціях про користування продукцією і видачу рекомендацій; Уживання заходів по своєчасному одержанню форм, специфікацій і інших документів на постачання.

1.2.3 Взаємодія відділу збуту з другими відділами

Працівники інтернет-крамниці ПрАТ «Оболонь», яка входить та підпорядковується відділу збуту пов’язана та з іншими відділами підприємства, що можна побачити у табл.1.2.3.1.

Таблиця 1.2.3.1 Взаємодія підрозділів

№	Підрозділ	Одержання	Надання
1	З виробничими підрозділами	- звітів про виконання виробничих завдань з обсягу виробництва продукції - готової продукції по затвердженій номенклатурі; - документації про здачу продукції у відділ збуту;	- планів постачань продукції - запитів про причини порушення планів передачі готової продукції на склади готової продукції відділу збуту; - пропозицій щодо змін термінів і графіків передачі продукції на склади готової продукції; - документації про прийом продукції від виробничих підрозділів;
2	З відділом розвитку	- документів, що підтверджують якість продукції - документів, що супроводжують продукцію і належать до передачі покупцям	- рекламацій і претензій до якості продукції, отриманих підприємством від споживачів

Таблиця 1.2.3.1 Взаємодія підрозділів (Продовження)

3	З відділом головного технолога	- повідомлень про технологічні зміни, внесені у продукцію, що випускається - проектів введення нових технологій - технічної документації по якості	- пропозицій по збільшенню обсягів виробництва продукції, що користується найбільшим попитом, і по запуску у виробництво нових видів продукції; - інформації про рекламації до якості продукції; обсяги реалізації продукції по окремих позиціях товарної номенклатури;.
4	З фінансовим відділом	- інформації про неоплачені рахунки; - повідомлень про застосування фінансових санкцій до покупців (замовників), що порушили зобов'язання по перерахуванню коштів за придбані товари;	- проектів договорів і угод на постачання, продаж готової продукції; - прогнозів і планів реалізації продукції; - даних про стан запасів готової продукції і їхню відповідність затвердженим нормативам; - даних про залишки продукції на складах;

Таблиця 1.2.3.1 Взаємодія підрозділів (Продовження)

5	З головною бухгалтерією	- даних про обіг продукції; - норм запасів продукції на складах; - норм природного збитку; - підсумків інвентаризації продукції;	- товарно-супровідної документації; - даних про стан запасів продукції на складах готової продукції; - інформації про витрати, зроблених на відвантаження продукції;
6	З відділом маркетингу	- узагальненої інформації про попит на продукцію, що випускається підприємством - інформації про стан товарного ринку; - відомостей про великих покупців продукції - відомостей про плановані виставки, ярмарки;	- даних про укладені угоди постачання; - дані реалізації продукції на місяць, квартал, рік; - документів, необхідних для оформлення участі у виставках, ярмарках

1.3 Діаграма діяльності роботи інтернет-крамниці

Під час проходження практики на підприємстві для дослідження функціонування інтернет-крамниці було знайдено працівників та проведено

опитування про етапи замовлення та створено діаграму діяльності, яка включає наступні етапи: формування замовлення, комплектування замовлення, закриття замовлення. Діаграма діяльності відображена на Рис. 1.3.1.

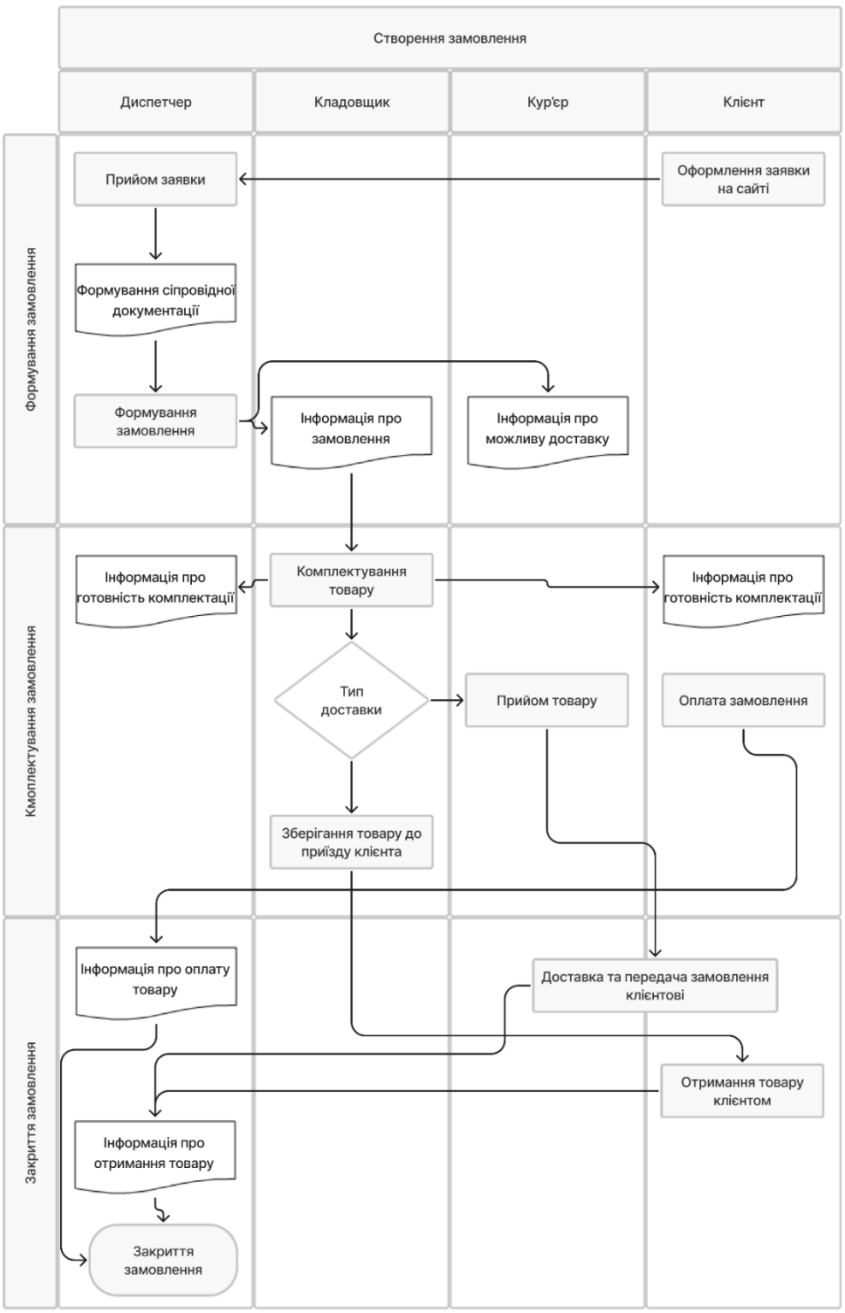


Рис.1.3.1. Діаграма діяльності

1.4 Стан автоматизації відділу

Програмне забезпечення, яким користуються в відділі збуту – це інтернет магазин написаний на фреймворці Django.

Django (Джанго) - високорівневий відкритий Python-фреймворк для розробки вебсистем. Щодо його можливостей, які активно використовуються – це вбудована панель адміністратора. Безпосередньо через неї і керую контентом інтернет крамниці, змінюючи записи в базі даних. Вибірка інформації з БД для потреб працівників відбувається через адміністраторів системи, оскільки тільки в них є навички для конструювання SQL-запитів.

1.5 Розроблення функціональної моделі та аналіз існуючих бізнес-процесів

1.5.1 Функціональна модель діяльності інтернет-крамниці ПрАТ «Оболонь»

Перед створення інформаційної системи слід проаналізувати предметну область та визначити всі етапи замовлення. Для цього завдання добре відходить інструментарій Erwin Process Modeler. Спочатку потрібно створити контекстну діаграму, далі провести два рівні декомпозиції. Загальний вигляд функціональної моделі в нотації IDEF0 можна побачити в додатках до цієї роботи.

Також слід визначити документи, які використовуються у бізнес-процесі інтернет-крамниці. До них слід віднести:

- Законодавство України
- Організаційні документ

Також слід визначити механізми, які використовуються в роботі інтернет-крамниці.

- Персонал
- Клієнт
- Програмне забезпечення

Контекстна діаграма декомпозується на нижчі рівні, моделі також можливо побачити в додатках до цієї роботи.

Також слід визначити на які питання відповідає модель. До цих питань можна віднести:

1. Як організована робота інтернет крамниці?
2. Як відбувається авторизація чи реєстрація клієнта?
3. Хто є кінцевим споживачем продукції ПрАТ "Оболонь"?
4. Які документи формуються на етапі організації доставки?
5. Як і коли розраховується вартість доставки?
6. Як і коли розраховується дата доставки?
7. Як і в який час виконується оплата за товар?
8. Які є етапи перевірки вантажу?
9. Чим керуються коли вибирають транспорт для перевезення замовлення?
10. Які документи формуються на етапі авторизації клієнта?
11. Які документи формуються на етапі оформлення замовлення?
12. Які документи формуються на етапі комплектації товару?

1.5.2 Виявлені проблеми

Працівники компанії виконують значить обсяг роботи, від зміни контенту в крамниці до підрахунку статистичних даних для прогнозування майбутніх продаж. Від зручності використання панелі управління інтернет-крамницею залежить результативність роботи персоналу. Заплутана та непродумана панель не тільки ускладнює роботу працівників, а й підвищує ймовірність помилок, а отже потребує додаткових витрат на їх усунення.

Головна проблема – це відсутність автоматизаваної статистики, яка потрібна для прийняття рішень, які безпосередньо впливають на дохід підприємства. Також оскільки системи обліку наявна тільки у адміністраторів системи, то це сповільнує роботу всіх відділів, які залежать від інформації наявної в ній. Всі виявлені проблеми можливо згрупувати на наступні пункти:

- Для відображення статистики потрібно кожного разу конструювати запити.
- Для зміни даних в БД використовується заплутана та непродумана стандартна панель
- Відсутність виведення даних в графічному вигляді
- Відсутність кабінету директора
- Відсутність експорту до таблиць Excel
- Обмежена фільтрація до наборів даних з БД

1.5.3 Задачі автоматизації

Розроблена система має бути більше, ніж просто облік продаж та контроль за контентом інтернет-крамниці. Вона повинна автоматизувати більшість процесів підтримки діяльності крамниці.

Ціль системи – це спрощення роботи працівників компанії, автоматизуючи розрахунки, які потрібні для прийняття більшості рішень. Також слід покращити процес маніпулювання даними, тобто введення та редагування даних.

Але сучасне підприємство реагує на безперервні зміни у зовнішньому середовищі, трансформуючи свою поведінку та бізнес-процеси. Трансформації бізнесу викликають зміни в інформаційних системах, що підтримують бізнес-процеси, та зміни у відповідній IT-інфраструктурі. Ці зміни заздалегідь не передбачити. З одного боку, інформаційна система повинна забезпечувати ефективність поточної моделі бізнесу, а з іншого мати необхідний рівень гнучкості для реалізації непередбачуваних змін. Головною задачею є розробити гнучку систему, яка готова адаптовуватись під зміни на підприємстві.

На поточний момент постає питання розробити систему для керівників, менеджерів, операторів бази даних та адміністратора системи для управління забезпеченням функціями інших користувачів системи.

Обов’язками оператора бази даних та менеджера є заповнення БД та отримання різноманітної статистики для подальшого оформлення звітності в різних середовищах розробки.

Обов’язками керівника є слідкування за персоналом та прийняття важливих рішень з продажу продукції в магазині.

Обов’язками адміністратора системи є слідкування за користувачами системи, тісно контактуючи з ними та виконуючи їхні прохання щодо корегування функцій та налаштувань системи.

Кожна з ролей повинна забезпечувати доступ тільки до тієї низки інформації, що потрібно для якісного виконання задач.

Інформаційна система повинна виконувати такі функції:

- Виведення даних у зручному вигляді.
- Пошук інформації у таблицях за певними критеріями.
- Прогнозування продаж за допомогою методів прийняття рішень.
- Імпорт даних з таблиць Excel.
- Експорт даних в таблиці Excel.
- Формування різноманітної статистики та звітності.

1.6 Огляд існуючих рішень для розв’язання виявлених проблем

Для вирішення виявлених проблем можливо вже існує система, яка забезпечить всі функції, тому слід проаналізувати їх та прийняти рішення щодо необхідності розробки системи.

Таблиця 1.6.1. Огляд існуючих рішень

Назва системи	**DilayQ**	**OdeeStats**	**ArchitectUI**
ОС	Windows, Linux, Mac OS	Windows	Windows, Mac OS, Linux, Android, IOS
Вартість	Від 18000 грн	Від 1200 за одне робоче місце	Від 5000 грн / щомісячно
Статистика даних	Є всі необхідні модулі для формування статистичних даних	Базові функції, для формування статистики, без можливості їх адаптації під певну структуру	Є всі необхідні модулі для формування статистичних даних
Кабінет директора	Є	Немає	Є
Прогнозування даних продажу	Немає	Немає	Немає
Виведення статистичних даних в графічному вигляді	Немає	Немає	Частково, тільки до деякого набору даних і тільки в круговій діаграмі
Останнє оновлення	2018 рік	2012 рік	2022 рік

Отже розглянувши системи, які описані в таблиці 1.6.1. можна зробити висновок, що DilayQ та ArchitectUI є дорогими та не забезпечать всіма необхідними функціями працівників підприємства. А більше дешева система OdeeStats не забезпечить й половину потрібних функцій. До того ж останнє оновлення цієї системи було аж 10 років тому.

1.7 Обґрунтування доцільності проектування й розроблення інформаційної підтримки для інтернет-крамниці підприємства ПрАТ «Оболонь»

На основі даних з аналізу систем, які використовуються на поточний момент підприємством можна зробити висновок, що необхідна система повинна забезпечувати всіма необхідними функціями та вирішити всі виявлені проблеми. При цьому всі існуючі системи або досить дорогі, або взагалі не підходять нам. І тому було прийнято рішення розробити власну інформаційну систему, яка буде забезпечувати всі необхідні функції та буде адаптована під ситуацію в поточній бізнес-моделі та під структуру підприємства. А в подальшому буде підлаштовуватись під зміни та буде мати переваги над вже існуючими системами.

Отже, розроблення та впровадження інформаційної системи для керування інтернет-крамницею є доцільним для вирішення всіх наявних проблем.

РОЗДІЛ 2. ОПИС КОМПЛЕКСУ ЗАДАЧ АВТОМАТИЗАЦІЇ

2.1. Інформаційне забезпечення системи

Оскільки головний пріоритет керівництва зростаючої компанії це можливість швидкого масштабування продукту, то потрібно розробити кросплатформлену систему, яка відображає будь-яку інформації згідно прописаних інструкцій.

В сучасному світі більшість програм, що створюються для кінцевого користувача, мають графічний інтерфейс. Ця інформаційна система не виняток, вона розроблена з використанням мова програмування Python та фреймворку для створення інтерфейсу - PyQT5, також обрано СУБД – PostgreSQL.

З самого початку потрібно розробити модель бази даних в нотації IDEF1X, для цього завдання обратно продукт AllFusion ERWin Data Modeler. Ці моделі можна побачити в додатках до цієї роботи.

Наступний етап - це перенесення структури бази до обраного сервера та забезпечення підключення бд до системи. Структуру показано в таблиці 2.1.

Таблиця 2.1. Структура фізичної моделі та правила валідації

№	Таблиці	Колонки та їх типи даних	Правила валідації
1	Доставка	o КодДоставки (PK): integer o ВартістьДоставки: double o СтатусВідправлення: character varying(20) o ДатаДоставки: date o ПлановаДатаДоставки: date o ТипДоставки: character varying(20)	"ВартістьДоставки" >= 0 "ТипДоставки" = 'Самовивіз' OR "ТипДоставки" = 'Курєрська доставка'

Таблиця 2.1. Структура фізичної моделі та правила валідації (продовження)

2	Замовлення	○ КодЗамовлення (PK): integer ○ ДатаЗамовлення: date ○ ЗагальнаВартість: double ○ НомерЗамовлення integer ○ КодДоставки (FK): integer ○ КодКлієнта (FK): integer	"ЗагальнаВартість" >= 0
3	Квитанція	○ КодКвитанції (PK): integer ○ КодЗамовлення (FK): integer ○ Сума: double ○ Дата: date ○ НомерКвитанції integer	"Сума" >= 0
4	Клієнт	○ КодКлієнта (PK): integer ○ Прізвище: character varying(40) ○ Імя: character varying(20) ○ ПоБатькові: character varying(40) ○ НомерТелефону: integer ○ Почта: character varying(40) ○ Пароль: character varying(255) ○ ДатаРеєстрації: date ○ ДатаОстанньогоВходу: date ○ ІндифікаційнийКод: integer	

5	КурєрськаСлужба	○ КодЗапису (PK): integer ○ КодДоставки (FK): integer ○ НазваКурєрськоїСлужби: character varying(18) ○ НомерВідправлення: integer ○ ВартістьДоставки: integer	
6	Прайслист	○ КодЗапису (PK): integer ○ КодТовару (FK): integer ○ Ціна: double ○ ДатаПочаткуДії: date ○ ДатаКінцяДії: date	"Ціна" >= 0
7	РядокЗамовлення	○ КодРядкаЗамовлення (PK): integer ○ КодЗамовлення (FK): integer ○ Кількість: integer ○ Ціна: double ○ КодТовару (FK): integer	
8	Самовивіз	○ КодЗапису (PK): integer ○ КодДоставки (FK): integer ○ НомерВідправлення: integer ○ АдресаПунктуВидачі: character varying(20)	

Таблиця 2.1. Структура фізичної моделі та правила валідації (продовження)

9	Склад	○ КодСкладу (PK): integer ○ КодТовару (FK): integer ○ Кількість: integer ○ Адреса: character varying(80) ○ ДатаВнесенняЗмін: date	"Кількість" >= 0
10	Товар	○ КодТовару (PK): integer ○ НазваТовару: character varying(80) ○ Одиниця_виміру: character varying(20)	

2.2. Підключення бази даних

Дані для підключення до бази даних прописуємо в окремому файлі, щоб при необхідності була можливість змінити їх.

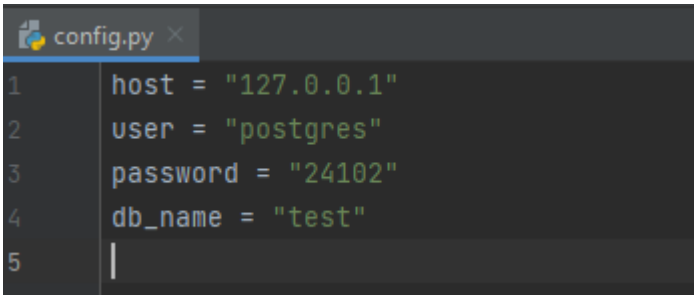


Рис. 2.2.1. Конфігурація параметрів для відкриття з’єднання з базою даних

Запити до сервера виконані в окремому файлі, для початку робити запиту потрібно викликати функцію та передати до неї відповідні параметри. Приклад запиту для відображення всіх даних за назвою таблиці показано на Рис. 2.2.

```
def getcon(name):
    try:
        connection = psycopg2.connect(
            host=host,
            user=user,
            password=password,
            database=db_name
        )
        connection.autocommit = True

        with connection.cursor() as cursor:
            string = "SELECT * FROM " + name
            cursor.execute(
                string
            )
            print(string)
            Arr = []
            for i in cursor:
                Arr.append(i)
            return Arr

    except Exception as _ex:
        print("[INFO] Error with DB:", _ex)
        s = "Помилка в БД:\n" + str(_ex)
        startMessage(s)
```

Рис. 2.2.2. Функція для отримання даних з БД

2.3. Форма авторизації

Першим кроком потрібно створити вікно авторизації, щоб кожен з користувачів міг зайти під свою систему, яка забезпечить його інформацією та функціоналом для якісного виконання своїх обов’язків.

Для збереження інформації про користувачів системи є відповідна таблиця, яка в собі містить інформацію про логін, пароль та роль користувача в системі.

```
"КодЗапису" integer NOT NULL DEFAULT nextval('"Користувач_КодЗапису_seq"'::regclass),
"ПІБ" character varying COLLATE pg_catalog."default",
"Статус" character varying COLLATE pg_catalog."default",
"Логін" character varying COLLATE pg_catalog."default",
"Пароль" character varying COLLATE pg_catalog."default",
"Права" character varying COLLATE pg_catalog."default",
CONSTRAINT "Користувач_pkey" PRIMARY KEY ("КодЗапису")
```

Рис 2.3.1. Відображення SQL-запиту для створення таблиці в БД для забезпечення авторизації

Для отримання інформації про користувача потрібно відправити до БД дані входу, які ввів користувач, та отримати інформацію з БД, щодо їх присутності.

```
with connection.cursor() as cursor:
    query = "select Права from Користувач where \"Логін\" = '"
    query += login
    query += "' and \"Пароль\" = '"
    query += passw
    query += "'"
    print(query)
    cursor.execute(
        query
    )
```

Рис 2.3.2. Відображення SQL-запиту для отримання інформації з БД

```
def loginCommand():
    global openWindow, uiLog, loginWin
    login = uiLog.lineEdit.text()
    user = (datadb.getUserStatus(login, uiLog.lineEdit_2.text()))
    if len(user) == 0:
        messageinformation.startMessage("Немає такого користувача")
        uiLog.lineEdit_2.setText("")
        uiLog.lineEdit.setText("")
    elif user[0] == 'OperatorDB':
        appValue.getValue().closeAllWindows()
        import test2
        setCurrentWindow("DBForm")
        test2.mainStart(login)
    elif user[0] == 'Admin':
        appValue.getValue().closeAllWindows()
        import admForm.startWindow
        admForm.startWindow.mainStart()
    elif user[0] == 'Director':
        appValue.getValue().closeAllWindows()
        import directorForm.startWindow as directorStartForm
        setCurrentWindow("DirectorForm")
        directorStartForm.mainStart(datadb.getother("SELECT ПІБ From Користувач WHERE Логін = '" + login + "'")[0][0])
```

Рис 2.3.3. Перевірка ролі клієнта в системі та відкриття потрібної форми

Також слід забезпечити можливість надсилання запиту на зміну паролю для входу систему. При цьому також розробити перевірку на присутність логіна в системі.

```
def resetPasswordCommand():
    if uiLog.lineEdit.text() == "":
        messageinformation.startMessage("Введіть логін")
        return
    logState = main.getother("SELECT count(\"Логін\") FROM public.\"Користувач\" where \"Логін\" = '" + uiLog.lineEdit.text()
    if int(logState[0][0]) > 0:
        main.getother("update public.\"Користувач\" set \"Статус\" = 'NeedNewPassword' where \"Логін\" = '" + uiLog.lineEdit.t
        messageinformation.startMessage("Запит надіслано")
```

Рис 2.3.4. Перевірка ролі клієнта в системі та відкриття потрібної форми

Загальний вигляд форми представлено на Рис 2.3.5.

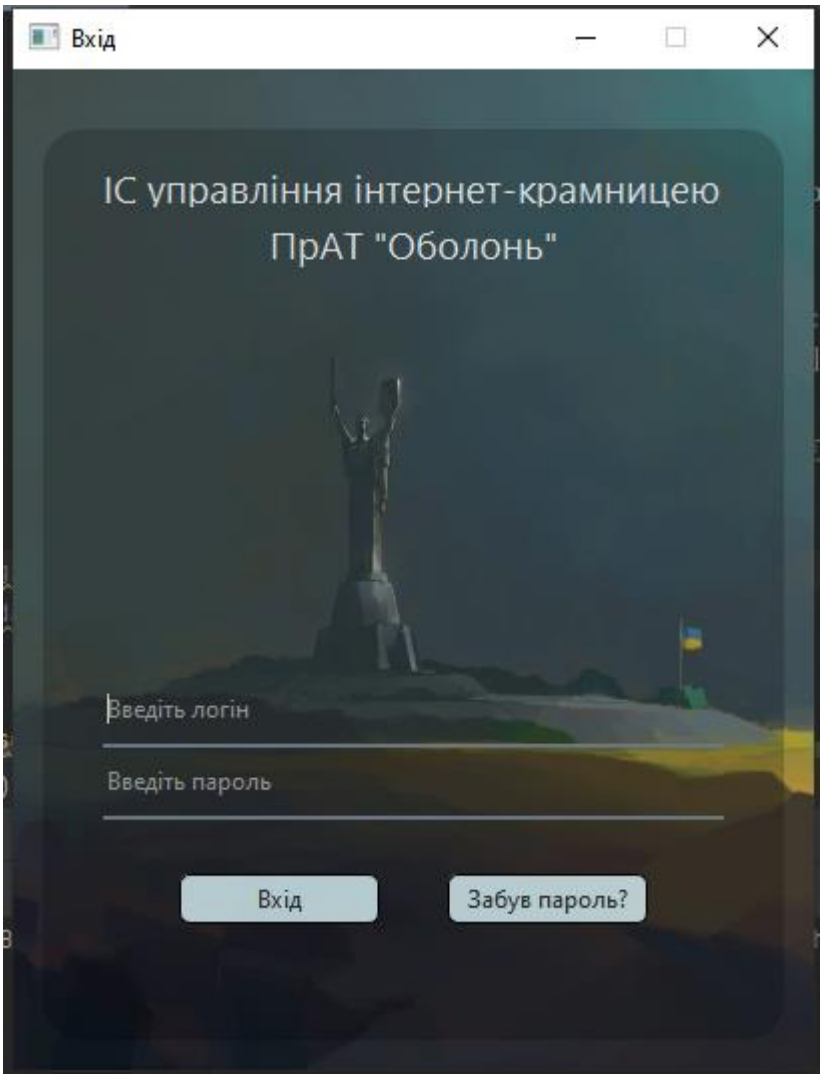


Рис 2.3.5. Форма авторизації

2.4. Сповіщення

Одним із головних завдань кожної інформаційної системи є забезпечення користувача інформацією про те, що відбувається під час його тих або інших дій.

Розуміючи те, що відбувається у додатку, можна не помилитися у діях та прийняти правильне рішення.

Це питання було вирішено за допомогою розробки модуля сповіщень. Всі виявлені помилки під час користування додатком та результат дій користувача, які несуть за собою зміни в БД відображаються в вигляді сповіщень на екрані.

```
def __init__(self, message):
    super(PopupWindowClass, self).__init__()
    self.setWindowFlags(QtCore.Qt.SplashScreen | QtCore.Qt.FramelessWindowHint | QtCore.Qt.WindowStaysOnTopHint)

    self.setMinimumSize(QtCore.QSize(300, 100))
    self.animation = QtCore.QPropertyAnimation(self, b"windowOpacity", self)
    self.animation.finished.connect(self.hide)
    self.timer = QtCore.QTimer()
    self.timer.timeout.connect(self.hideAnimation)
    self.setupUi()
    self.setPopupText(message)
```

Рис. 2.4.1. Код ініціалізації віджету «сповіщення»

```
def move2RightBottomCorner(win, countMessage):
    try:
        screen_geometry = QtWidgets.QApplication.desktop().availableGeometry()
        screen_size = (screen_geometry.width(), screen_geometry.height())
        win_size = (win.frameSize().width(), win.frameSize().height())
        x = screen_size[0] - win_size[0] - 10
        y = screen_size[1] - win_size[1] - 10 - (countMessage * 110)
        win.move(x, y)
    except Exception as e:
        print(e)
```

Рис. 2.4.2. Код управління розташування сповіщеннями на моніторі користувача

```
def show(self):
    self.setWindowOpacity(0.0)
    self.animation.setDuration(1500)
    self.animation.setStartValue(0.0)
    self.animation.setEndValue(1.0)
    QtWidgets.QWidget.show(self)
    self.animation.start()
    self.timer.start(5000)

def hideAnimation(self):
    self.timer.stop()
    self.animation.setDuration(1000)
    self.animation.setStartValue(1.0)
    self.animation.setEndValue(0.0)
    self.animation.start()

def hide(self):
    if self.windowOpacity() == 0:
        QtWidgets.QWidget.hide(self)
        self.popuphidden.emit()
        global numbers
        numbers = numbers + 1
```

Рис. 2.4.3. Код анімацій для відтворення та приховання сповіщення

2.5. ІС для операторів БД та менеджерів інтернет-магазину

2.5.1. Меню

Для можливості дистанційного корегування меню потрібно створити окрему таблицю в базі даних, де потрібно буде прописувати інструкції для відображення даних.

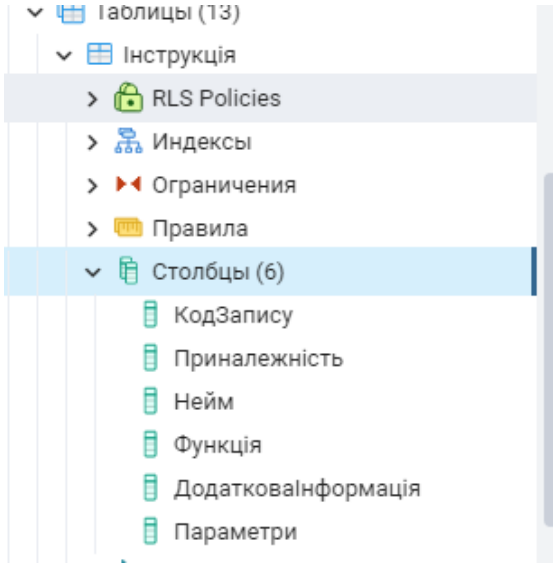


Рис. 2.5.1.1. Деталізація таблиці «Інструкція»

Алгоритм відображення меню:

1. Отримуємо інструкцію з БД.
2. Відображаємо меню в відповідних блоках, згідно інструкції.
3. Присвоюємо кожній кнопці функцію, яку вона повинна виконувати, згідно отриманих інструкцій.

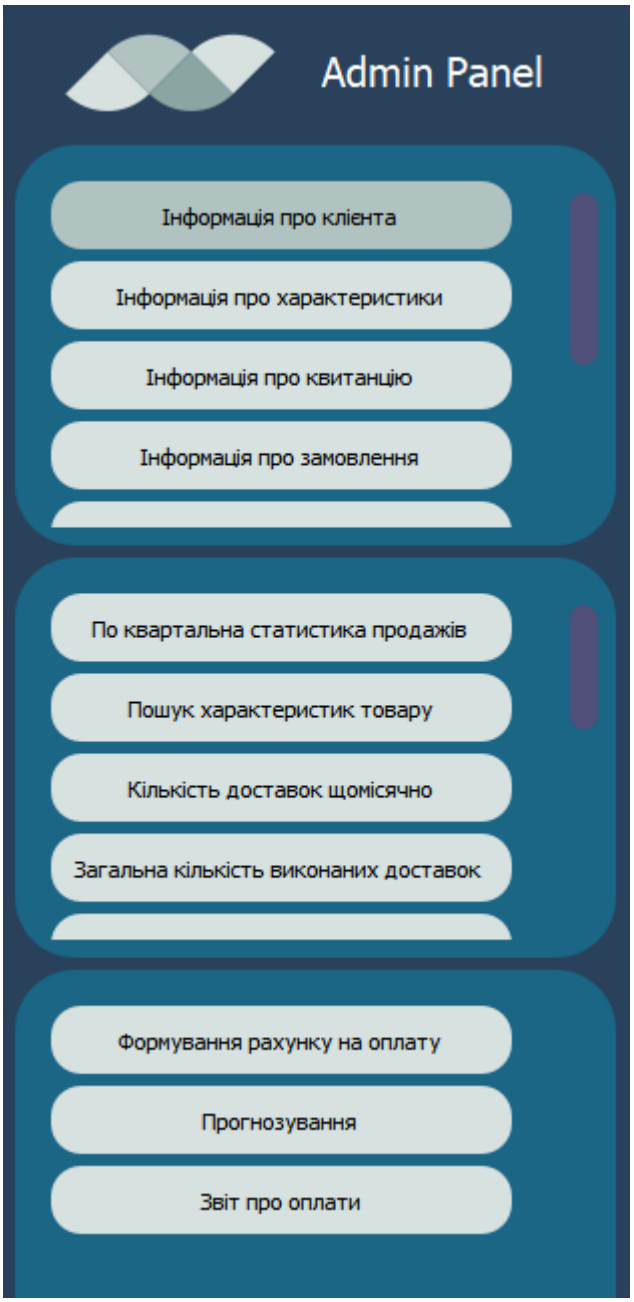


Рис. 2.5.1.2. Меню користувача системи з правами доступу «OperatorDB»

2.5.2. Форми введення та редагування даними

Для виведення даних використано тільки одну форму, яка змінюється в ході керування через меню. Також до кожного з типів даних застосовуємо відповідні правила валідації даних, які вказані в таблиці 2.1.

Після натискання кнопки виведення даних в меню відбуваються наступні етапи:

1. Очищення минулого листа
    a. Якщо попередня сторінка була також форма для введення, тоді видаляються тільки надписи стовбців, поля для введення, дочірні таблиці, поля для введення дочірніх таблиць.
    b. Якщо попередня сторінка була форма відображення запитів, тоді видаляються всі елементи сторінки, кнопки приховуються.
2. Отримуються наступні дані з БД:
    a. Дані з таблиці з БД за назвою таблиці
    b. Дані, щодо імен полів таблиці
    c. Дані, щодо типу даних полів
3. Завантаження всіх надписів з іменами та полів для введення
    a. Якщо дані в БД за таким іменем відсутні – автоматично завантажується пуста форма
    b. Якщо дані в БД присутні – відкривається форма, де в поля для введення підставляються значення згідно першого запису з БД
4. Якщо даних немає, або натиснута кнопка для нового запису. В поле з ключем автоматично заносить наступна цифра.

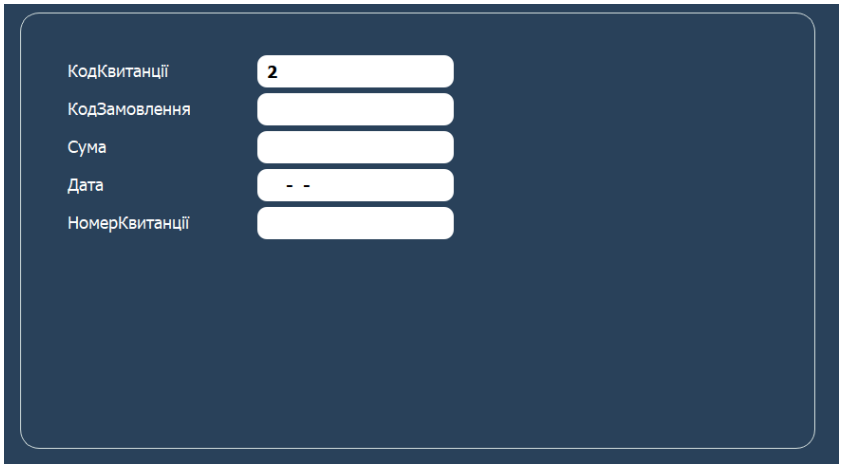


Рис. 2.5.2.1. Демонстрація автоматичного підставлення в поле ключ

5. Якщо записи в указаній таблиці присутні:
   a) Якщо тип поля дата – створюємо поле для введення календар
   b) Якщо це інший тип даних – створюємо звичайне поле для введення
   c) Якщо це поле, назва якого розпочинається на «Ключ», тобто ключ до дочірньої таблиці, тоді створюємо ComboBox та добавляємо до нього записи з БД. Також розроблено можливість налаштування випадаючих списків, а саме стовбців, які присутні замість ключа.

      Алгоритм створення випадаючого списку:

      1. Дізнаємося якій таблиці належить поле з таким ключем.
      2. Витягуємо записи за отриманою назвою таблиці.
      3. Дізнаємося з інструкції, які саме колонки потрібно групувати та виводити в значеннях випадаючого списку.
      4. Добавляємо елементи та встановлюємо значення, яке було вказано в відповідному записі на формі

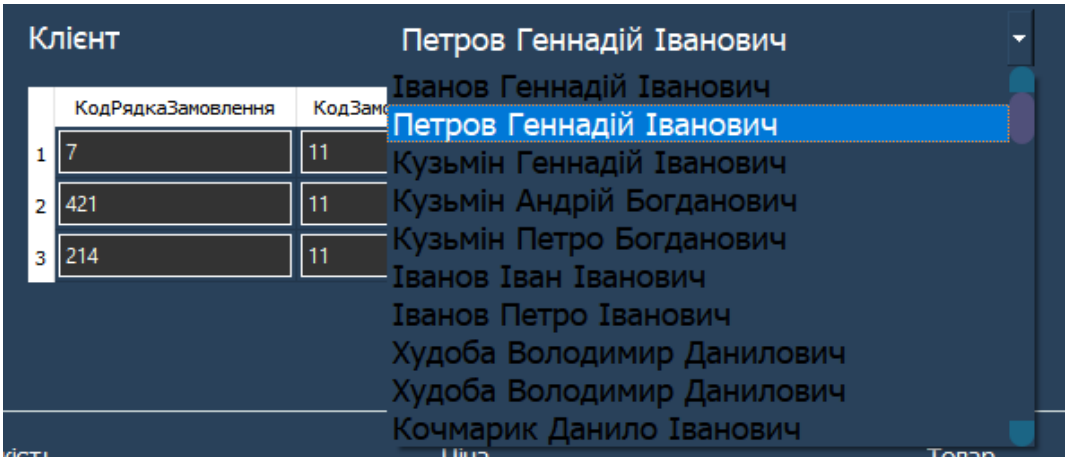


Рис. 2.5.2.2. Демонстрація випадаючого списку

d) Далі добавляємо кнопки на форму. Також, якщо присутня дочірня таблиця – відображаємо її в вигляді таблиці:

Алгоритм відображення дочірньої таблиці:

1) Створюємо таблицю, куди записуємо дані витягнуті з БД
2) Якщо даних немає – виводимо поля для введення, без таблиці
3) Якщо дані присутні – і поля, і таблицю

e) Створюємо поля для введення в дочірню таблицю та кнопки для маніпулювання даними.

Загальний вигляд форми введення та редагування даних наведений в додатках.

2.5.3. Запити

Наступний крок це розроблення можливості відображення запитів. В системі потрібно передбачити декілька видів запитів:

1. Звичайний на вибірку даних
2. Параметричний
3. Перехресна таблиця

Запит мовою SQL та результат його дії наведені в додатках.

Для кожного з виду розроблений наступний метод відображення:

Алгоритм дій:

a. Зчитування з БД структури запиту.
b. Перевірка виду запиту
    i. Якщо це звичайний запит:
        1. Відображення полів та заголовків до них, відповідно до даних з інструкцій
    ii. Якщо це параметричний запит:
        1. Створюємо віджет з кнопкою та полем для параметру на основі інструкцій до запиту.
        2. Поле для введення може бути як і звичайним полем для введення даних, або календарем, або випадаючим списком з відповідним набором даних.

3. Далі систему зчитує дані з поля для введення та підставляє до запиту в вказаному місці.
4. Виконується запит та відображається на формі згідно типу даних та заголовків до полів.

iii. Якщо це перехресна таблиця:

1. Фільтруються дані згідно прописаних інструкцій
2. Користувачу також може бути запропоновано створити сумуючі стовбці в разі необхідності.
3. Відображення даних забезпечено тільки в вигляді таблиці.

c. Якщо дані відсутні за вказаними параметрами, тоді надається користувачу відповідне повідомлення.

Також до будь якого набору даних, хоть дані з звичайної таблиці, хоть дані з запиту розроблена можливість відображення даних в вигляді таблиці.

Приклади такого відображення наведені в додатках.

Результат виконання запитів та запит мовою SQL наведені в додатках.

2.5.4. Імпорт та експорт Excel

Імпорт та експорт даних до таблиць Excel забезпечує бібліотека openpyxl. Для експорту даних розроблено функцію, яка відображає інформацію на листі Excel згідно параметрів, які надійшли. Для роботи функції необхідно мати наступні параметри:

- Назву листа
- Набір даних
- Набір заголовків до даних
- Тип даних

```
def createFile(name, data, header, type):
    wb = Workbook()
    ws = wb.active
    ws.title = str(name)
    qq = 1
    size = []
    print(header)
    for ii in header:
        ws.cell(1, qq, ii)
        qq += 1
    i = 2
    for row in data:
        j = 1
        for col in row:...
        i += 1

    path = QtWidgets.QFileDialog.getSaveFileName()[0]
    wb.save(filename=path)
```

Рис. 2.5.4.1. Демонстрація коду функції, яка забезпечує експорт до Excel

Для імпорту даних розроблено можливість користувачем вибору шляху до файлу, далі система запропонує вибрати йому стовбці та строки для заповлення, при цьому вибір є тільки серед стовбців, в яких містяться дані.

```
for i in self.dataHead:
    if self.dataHead.index(i) == 0:
        continue
    ent = QtWidgets.QComboBox()
    for i in range(1, self.sheet_ranges.max_column+1):
        ent.addItem(chr(i + 64))
    self.vboxes[0].addWidget(ent)
    self.ents.append(ent)
```

Рис. 2.5.4.2. Демонстрація коду функції, яка забезпечує фільтрування доступних стовбців

```
def upload(self):
    self.x = int(self.otdo[1].text())
    self.y = int(self.otdo[0].text())
    for row in range(self.x, self.y):
        Arr2 = ['0']
        string = ''
        for item in self.ents:
            Arr2.append(self.sheet_ranges[row][ord(item.currentText())-65].value)
        #query
        main.insert(Arr2, self.tablename, main.getType(self.tablename))
        Arr2.clear()
```

Рис. 2.5.4.3. Демонстрація коду функції, яка забезпечує імпорт до Excel

Результат роботи імпорту та експорт можливо побачити в інструкції користувача та в додатках до цієї роботи.

.

2.5.5. Фільтрація даних

Для виконання цього завдання було розроблено модуль, який генерує поля для пошуку самостійно, на основі типу даних та стовбців таблиць. Також заздалегідь продумано різні методи фільтрації. (Від значення 1 до значення2, при цьому є можливість вводити тільки одне значення; Є можливість вибору між параметром and / or)

Якщо вже відкрито вікно фільтрації, то відкрити повторно без закриття не вийду.

Методи фільтрації:

Алгоритм фільтрації:

Зчитування натиснення на кнопку

Якщо вікно фільтра не відкрито:

1. Запустити відображення вікна фільтра, куди передати значення заголовків полів та їх тип даних
2. Після цього створюється клас для відображення вікна

3. На форму вікна додаються надписи та елементи введення на основі типу даних (для кожного типу даних застосовані правила валідації)
4. Також кнопки:

Якщо вікно фільтра відкрито та натискається кнопка «Застосувати»:

1. Отримуємо параметри з форми:
2. Фільтруємо дані. В функцію також не забуваємо передати дані про стан кнопок.

Якщо кнопка «Від/до» вимкнена та параметр «AND», тобто звірка на точну копію параметра.

a) Проходимось по кожному запису з масиву переданих записів
b) Якщо хоч один параметр не співпадає з записом, то цей запис не додається в новий масив, який і буде відображено на формі.

Якщо кнопка «Від/до» вимкнена та параметр «OR», тобто звірка на хоча б одне співпадіння.

a) Проходимось по кожному запису з масиву переданих записів
b) Якщо хоч один параметр співпадає з записом, то цей запис додається в новий масив, який і буде відображено на формі.

Якщо параметр «Від/до» застосований та параметр «OR», тобто звірка на хоча б одне співпадіння.

a) Звіряємо який саме параметр нам передано (які поля заповнені, перше та друге, просто друге, просто перше, або поле пусте)
b) Якщо обидва поля заповнені

- Визначаємо тип даних цього параметру:
- І якщо хоча б одне поле відповідає цьому параметру, то запис додається в новий масив.

c) Якщо одне з полів пусте, а інше має вказаний параметр

- Визначаємо тип даних цього параметру:
- Якщо це значення більше (в випадку коли вказане значення «від») або менше (в випадку коли вказане значення «до»), тоді добавляємо цей запис в новий масив.

d) Якщо поля пусті пропускаємо цей параметр

Якщо параметр «Від/до» застосований та параметр «AND», тобто звірка на повну копію.

a) Звіряємо який саме параметр нам передано (які поля заповнені, перше та друге, просто друге, просто перше, або поле пусте)
b) Якщо хоча б одним стовбець не співпадає з введеними параметрами, то воно не додається до нового масиву. При цьому значення, окрім тексту, проходить саме перевірку на правило від та до

2.5.6. Прогнозування

Прогнозування розроблено на основі аналізу часових рядів.

Етапи аналізу:

1. Вибірка даних з БД

```
q = "select Товар.НазваТовару, extract(quarter from Замовлення.ДатаЗамовлення) as quarter, extract(year from Замовлення.Да
    " from Замовлення inner join (РядокЗамовлення inner join Товар on Товар.КодТовару = РядокЗамовлення.КодТовару) on Замо
    " group by quarter, yyyy, Товар.КодТовару" \
    " having Товар.НазваТовару = '"
q += str(lastValue)
q += "' order by Товар.КодТовару, yyyy, quarter"
dataSelect = getother(q)
```

Рис. 2.5.6.1. Вибірка даних з БД для аналізу часового ряду

2. Знаходження центрального ковзного середнього

```
for i in range(len(kovser)):
    Arr = []
    if i == 0:
        continue
    avg = (kovser[i]+kovser[i-1])/2
    if dataSelect[i+1][1] == 1:
        Arr1.append(dataSelect[i+1][3]-avg)
    elif dataSelect[i+1][1] == 2:
        Arr2.append(dataSelect[i+1][3]-avg)
    elif dataSelect[i+1][1] == 3:
        Arr3.append(dataSelect[i+1][3]-avg)
    elif dataSelect[i+1][1] == 4:
        Arr4.append(dataSelect[i+1][3]-avg)
    Arr.append(dataSelect[i+1][1])
    Arr.append(dataSelect[i+1][2])
    Arr.append(avg)
    centKovSer.append(Arr)
```

Рис. 2.5.6.2. Знаходження центрального ковзного середнього

3. Оцінка сезонної компоненти

```
res3 = 0
for i in Arr3:
    res3 += i
res3 /= len(Arr3)
res4 = 0
for i in Arr4:
    res4 += i
res4 /= len(Arr4)
sumRes = (res1 + res2 + res3 + res4)/4
res1 -= sumRes
res2 -= sumRes
res3 -= sumRes
res4 -= sumRes
seasonKop.append(res1)
seasonKop.append(res2)
seasonKop.append(res3)
seasonKop.append(res4)
```

Рис. 2.5.6.3. Оцінка сезонної компоненти

4. Формування десезоналізованих даних продажу

```
seasonData = []
for i in dataSelect:
    ArrS = []
    for j in range(len(i)):
        if j == 3:
            ArrS.append(float(i[j])-float(seasonKop[int(i[1])-1]))
            continue
        ArrS.append(i[j])
    seasonData.append(ArrS)
```

Рис. 2.5.6.4. Формування десезоналізованих даних

5. Розрахування тренду

```
for i in range(len(dataSelect)):
    x += i+1
    xx += ((i + 1) * (i + 1))
    y += seasonData[i][3]
    xy += ((i+1)*seasonData[i][3])
b = (len(dataSelect)*xy - x*y) / (len(dataSelect)*xx - (x*x))
a = (y/len(dataSelect)) - (b*x/len(dataSelect))
trend = []
for i in seasonData:
    ArrT = []
    ArrT.append('Тренд')
    ArrT.append(i[1])
    ArrT.append(i[2])
    ArrT.append((int(a) + int(b)*(seasonData.index(i)+1)))
    trend.append(ArrT)
```

Рис. 2.5.6.5. Розрахунок тренду

6. Прогнозування майбутніх продаж

```
for i in range(4):
    q1 = []
    q = (int(a) + int(b)*(len(seasonData)+i+1))
    pos = seasonData[len(seasonData)-1][1]+i+1
    count = 0
    while pos > 4:
        pos -= 4
        count += 1
    q += seasonKop[int(pos)-1]
    year = seasonData[len(seasonData)-1][2]
    q1.append('Прогнозування')
    q1.append(pos)
    q1.append(year+count)
    q1.append(q)
    prog.append(q1)
```

Рис. 2.5.6.6. Код для прогнозування даних

7. Встановлення адекватності та достовірності аналізу, розраховуючи середню квадратичну помилку.

```
pom2 = 0.00
for iTrend in trend:
    tr = 0.00
    tr += float(iTrend[3])
    tr -= float(seasonKop[int(iTrend[1]) - 1])
    tr -= dataSelect[len(iTrend)-1][3]
    pom += abs(tr)
    pom2 += (tr*tr)
```

Рис. 2.5.6.7. Розрахунок відхилення

```
"Достовірність ~ " + (str(100 - (sqrt(pom2) / len(dataSelect)))) + " %")
```

Рис. 2.5.6.8. Розрахунок середню квадратичної помилки

8. Виведення на графік всіх розрахунків.

```
from math import sqrt
q = ts.one(mainForm.frame_18, d, 1, 0, 2, "Достовірність ~ " + (str(100 - (sqrt(pom2) / len(dataSelect)))) + " %")
mainForm.pushButton_3.setEnabled(True)
```

Рис. 2.5.6.9. Виведення даних в графічному вигляді

2.5.7. Генерація QR-code

Для того щоб забезпечити зручне виведення даних з графіків, окрім виведення даних в таблицях Excel та можливості завантаження графіка картинкою, доцільно зробити генерацію QR, щоб можливо було імпортувати дані до інших пристроїв.

Генерувати QR коди потрібно при натиснені на графік, для цього потрібно відслідковувати натиснення на графік та саме на яких графік.

```
class LineSeries(QLineSeries):
    def __init__(self, *args, **kwargs):
        QLineSeries.__init__(self, *args, **kwargs)
        self.pressed.connect(self.on_pressed)
        global q
        self.name = q
        q += 1

    def on_pressed(self, point):
        qrGenerate.setData(self.name)
```

Рис. 2.5.7.1. Створення класу лінії функції, який забезпечує потрібні функції для відображення QR кодів

Після натиснення потрібно визначити, по якому саме графіку було натиснуто.

```
def setData(number):
    global data
    if currentWindow() == "DirectorForm":...
    elif currentWindow() == "DBForm":...
    else:
        pass
```

Рис. 2.5.7.2. Код для визначення відкритого вікна

```
if test2.ui.stackedWidget.currentIndex() == 1:
    if number == 0:
        d2 = prognoz.start.convertToQuarter(prognoz.start.prog)
        data = str(d2[0][0])
        data += " \n"
        for i in d2:
            for j in range(1, len(i)):
                data += str(i[j])
                data += " "
            data += "\n"
        showImg()
```

Рис. 2.5.7.3. Отримання даних

```
def showImg():
    global img
    img = qrcode.make(data)
    w = myGrid()
    w.show()
```

Рис. 2.5.7.4. Функція для створення QR-коду

```
im = ImageQt(img).copy()
pixmap = QtGui.QPixmap.fromImage(im)
pixmap_resized = pixmap.scaled(300, 300, QtCore.Qt.KeepAspectRatio)
label = QtWidgets.QLabel()
label.setPixmap(pixmap_resized)
vbox.addWidget(label)
self.setStyleSheet("background: white")
```

Рис. 2.5.7.5. Відображення QR в віджеті

2.5.8. Генерація рахунку на оплату

Генерацію рахунку на оплату також заздалегідь продумано, до будь якого замовлення. Для цього розроблено функцію, яка відкриває створений макет в середовищі Excel та вставляє відповідні дані по коміркам.

```
def createSale(client1, client2, client3, number, data, price):
    wb = load_workbook(filename=str('D:/nuft/work/sale.xlsx'))
    sheet_ranges = wb.active
    sheet_ranges.cell(16, 2, str("Рахунок №" + str(number) + " від " + str(data)))
    sheet_ranges.cell(21, 9, str(client1 + " " + client2 + " " + client3))
    sheet_ranges.cell(28, 9, str("Замовлення №" + str(number)))
    sheet_ranges.cell(28, 43, str(price))
    sheet_ranges.cell(28, 47, str(price))
    path = QtWidgets.QFileDialog.getSaveFileName()[0]
    wb.save(filename=path)
```

Рис. 2.5.8.1. Код для створення файла з рахунком на оплату

2.6. Кабінет директора

Наступним питанням постає розробка системи для керівників інтернет-магазину. Було прийнято рішень розробити сторінку з статистичними даними проданих товарів.

Потрібно розробити і загальну статистику, і статистику по кожному з виду товару. Відповідно потрібно меню для маніпулювання даними.

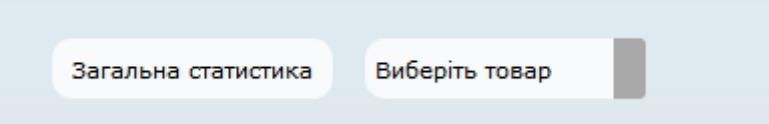


Рис. 2.6.1 Меню для кабінету директора

Забезпечена можливість відображення наступних статистичних даних:

Загальна статистика:

1. Загальна кількість продажів
2. Загальна сума продаж товару
3. Кількість клієнтів
4. Середній чек
5. Сума не реалізованого товару
6. Прибуток по місячно за поточний рік
7. Дані щодо загальної кількості доставок

8. Дані щодо компанії, які надають послуги з доставки замовлень

Відображення статистичних даних по кожному товару окремо:

1. Дані щодо кількості продажу щомісячно в поточному році
2. Дані щодо змін цін на товар
3. Дані про загальну кількість реалізованого товару
4. Дані про суму продаж вибраного товару
5. Дані про відсоток наявності товару у всіх замовлень
6. Дані про відсоток клієнтів, які купили поточний товар
7. Дані про кількість та суму нереалізованого товару

Запити та загальний вигляд форми наведенні в додатках.

Також для зручного переключення між товарами слід забезпечити відображення всіх товарів в випадаючому списку

```
def addCombo():
    data = main.getother("select КодТовару, НазваТовару from Товар")
    ui.menu2.addItem("Виберіть товар")
    ui.menu2.setItemData(0, "load")
    comboIndex = 1
    for i in data:
        ui.menu2.addItem(i[1])
        ui.menu2.setItemData(comboIndex, i[0])
        comboIndex += 1
    ui.menu2.currentTextChanged.connect(change)
```

Рис. 2.6.2. Функція для заповлення випадючого списку

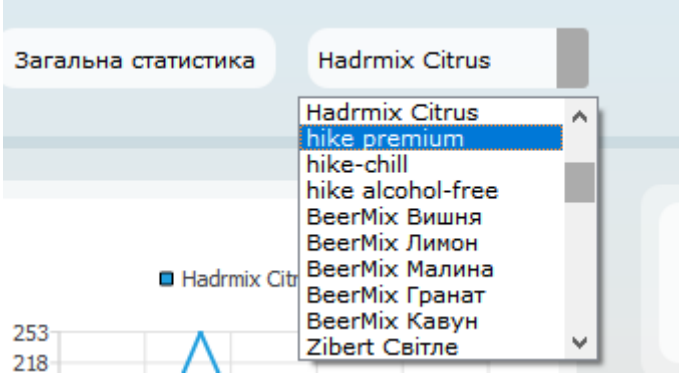


Рис. 2.6.3. Загальний вигляд випадаючого списку

2.7. Кабінет адміністратора системи

Розроблено забезпечення наступних функцій:

1. Конфігурація випадаючих списків

   Тобто всі випадючі списки в ІС для менеджерів та операторів, які відображаються поля замість ключів потрібно налаштовувати або змінювати.

```
ui.stackedWidget.setCurrentWidget(ui.page_4)
work = main.getother("select * from Інструкція2")
admForm.viewTableDB.viewtable(ui.tableWidget_2, work, main.gethead("Інструкція2"), main.getType('Інструкція2'), 'Інструкція2')
```

Рис. 2.7.1. Код для забезпечення відображення інструкцій випадаючих списків

2. Конфігурація інструкцій для відображення форм

   Тобто маніпулювання меню та створення нових запитів, звітів, та форм введення та виведення.

```
ui.stackedWidget.setCurrentWidget(ui.page_3)
work = main.getother("select * from Інструкція")
admForm.viewTableDB.viewtable(ui.tableWidget, work, main.gethead("Інструкція"), main.getType('Інструкція'), 'Інструкція')
```

Рис. 2.7.2. Код для забезпечення відображення інструкцій ІС

3. Довільні запити

   Тобто забезпечення можливості виконання будь якого запиту написаного мовою SQL.

```
def reset():
    text.clear()

def sub():
    value = main.getother(text.toPlainText())
    admForm.viewTable.viewtable(ui.tableView, value)
    reset()
```

Рис. 2.7.3. Код для забезпечення виконання довільного запиту

4. Конфігурація БД

   Тобто налаштування параметрів для відкриття з’єднання з БД.

```
def configDB():
    def save():
        config.setValue(ui.lineEdit.text(), ui.lineEdit_2.text(), ui.lineEdit_3.text(), ui.lineEdit_4.text())
    ui.stackedWidget.setCurrentWidget(ui.page)
    ui.lineEdit.setText(config.host)
    ui.lineEdit_2.setText(config.user)
    ui.lineEdit_3.setText(config.password)
    ui.lineEdit_4.setText(config.db_name)
    ui.pushButton_6.clicked.connect(save)
```

Рис. 2.7.4. Код для забезпечення зміни параметрів в БД

5. Управління користувача

   Тобто управління обліковими записи всіх користувачів ІС.

```
ui.stackedWidget.setCurrentWidget(ui.page_5)
work = main.getother("select * from Користувач")
admForm.viewTableDB.viewtable(ui.tableWork, work, main.gethead("Користувач"), main.getType('Користувач'), 'Користувач')
```

Рис. 2.7.5. Код для забезпечення відображення користувачів

2.8. Інструкція користувача

Після відкриття додатку перед користувачем з’являється вікно авторизації, де потрібно ввести ваш пароль та логін, який дасить вам ваше керівництво.

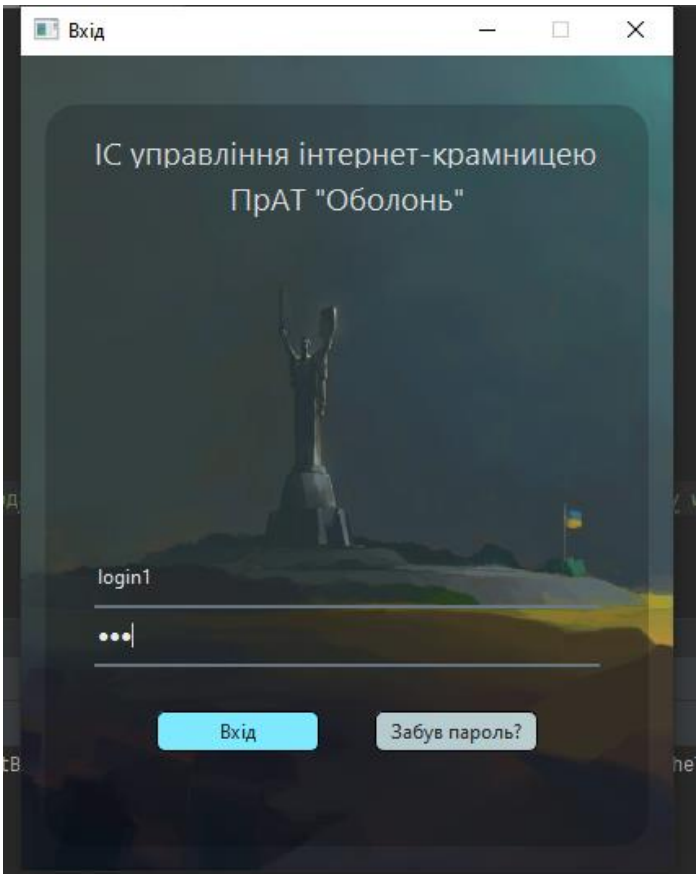


Рис. 2.8.1. Форма авторизації

Якщо пароль не вірний, або такого користувача немає вам буде показано відповідне сповіщення.

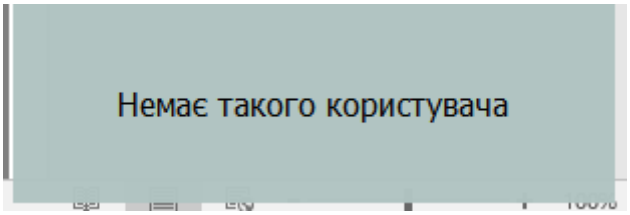


Рис. 2.8.2. Сповіщення про неправильно введені дані

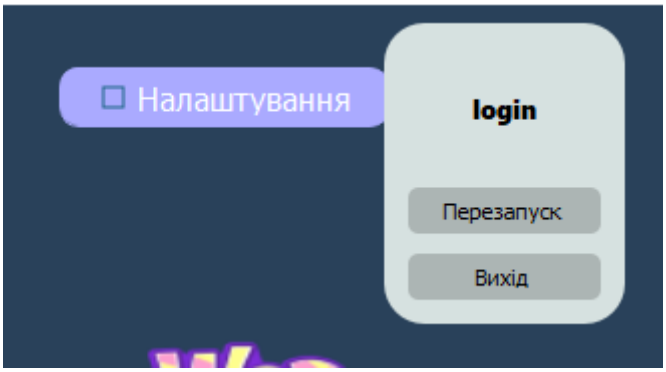
Якщо пароль та логін валідні, то вас перенаправить до головної форми тієї підсистеми до якої у вас надані права.

Якщо ви забули пароль, вам потрібно ввести свій логін в поле для логіну та натиснути кнопку «Забули пароль». Після цього запит надійде до адміністратора, і він зможе зробити вам новий пароль. При цьому також зберігається можливість входу, якщо пароль ви згадаєте.

2.8.1. ІС для операторів та менеджерів

Рис. 2.8.1.1. Загальний вигляд форми після авторизації

В правому верхньому кутку є кнопка «Налаштування». Після її натиснення відкриється меню, де можливо побачити логін під яким ви зайшли, кнопку перезавантаження системи та кнопку виходу.



2.8.1.2. Віджет для управління ІС

Якщо натиснути кнопку «Перезапуск» - відбудеться перезантаження системи. Рекомендуємо це робити, коли у нас виникла помилка, яка зазделегідь не передбачена.

Якщо натиснути кнопку «Вихід» - вас деавторизую та перекинь на форму авторизації.

В лівій частині вікна знаходиться меню. Його можно поділити на три відносних блока: форми для введення, запити до БД, звіти та прогнозування

Для введення та редагування даних в таблицях слід використовувати перший блок меню, який містить наступні кнопки:

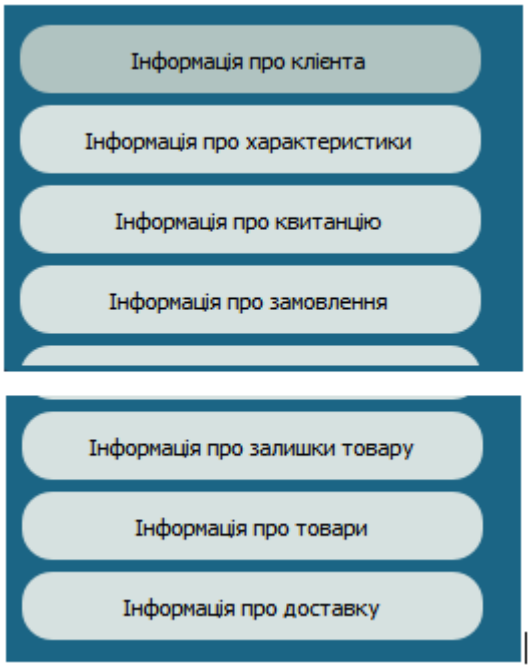


Рис. 2.8.1.3. Меню для управління форма для маніпулювання даними

Оскільки будь які записи з таблиць БД відкриваються на одній формі, то кнопки маніпулювання даними однакові на всіх формах.

В нижній частині фрейму знаходяться кнопки для управління даними.

Для переходу між записи слід натискати кнопки «Вліво» та «Вправо»

Для видалення запису слід натиснути кнопку «Видалити»

Для створення нового запису слід натиснути кнопку «Новий запис»

Після натиснення кнопки «новий запис» відобразиться пуста форма для введення, а кнопка змінить назву на «зберегти».

Для відображення фільтру слід натиснути кнопку «Фільтр».

Після натиснення кнопки «Фільтр» відобразиться форма для заповнення параметрів пошуку та налаштування типу фільтру.

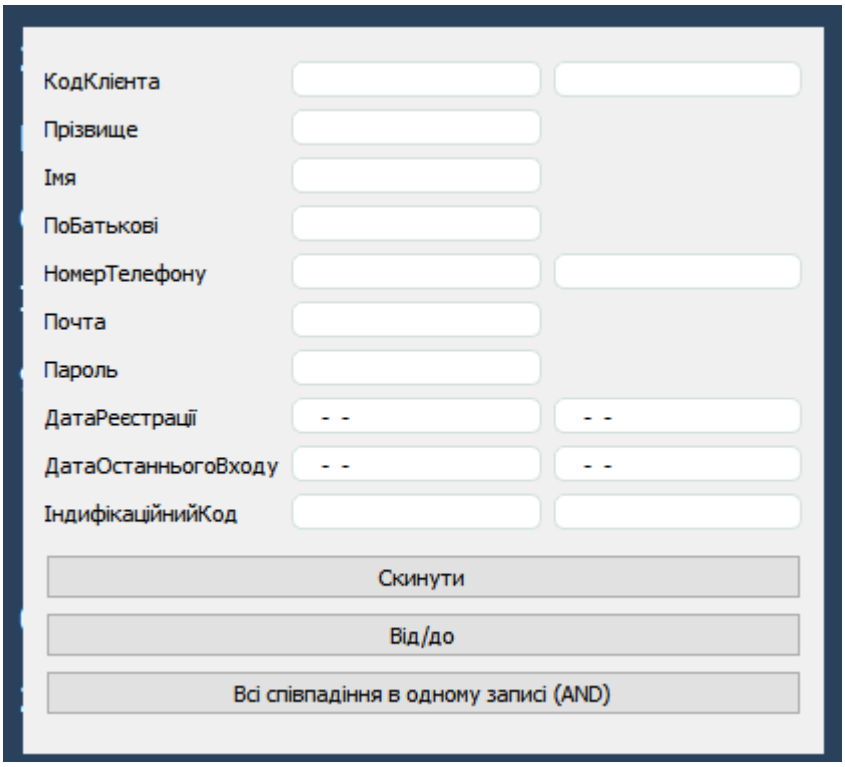


Рис. 2.8.1.5. Віджет «Фільтр» для запису про клієнта

Якщо вам потрібно шукати за декількома параметрами та ці всі параметри мають бути присутні в одному записі, то слід обрати параметр «AND»

Якщо вам потрібно шукати за декількома параметрами, та виводити запис коли в ньому є хоч одне співпадіння з параметром, то слід обрати параметр «OR»

Якщо вам потрібно шукати за параметром від або до, або від до, слід заповинити відповідні поля та обрати відповідний параметр.

Якщо нам потрібно знайти записи, параметри яких повністю співпадають з записом слід натиснути відповідну кнопку. Якщо на кнопці надпис з потрібним нам параметром значить він буде застосований. В даному випадку це надпис «Всі співпадіння в одному записі (AND)».

Приклад: Якщо нам потрібно знайти запис, де ключ дорівнює 2, то якщо ми введемо інші параметри, які не відповідають даним в цьому запису, тоді записів не буде знайдено. При цьому не обов'язково вводити всі поля, система автоматично розпізнає, що поле не заповнено і не буде застосовувати фільтр.

Якщо нам потрібно знайти записи, які співпадають хоча б з одним параметром, то даному випадку надпис на кнопці «Хоча б одне співпадіння (OR)».

Якщо нам потрібно застосувати параметр «Від значення1 до значення2», тоді нам потрібно натиснути відповідну кнопку. Якщо на кнопці надпис «Від/до» та з’явилось додаткове поле для введення, тоді ми в потрібному режимі. При цьому якщо тип даних «Текст», то немає додаткового поля, та фільтрація відбувається за повним співпадінням.

Приклад: Застосовуємо фільтр, де код доставки від 5 до 10. При цьому не важливо який параметр вказаний в виборі «AND/OR», оскільки тільки один параметр.

Залишаємо такий же параметр, але також шукаємо ще записи, де вартість доставки більше 101. При цьому не важливо щоб запис відповідала кожному з параметрів, достатньо й одного.

Як вже говорив фільтрація розроблена таким способом, що застосувати можливо до будь-якого набору даних з БД, якщо є такі параметри, як тип даних та назва стовбця. Застосування фільтру можливо також для запитів.

Для оновлення даних, які ви змінили на формі слід натиснути кнопку оновити.

Якщо ви не змінювали дані та натиснули кнопку «Оновити» відображується відповідне повідомлення.

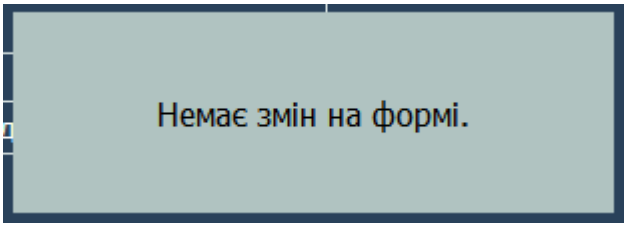


Рис. 2.8.1.6. Сповіщення про відсутність змін на формі

Якщо ви змінили дані та натиснули кнопку «Оновити» відображується відповідне повідомлення.

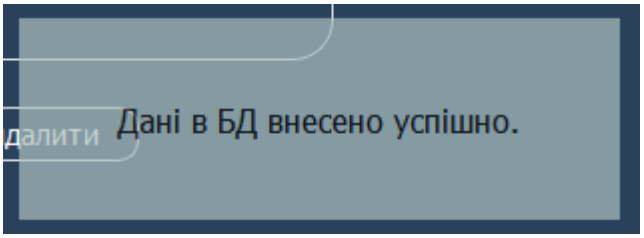


Рис. 2.8.1.7. Сповіщення про зміну запису в БД

Якщо ви не пройшли перевірки на правила валідації або під час виконання сталася помилка, вам надійде відповідне повідомлення.

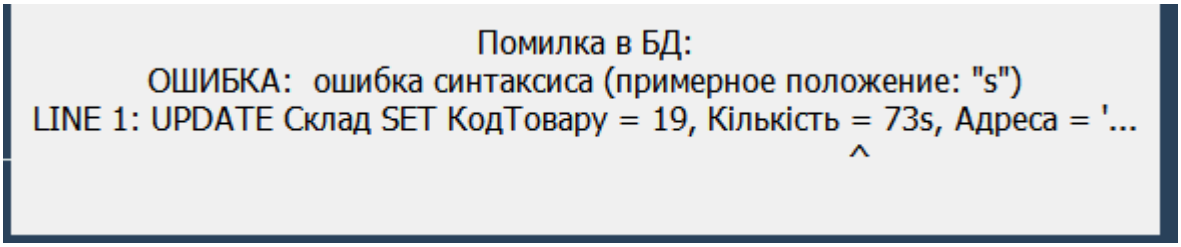


Рис. 2.8.1.8. Сповіщення про неправильно введені дані

Якщо натиснути на кнопку «Імпорт» відкриється вікно для вибору Excel-файла.

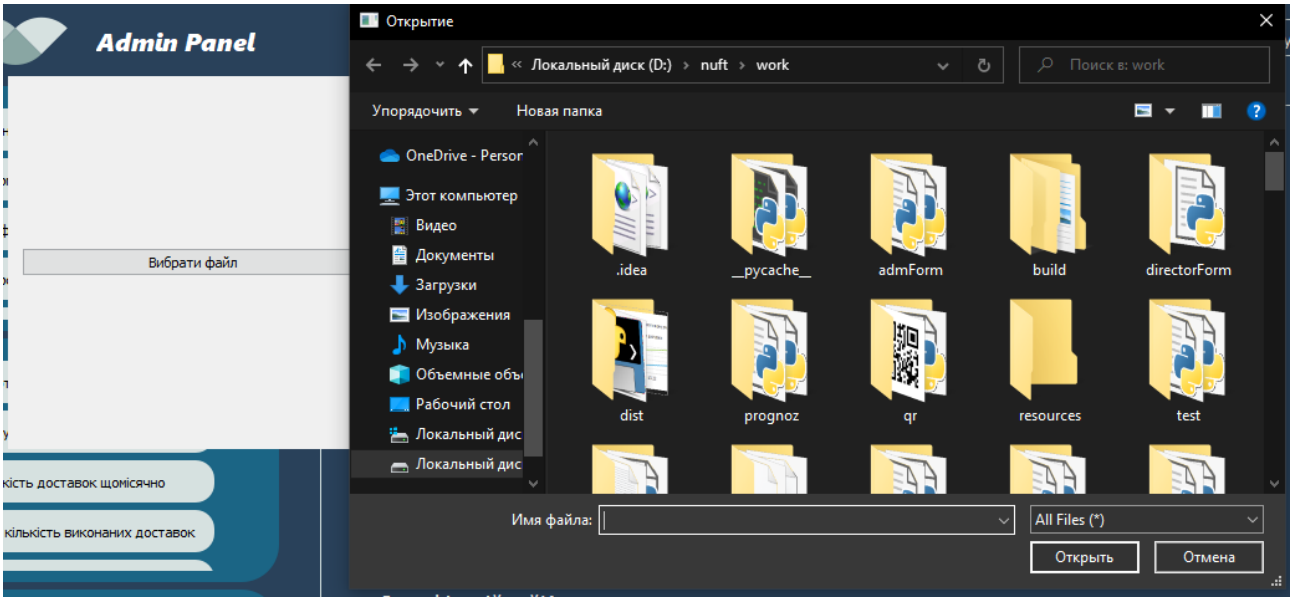


Рис. 2.8.1.9. Вибір шляху

Після вибору файла вам запропонує вибрати стовбці та ввести значення рядків, які потрібно імпортувати.

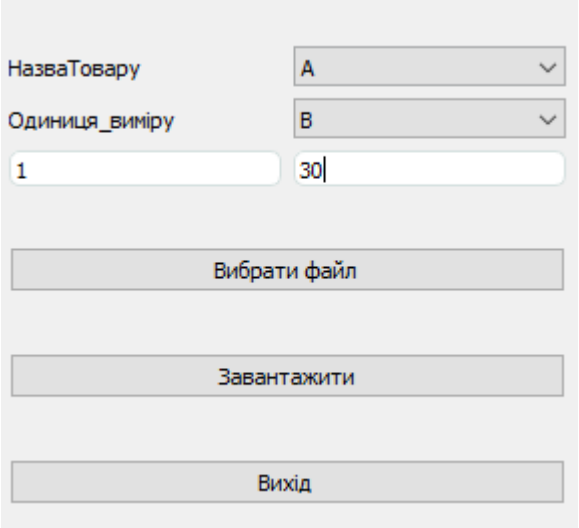


Рис. 2.8.1.10. Вибір параметрів імпорту даних

Якщо натиснути кнопку «Вибрати файл» повторно вибирається файл, видаляючи інформацію про поточний файл.

Якщо натиснути кнопку «Завантажити», то всі записи по строково будуть надсилатися до БД. Зверніть увагу, якщо великий об’єм інформації, то час який буде затрачено значно великий, слідкуйте за сповіщеннями в правому кутку.

Якщо натиснути кнопку «Вихід» - закриється поточне вікно «Імпорту».

Якщо вам потрібно видалити поточний запис – слід натиснути кнопку «Видалити»

Для перегляду даних в режимі «Таблиця» слід натиснути кнопку «Таблиця» в верхній частині фрейму.

Якщо в режимі перегляду «Таблиця» змінити запис надійде повідомлення про можливість оновлення бази даних.

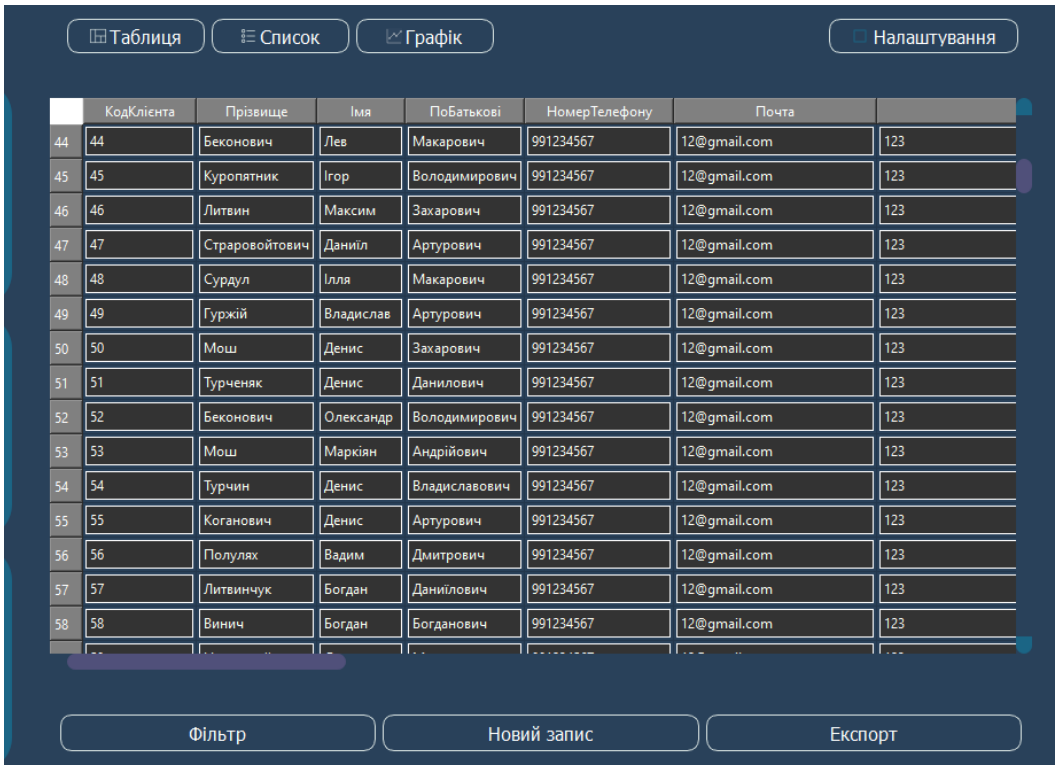


Рис. 2.8.1.11. Форма відображення «Таблиця»

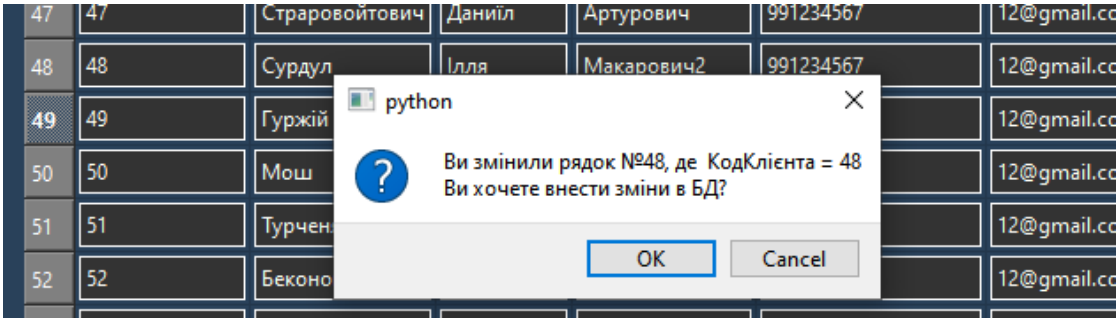


Рис. 2.8.1.12. Сповіщення про зміну даних

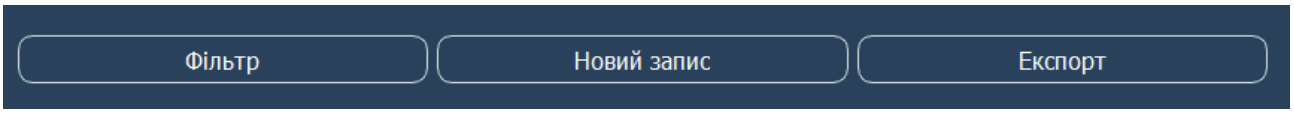


Рис. 2.8.1.13. Кнопки на формі відображення «Таблиця»

Якщо в режимі перегляду таблиця натиснути кнопку «Фільтр» - відкриється таке ж вікно для фільтрації, яке описано вище. Дивіться Рис. 2.8.1.5.

Якщо натиснути кнопку «Новий запис» відкриється форма з пустими полями в формі для виведення «Список».

Якщо натиснути кнопку «Експорт», то відкриється вікно для вибору шляху зберігання файла в форматі Excel – таблиці.

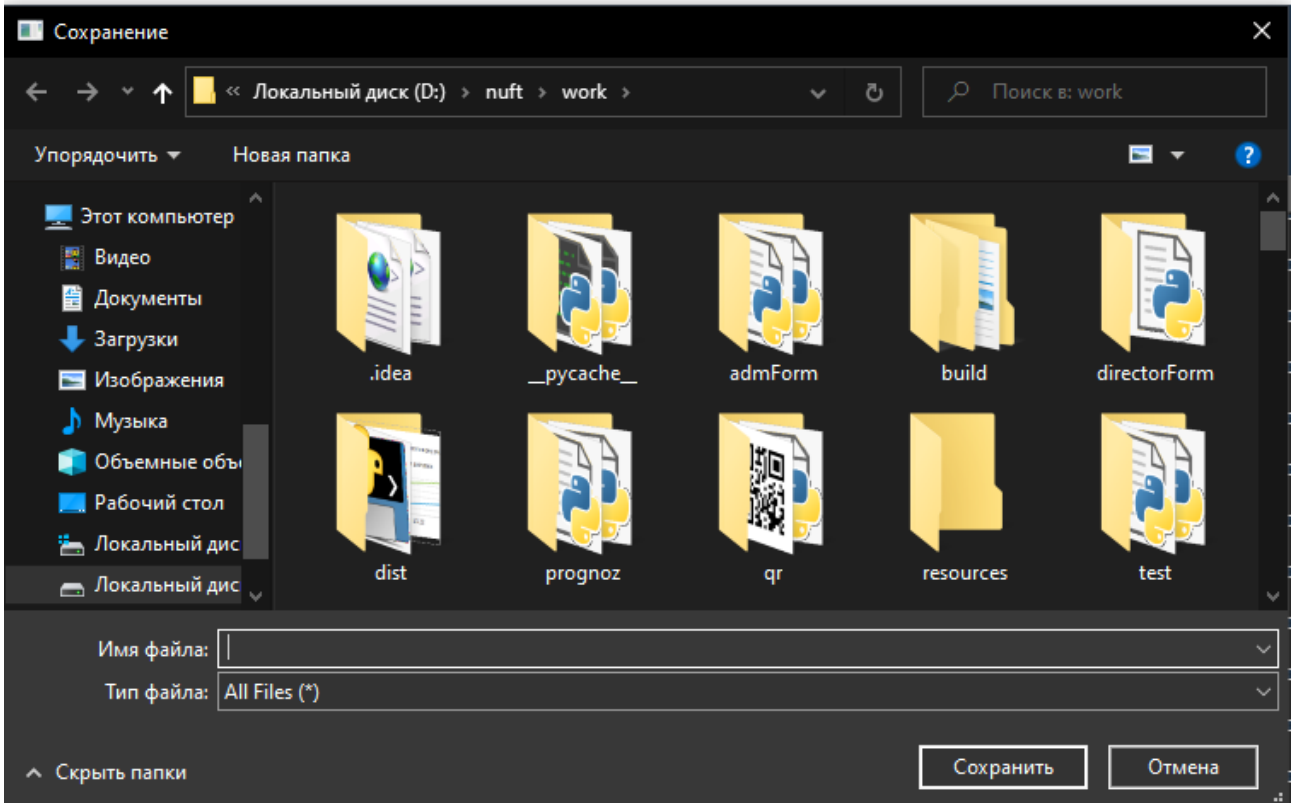


Рис. 2.8.1.14. Вибір шляху для зберігання файла

Результат експорту можливо побачити в додатках до цієї роботи.

На поточний момент прописані інструкції для відображення наступної інформації:

1. Інформація про клієнта:

   Містить інформацію про клієнта. ПІБ, Телефон, Почта, Пароль, ДатаРеєстрації, ДатаОстанньогоВходу, Індифікаційний номер. Інформація про характеристики:

Містить поле для вибору товару, поле для введення назви характеристики та значення характеристики.

2. Інформація про квитанцію:

Поле замовлення містить дані щодо номеру замовлення та загальної суми замовлення. Також на формі розміщенні дані щодо суми, дати та номера квитанції. Для заповнення слід орієнтуватись на надписи біля полів введення.

3. Інформація про замовлення:

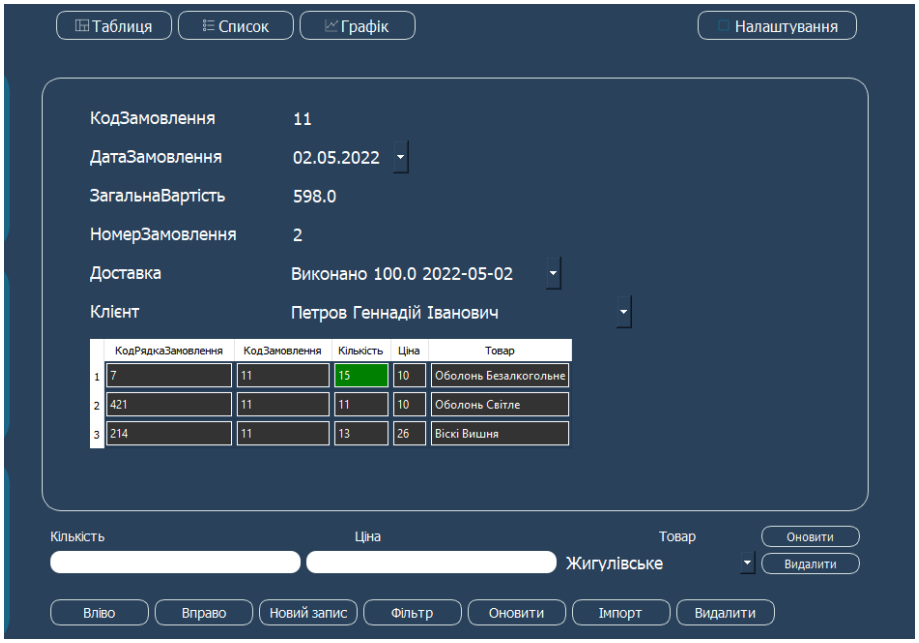


Рис. 2.8.1.15. Форма відображення таблиці «Замовлення»

Окрім звичайних полів для введення на формі розміщено дочірню таблицю Рядок замовлення.

Щоб добавити новий запис потрібно ввести дані в відповідних полях на натиснути кнопку оновити.

Якщо натиснути на будь яку клітинку в таблиці в поля для введення автоматично занесуться дані з рядку, по якому було натиснуто.

Якщо натиснути на кнопку «Оновити» дані оновлюються в таблиці.

Для видалення запису з дочірньої таблиці слід натиснути кнопку «Видалити».

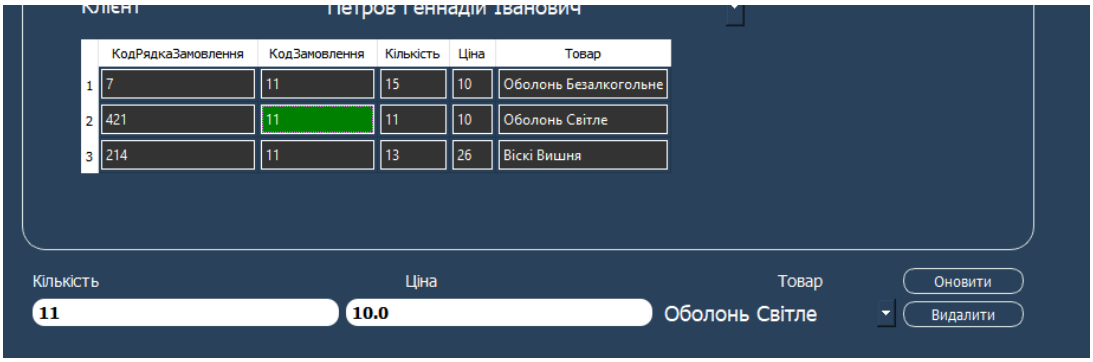


Рис. 2.8.1.16. Поля для введення та кнопки для маніпулювання даними

При цьому якщо ви змініте якийсь рядок в замовлені сума замовлення автоматично розрахується на формі, але для зміни суми замовлення в БД слід натиснути кнопку «Оновити» в нижній частині фрейму.

4. Інформація про залишку товару:

Містить інформацію щодо товару та залишку його на складах підприємства

Окрім звичайних полів для введення на формі розміщено дочірню таблицю Склад.

Управління таким видом відображення описано вище, в пункті 4 поточного списку.

5. Інформація про товари:

Містить інформацію щодо товару та прайслисту до нього

Окрім звичайних полів для введення на формі розміщено дочірню таблицю Прайслист.

Управління таким видом відображення описано вище, в пункті 4 поточного списку.

6. Інформація про доставку:

Містить інформацію про доставку

Окрім звичайних полів для введення на формі розміщено дочірню таблицю Самовивіз та Курєрська доставка.

Якщо тип доставки вказаний «Самовивіз» - відображається дочірній запис з таблиці «Самовивіз»

Якщо тип доставки вказаний «Курєрська доставка» - відображається дочірній запис з таблиці «Курєрська доставка»

Більше одного запису внести до БД не вийде.

Якщо запису в дочірніх таблицях немає, тоді відображаються поля введення.

Загальний вигляд форм можна побачити в додатках до цієї роботи.

Для відображення запитів слід в меню натиснути кнопки в другому блоці.

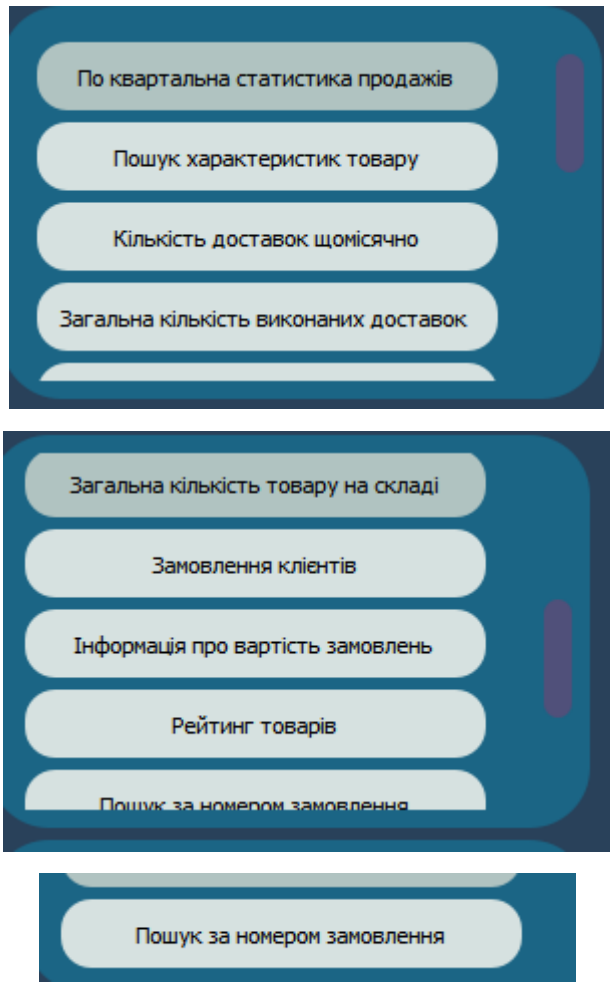


Рис. 2.8.1.17. Меню «Запити»

1. Поквартальна статистика продажів товару

Після натиснення кнопки відобразиться форма для вибору товару, по якому слід зробити пошук.

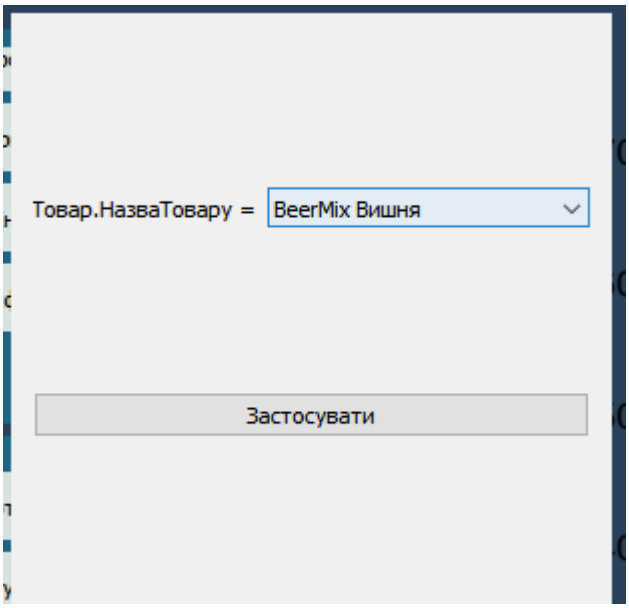


Рис. 2.8.1.18. Вибір параметру «НазваТовару»

Результати запиту в режимі перегляду «Таблиця»:

	НазваТовару	Квартал	Рік	Кількістьпроданноготовару
1	BeerMix Вишня	1	2019	524
2	BeerMix Вишня	2	2019	568
3	BeerMix Вишня	3	2019	654
4	BeerMix Вишня	4	2019	289
5	BeerMix Вишня	1	2020	575
6	BeerMix Вишня	2	2020	690
7	BeerMix Вишня	3	2020	817
8	BeerMix Вишня	4	2020	232
9	BeerMix Вишня	1	2021	622
10	BeerMix Вишня	2	2021	543
11	BeerMix Вишня	3	2021	811
12	BeerMix Вишня	4	2021	882
13	BeerMix Вишня	1	2022	741
14	BeerMix Вишня	2	2022	342

Рис. 2.8.1.19. Форма відображення «таблиця»

Пошук характеристик товару:

Після натиснення кнопки відобразиться форма для вибору товару, по якому слід зробити пошук.

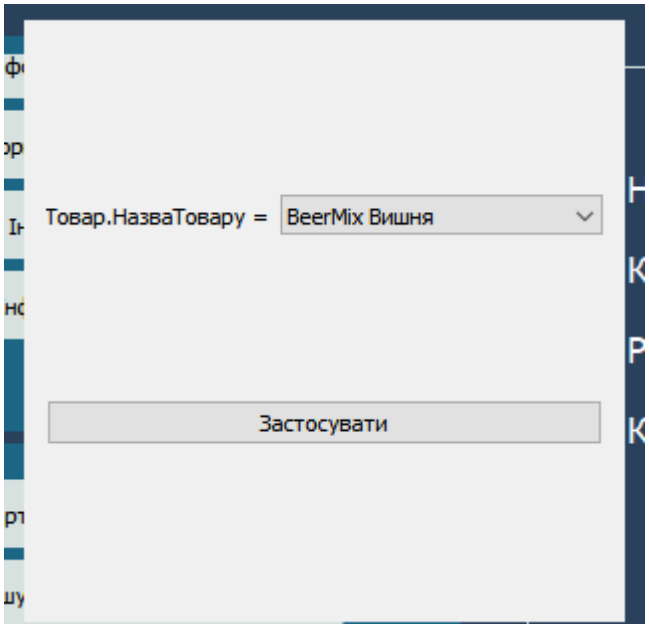


Рис. 2.8.1.20. Форма для введення параметру «НазваТовару»

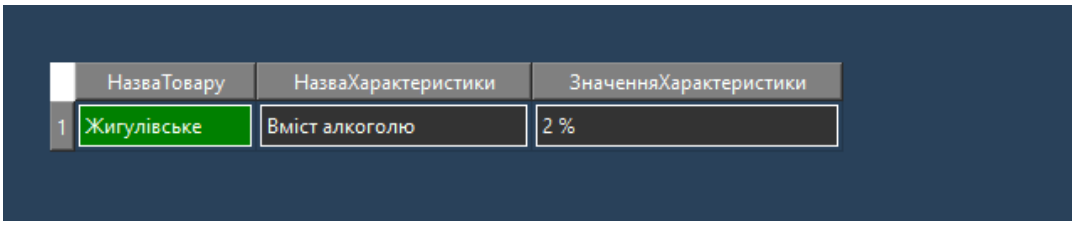
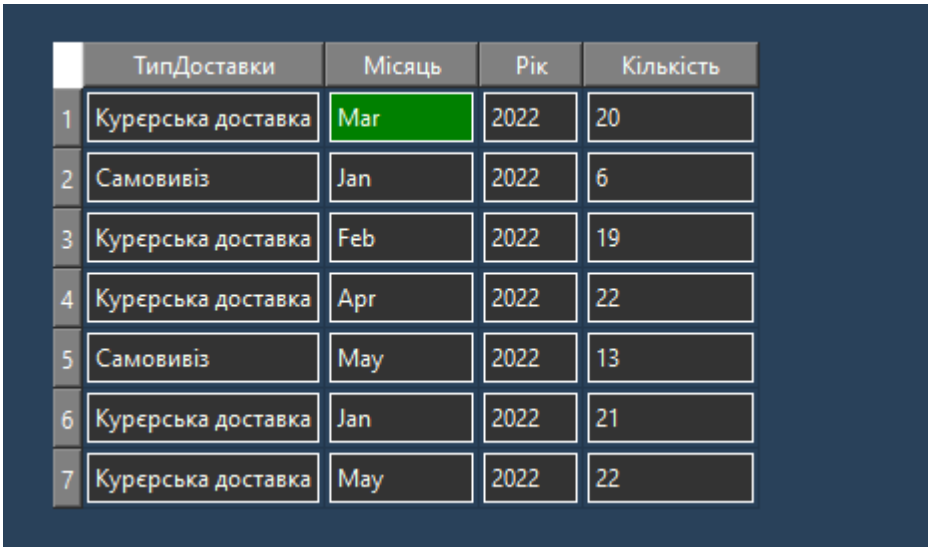


Рис. 2.8.1.21. Форма для виведення даних запиту про характеристики

Кількість доставок щомісячно:

Після натиснення кнопки відкриється форма з даними про доставки, группуючи їх помісячно:

Рис. 2.8.1.22. Форма відображення «Список»



	ТипДоставки	Місяць	Рік	Кількість
1	Курєрська доставка	Mar	2022	20
2	Самовивіз	Jan	2022	6
3	Курєрська доставка	Feb	2022	19
4	Курєрська доставка	Apr	2022	22
5	Самовивіз	May	2022	13
6	Курєрська доставка	Jan	2022	21
7	Курєрська доставка	May	2022	22

Рис. 2.8.1.23. Форма відображення «Таблиця»

Загальна кількість доставок за період:

Показує загальну кількість доставок за вказаний період по кожному типу доставки

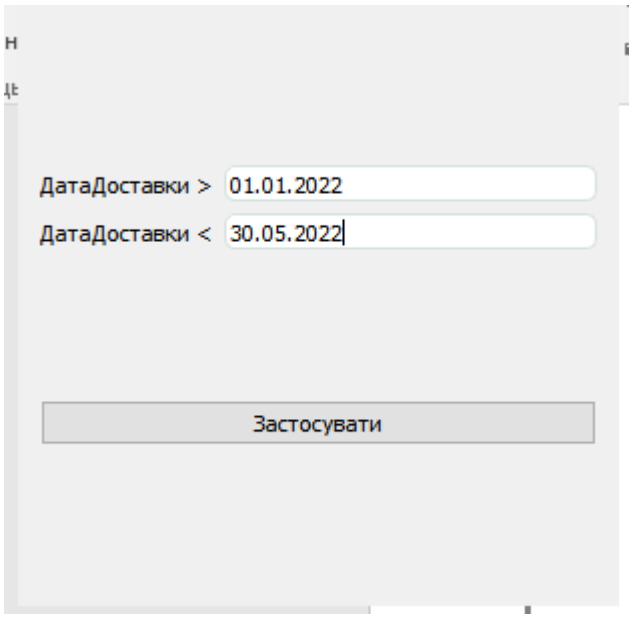


Рис. 2.8.1.25. Введення параметру дати

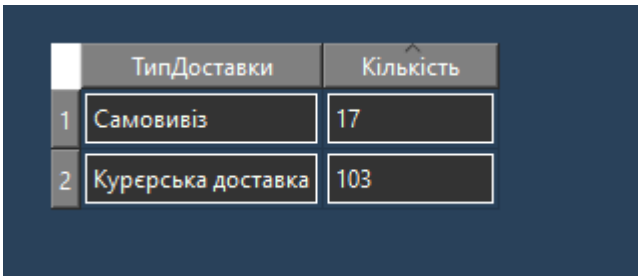


Рис. 2.8.1.26. Результат виконання запиту в формі відображення «Таблиця»

Загальна кількість товару на складі:

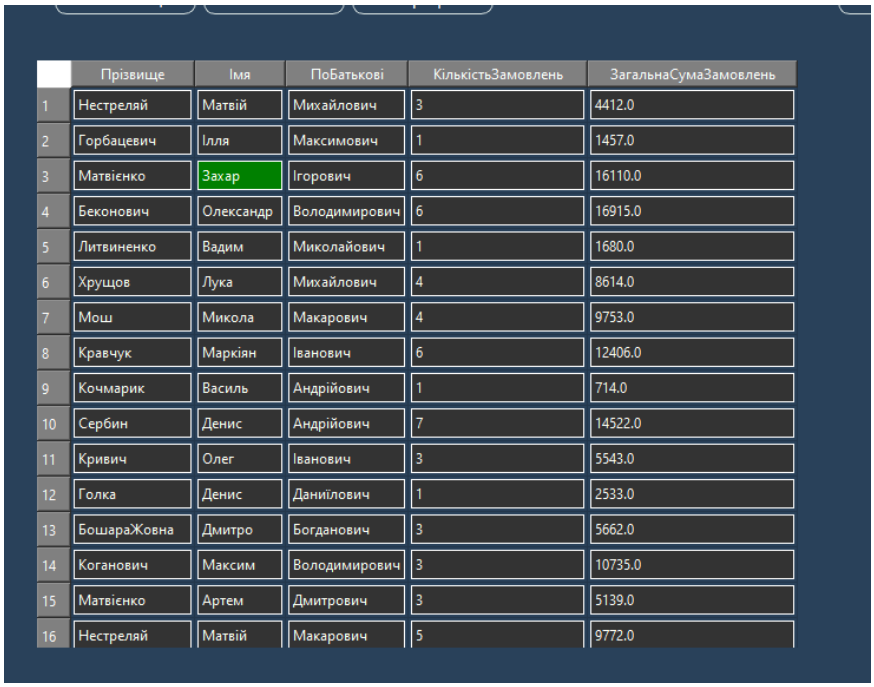
Показує дані щодо суми кількості товару на кожному складі:

	Адреса	Назватовару	Сума
1	м. Київ, пр. Степана Бандери 11	Живчик Яблуко негазований	13
2	м. Київ, пр. Степана Бандери 11	Оболонь Трофейне	96
3	м. Київ, пр. Степана Бандери 11	Джин-тонік	8
4	м. Київ, пр. Степана Бандери 11	Живчик Яблуко	26
5	1	Живчик Лимон	76
6	1	BeerMix Гранат	41
7	м. Київ, пр. Степана Бандери 11	BeerMix Гранат	19
8	м. Київ, пр. Степана Бандери 11	Zlata Praha	94
9	1	Охтирська	77
10	м. Київ, пр. Степана Бандери 11	hike-chill	51
11	1	Живчик Яблуко	52
12	м. Київ, пр. Степана Бандери 11	Рідний Шубін Світле	88
13	м. Київ, пр. Степана Бандери 11	BeerMix Малина	65
14	1	Оболонь Безалкогольне	28
15	1	Оболонь Трофейне	11
16	1	Zibert Weissbier	68

Рис. 2.8.1.27. Форма відображення запиту про кількість товару на складі

Замовлення клієнтів:

Відображає дані щодо кількості замовлень клієнтів та загальну суму покупок:



	Прізвище	Ім'я	ПоБатькові	КількістьЗамовлень	ЗагальнаСумаЗамовлень
1	Нестреляй	Матвій	Михайлович	3	4412.0
2	Горбацевич	Ілля	Максимович	1	1457.0
3	Матвієнко	Захар	Ігорович	6	16110.0
4	Беконович	Олександр	Володимирович	6	16915.0
5	Литвиненко	Вадим	Миколайович	1	1680.0
6	Хрущов	Лука	Михайлович	4	8614.0
7	Мош	Микола	Макарович	4	9753.0
8	Кравчук	Маркіян	Іванович	6	12406.0
9	Кочмарик	Василь	Андрійович	1	714.0
10	Сербин	Денис	Андрійович	7	14522.0
11	Кривич	Олег	Іванович	3	5543.0
12	Голка	Денис	Даниїлович	1	2533.0
13	БошараЖовна	Дмитро	Богданович	3	5662.0
14	Коганович	Максим	Володимирович	3	10735.0
15	Матвієнко	Артем	Дмитрович	3	5139.0
16	Нестреляй	Матвій	Макарович	5	9772.0

Рис. 2.8.1.28. Результат виконання запиту про замовлення клієнтів

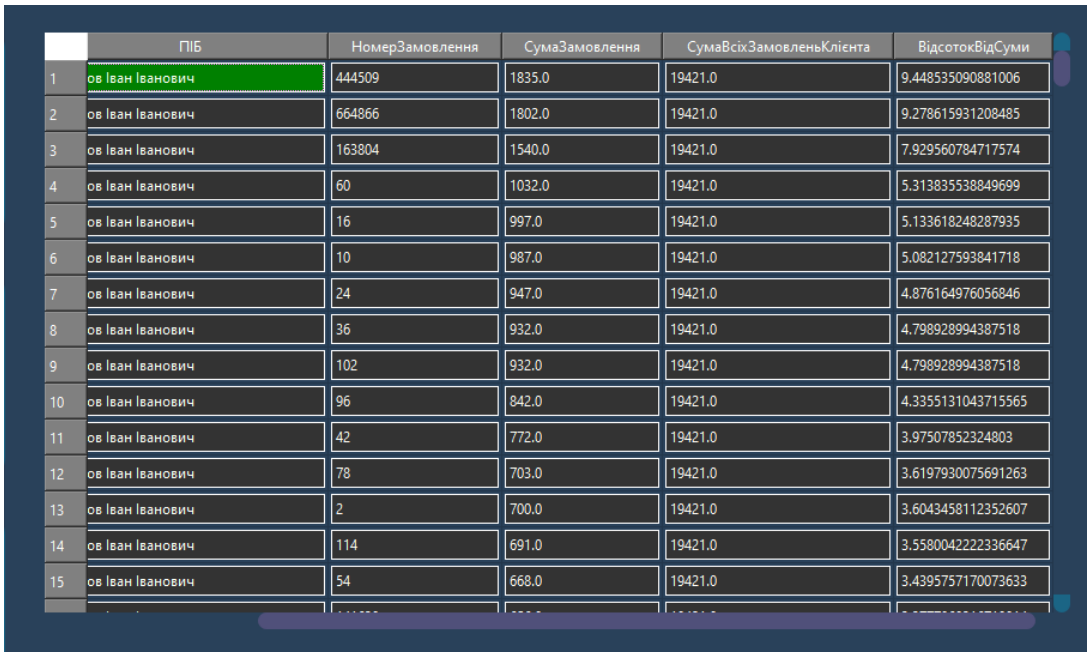
Інформація про вартість замовлень:

Відображає інформацію по кожному клієнту та його замовленнях

СумаЗамовлення – сума вказаного замовлення за номером

СумаВсіхЗамовленьКлієнта – сума всіх його замовлень:

Відсоток від суми = Відсоток СумиЗамовлення від СумиВсіхЗамовленьКлієнта



	ПІБ	НомерЗамовлення	СумаЗамовлення	СумаВсіхЗамовленьКлієнта	ВідсотокВідСуми
1	ов Іван Іванович	444509	1835.0	19421.0	9.448535090881006
2	ов Іван Іванович	664866	1802.0	19421.0	9.278615931208485
3	ов Іван Іванович	163804	1540.0	19421.0	7.929560784717574
4	ов Іван Іванович	60	1032.0	19421.0	5.313835538849699
5	ов Іван Іванович	16	997.0	19421.0	5.133618248287935
6	ов Іван Іванович	10	987.0	19421.0	5.082127593841718
7	ов Іван Іванович	24	947.0	19421.0	4.876164976056846
8	ов Іван Іванович	36	932.0	19421.0	4.798928994387518
9	ов Іван Іванович	102	932.0	19421.0	4.798928994387518
10	ов Іван Іванович	96	842.0	19421.0	4.3355131043715565
11	ов Іван Іванович	42	772.0	19421.0	3.97507852324803
12	ов Іван Іванович	78	703.0	19421.0	3.6197930075691263
13	ов Іван Іванович	2	700.0	19421.0	3.6043458112352607
14	ов Іван Іванович	114	691.0	19421.0	3.5580042222336647
15	ов Іван Іванович	54	668.0	19421.0	3.4395757170073633

Рис. 2.8.1.29. Результат виконання запиту

Рейтинг товарів:

Відображення рейтингу товару наступною формулою:

Відсоток товару в кожному замовлень + Відсоток клієнт які купили товар

	ПозиціяРейтингу	НазваТовару	Рейтинг
1	1	Бренді-кола	35.37543524956699
2	2	Оболонь Світле	34.907284222001486
3	3	hike-chill	34.7491420518319
4	4	Водка Лайм	34.527707153691956
5	5	Джин-тонік	34.00899366355522
6	6	BeerMix Лимон	33.87613272467125
7	7	Ciber Квіти бузини	33.20555254730818
8	8	Аквабаланс	33.06641612548097
9	9	Оболонська плюс лимон	32.68683905717144
10	10	BeerMix Вишня	32.66155782588583
11	11	Оболонська	32.218688029605936
12	12	Zibert Weissbier	32.193406798320325
13	13	Оболонь Безалкогольне	31.99725313146599
14	14	Оболонь Нефільтроване	31.718980287811576
15	15	Охтирське козацьке	31.567113600585234
16	16	Байкал	31.49754538967163

Рис. 2.8.1.30. Результат виконання запиту в формі відображення «Таблиця»

Пошук за номером замовлення:

Відображається інформація щодо замовлення

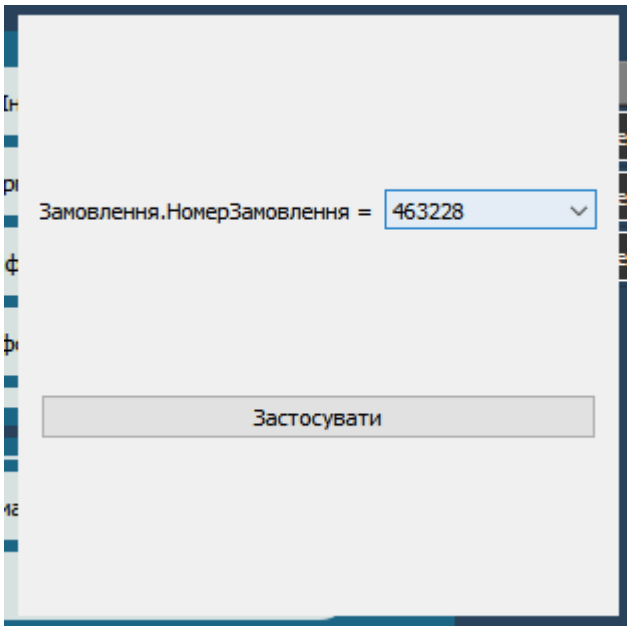


Рис. 2.8.1.31. Вибір параметру запиту

	ПІБ	НомерЗамовлення	НазваТовару	Кількість	Ціна	Сума
1	Хрущов Дмитро Назарович	463228	Бренді-кола	88	20.0	1760.0
2	Хрущов Дмитро Назарович	463228	Ciber Квіти бузини	73	10.0	730.0
3	Хрущов Дмитро Назарович	463228	Оболонська 2	42	15.0	630.0

Рис. 2.8.1.32. Результат роботи пошуку замовлень

Перехресна таблиця про кількість товару на складах:

Після натиснення кнопки пронується добавити стовбці з сумою та кількостю

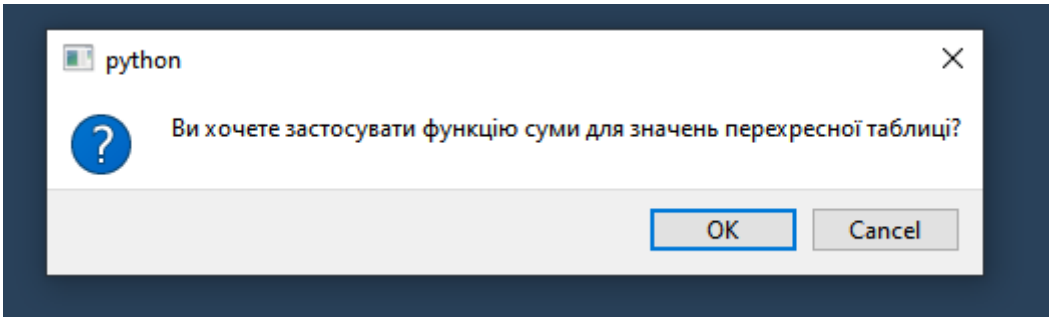


Рис. 2.8.1.33. Пропозиція системи про підрахунок суми в перехресній таблиці

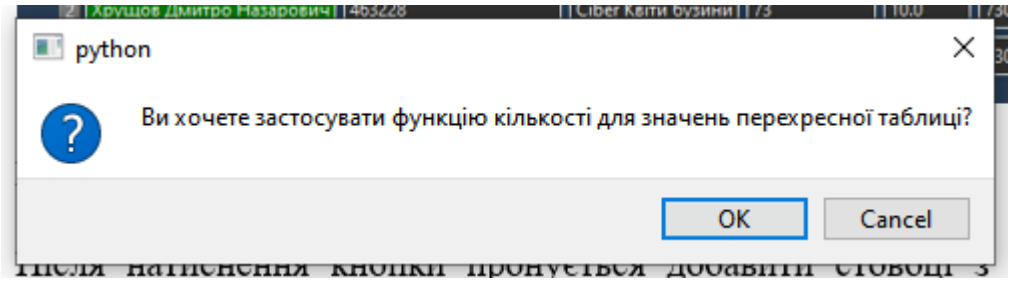
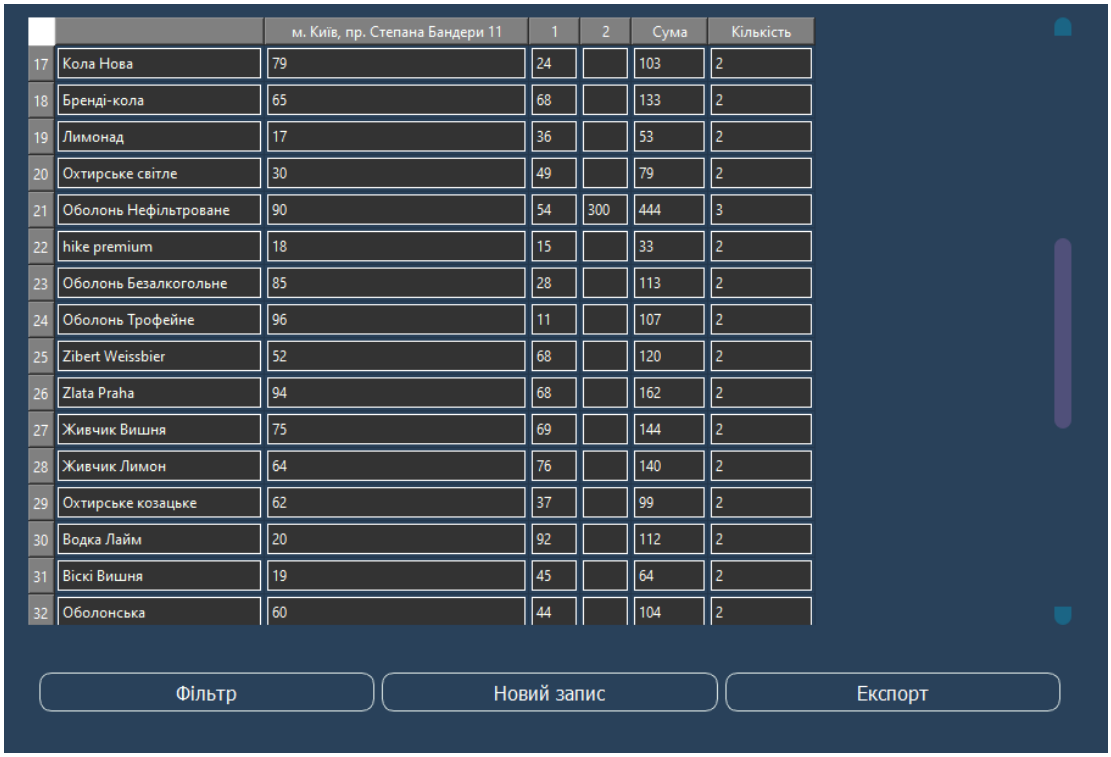


Рис. 2.8.1.34. Пропозиція системи про підрахунок кількості значень в перехресній таблиці



		м. Київ, пр. Степана Бандери 11	1	2	Сума	Кількість
17	Кола Нова	79	24		103	2
18	Бренді-кола	65	68		133	2
19	Лимонад	17	36		53	2
20	Охтирське світле	30	49		79	2
21	Оболонь Нефільтроване	90	54	300	444	3
22	hike premium	18	15		33	2
23	Оболонь Безалкогольне	85	28		113	2
24	Оболонь Трофейне	96	11		107	2
25	Zibert Weissbier	52	68		120	2
26	Zlata Praha	94	68		162	2
27	Живчик Вишня	75	69		144	2
28	Живчик Лимон	64	76		140	2
29	Охтирське козацьке	62	37		99	2
30	Водка Лайм	20	92		112	2
31	Віскі Вишня	19	45		64	2
32	Оболонська	60	44		104	2

Рис. 2.8.1.35. Результат виведення запиту

Де м.Київ, пр Степана Бандери 11, 1, 2 – Адреси Складів.

Також слід зазначити, що до кожного набору даних з запитів можливо застосувати фільтр:

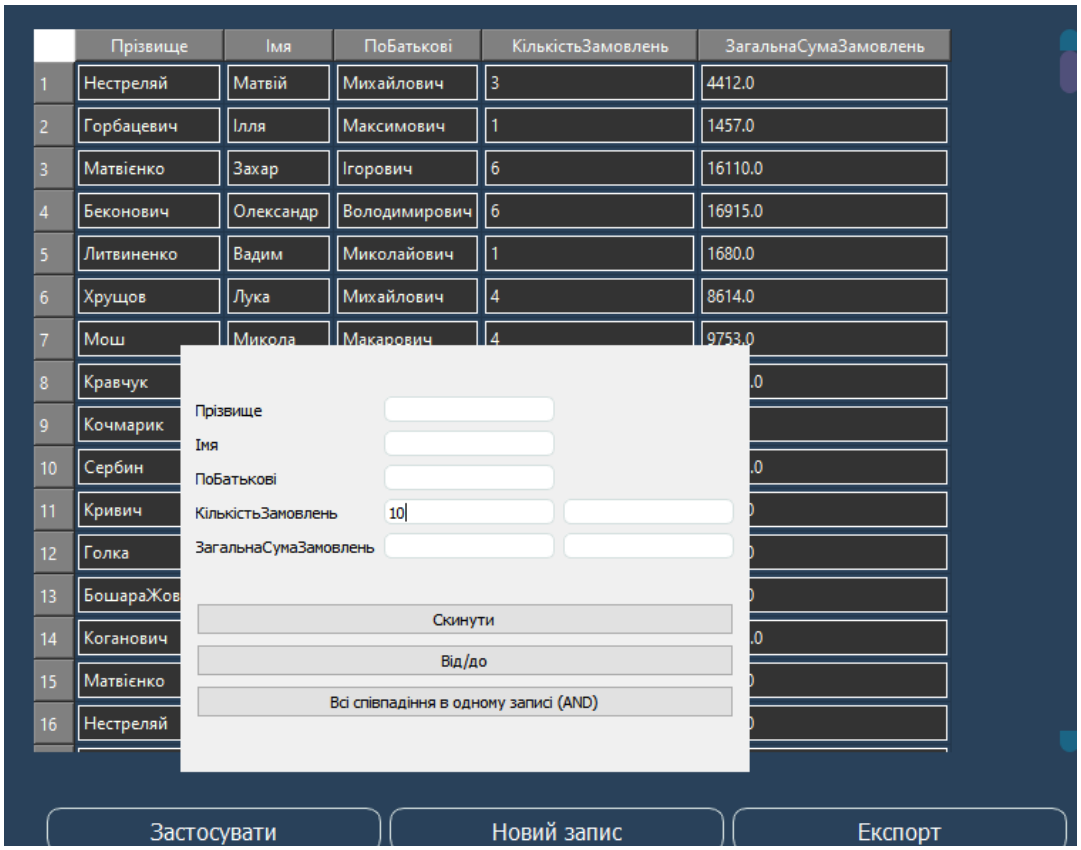


Рис. 2.8.1.36. Введення параметрів для фільтрації даних отриманих після запиту

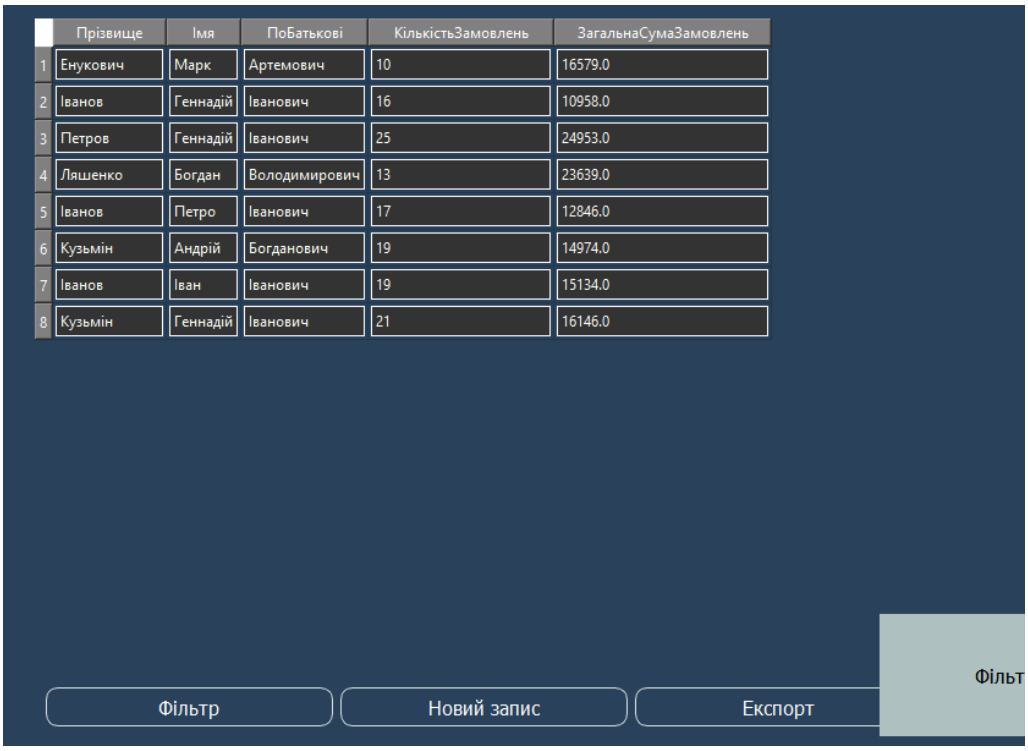


Рис. 2.8.1.37. Результат фільтрації даних

Для отримання звіту про оплату натискається відповідна кнопка в меню. Після натискання кнопки слід обрати шлях для збереження Excel файлу

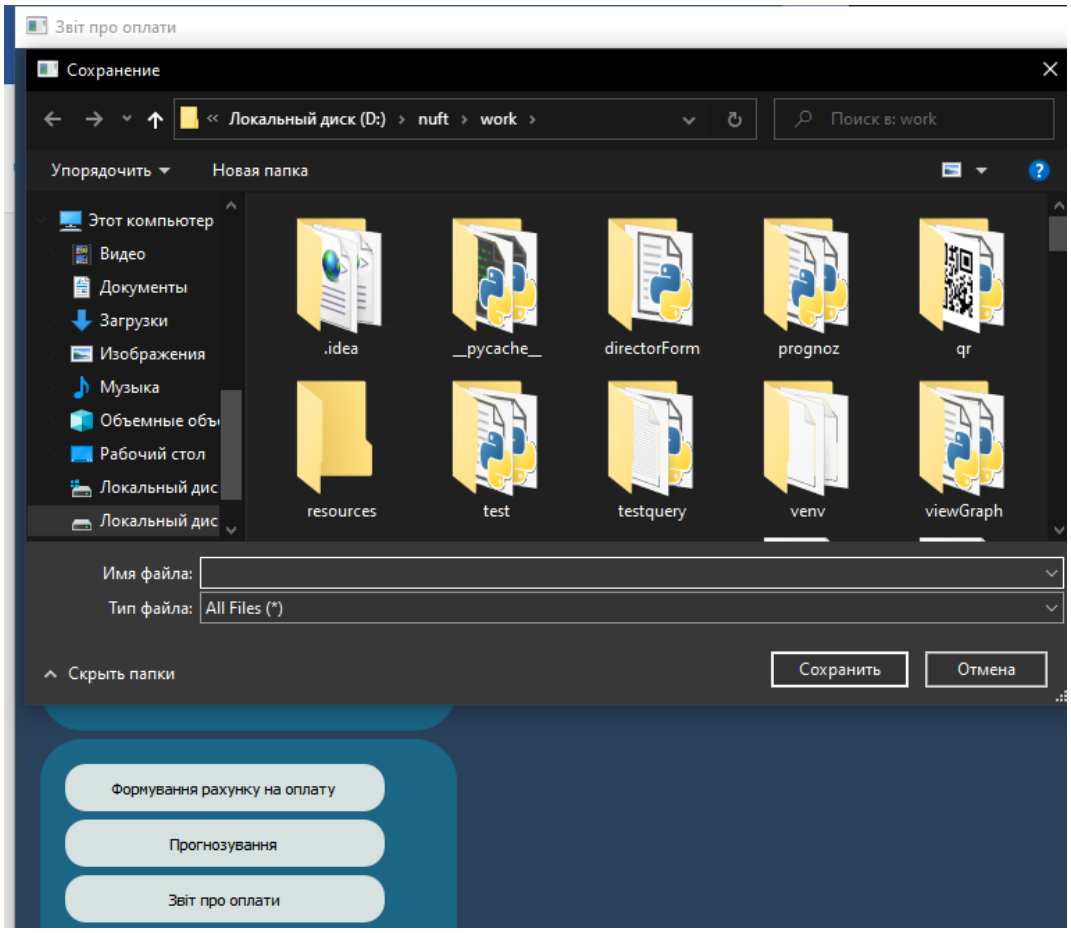


Рис. 2.8.1.38. Вибір шляху для імпорту звіту

	A	B	C	D	E	F
1	ПІБ	НомерЗамовлення	Дата	СумаОплати	ЗагальнаСума	
2	Іванов Іван Іванович	664866	2019-05-23	1802	19421	
3	Іванов Іван Іванович	163804	2019-07-09	1540	19421	
4	Іванов Іван Іванович	444509	2021-03-16	1835	19421	
5	Іванов Іван Іванович	564593	2021-03-31	510	19421	
6	Іванов Іван Іванович	141630	2021-05-13	656	19421	
7	Іванов Іван Іванович	10	2022-01-10	987	19421	
8	Іванов Іван Іванович	24	2022-01-10	947	19421	
9	Іванов Іван Іванович	84	2022-01-13	230	19421	
10	Іванов Іван Іванович	54	2022-01-21	668	19421	
11	Іванов Іван Іванович	114	2022-01-24	691	19421	
12	Іванов Іван Іванович	16	2022-02-16	997	19421	
13	Іванов Іван Іванович	30	2022-02-16	570	19421	
14	Іванов Іван Іванович	90	2022-02-19	392	19421	
15	Іванов Іван Іванович	60	2022-02-27	1032	19421	
16	Іванов Іван Іванович	66	2022-03-14	303	19421	
17	Іванов Іван Іванович	36	2022-03-22	932	19421	
18	Іванов Іван Іванович	96	2022-03-25	842	19421	
19	Іванов Іван Іванович	102	2022-04-12	932	19421	
20	Іванов Іван Іванович	72	2022-04-20	491	19421	
21	Іванов Іван Іванович	42	2022-04-28	772	19421	
22	Іванов Іван Іванович	2	2022-05-01	700	19421	
23	Іванов Іван Іванович	48	2022-05-15	418	19421	

Звіт про оплати

Рис. 2.8.1.39. Звіт в середовищі Excel

	A	B	C	D	E	F
2085	Недайхлеб Максим Васильйови	947239	2021-09-02	1131	8108	
2086	Недайхлеб Максим Васильйови	796225	2021-12-28	1835	8108	
2087	Беконович Даниїл Дмитрович	803394	2019-06-01	1843	14441	
2088	Беконович Даниїл Дмитрович	324682	2019-07-13	2020	14441	
2089	Беконович Даниїл Дмитрович	650983	2019-07-25	2227	14441	
2090	Беконович Даниїл Дмитрович	113287	2019-07-30	1331	14441	
2091	Беконович Даниїл Дмитрович	578783	2020-08-25	2396	14441	
2092	Беконович Даниїл Дмитрович	207947	2020-08-28	2480	14441	
2093	Беконович Даниїл Дмитрович	701074	2022-02-24	2144	14441	
2094	Гудзевич Ігор Олегович	242567	2019-03-29	1190	5297	
2095	Гудзевич Ігор Олегович	647679	2020-08-14	1600	5297	
2096	Гудзевич Ігор Олегович	221033	2022-05-16	2507	5297	
2097	Строганович Лука Назарович	110899	2020-03-09	4203	4203	
2098	Беконович Назар Олегович	187541	2019-08-04	2580	8025	
2099	Беконович Назар Олегович	937990	2020-12-21	1490	8025	
2100	Беконович Назар Олегович	196039	2021-08-23	788	8025	
2101	Беконович Назар Олегович	509936	2021-12-25	3167	8025	
2102	Мош Артур Вадимович	507920	2019-07-18	1920	3818	
2103	Мош Артур Вадимович	849788	2019-09-07	1898	3818	
2104	Турченяк Лука Дмитрович	883122	2020-02-22	1116	4779	
2105	Турченяк Лука Дмитрович	416882	2021-02-02	2384	4779	
2106	Турченяк Лука Дмитрович	831322	2021-07-02	1279	4779	
2107	Матвієнко Артем Дмитрович	860940	2019-08-16	2152	5139	

Рис. 2.8.1.40. Звіт в середовищі Excel

Для отримання звіту з прогнозування продажів потрібно натиснути кнопку «Прогнозування», після її натискання відкриється форма, де потрібно з випадаючого списку вибрати товар. Одразу після вибору товару в цьому ж вікну проектується часовий ряд, який містить дані щодо продаж, які враховані під час аналізі, дані тренду, а також прогнозовані дані. Кожний вид даних має свій колір. Для експорту даних в Excel та зберігання графіка картинкою потрібно натиснути кнопку «Зберегти картинкою» - файли збережуться по вказаному вами шляху.

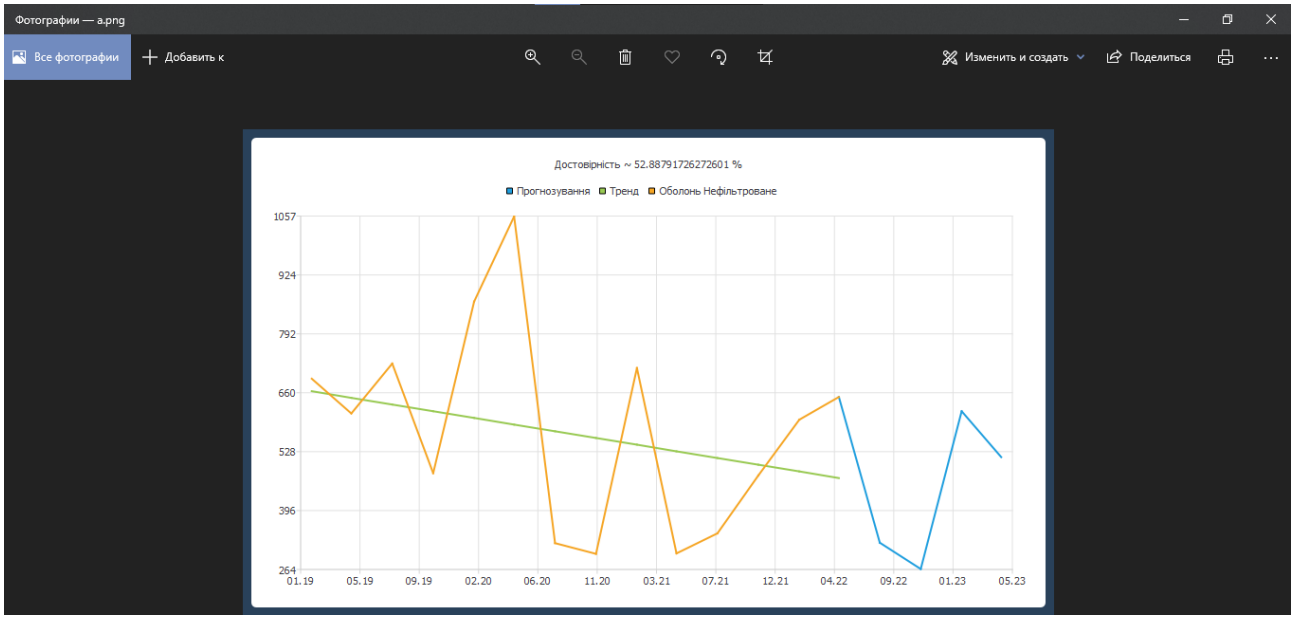


Рис. 2.8.1.41. Збережена картинка графіку «Прогнозування»

	A	B	C	D
1	name	date	value	
2	Прогнозування	2 квартар 2022	651	
3	Прогнозування	3 квартар 2022	324	
4	Прогнозування	4 квартар 2022	265	
5	Прогнозування	1 квартар 2023	619	
6	Прогнозування	2 квартар 2023	516	
7	Тренд	1 квартар 2019	664	
8	Тренд	2 квартар 2019	649	
9	Тренд	3 квартар 2019	634	
10	Тренд	4 квартар 2019	619	
11	Тренд	1 квартар 2020	604	
12	Тренд	2 квартар 2020	589	
13	Тренд	3 квартар 2020	574	
14	Тренд	4 квартар 2020	559	
15	Тренд	1 квартар 2021	544	
16	Тренд	2 квартар 2021	529	
17	Тренд	3 квартар 2021	514	
18	Тренд	4 квартар 2021	499	
19	Тренд	1 квартар 2022	484	
20	Тренд	2 квартар 2022	469	
21	Оболонь Нефільтроване	1 квартар 2019	692	
22	Оболонь Нефільтроване	2 квартар 2019	614	
23	Оболонь Нефільтроване	3 квартар 2019	726	

Data

Рис. 2.8.1.43. Звіт «Прогнозування продаж»

Щоб отримати розрахунки про рейтингу продаж потрібно натиснути кнопку «Звіт про тенденцію продаж» в меню користувача.

Після натиснення сформується таблиця, яка містить дані щодо назви товару та значення рейтингу, розраховане за формулою: Від кінцевого значення тренду відняти початкове значення тренду.

2.8.2. Кабінет директора

Після авторизації в обліковому запису до якого прив’язаний кабінет директора відкривається вікно, яке можна поділити на три пункти.

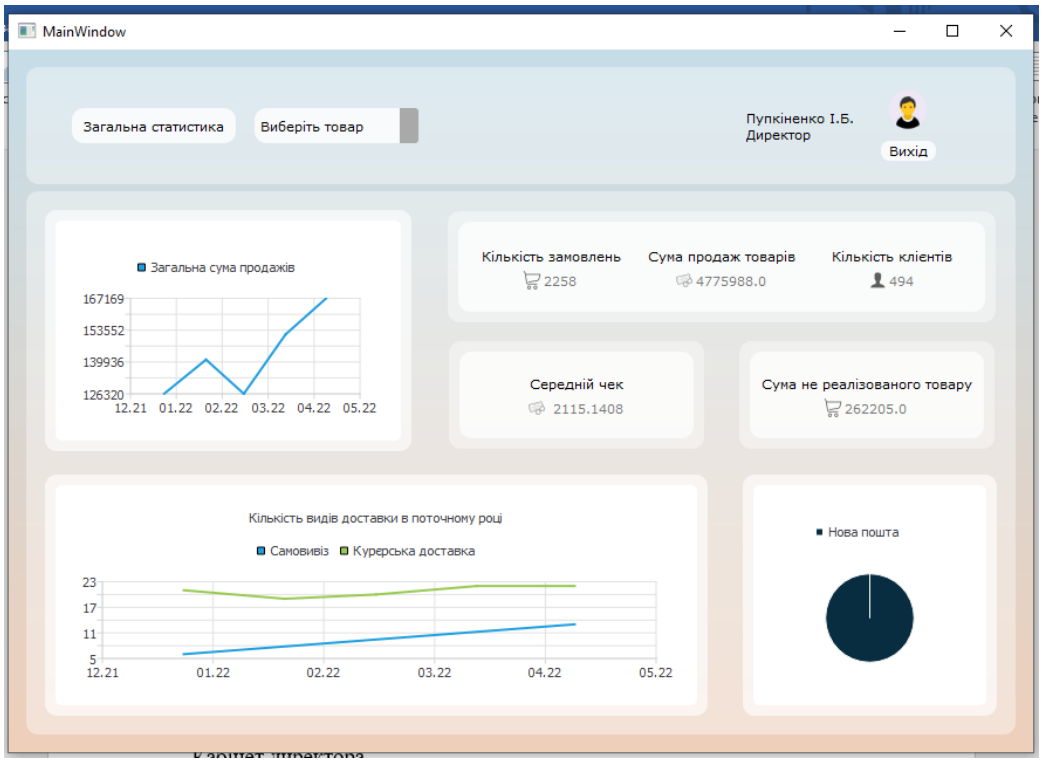


Рис. 2.8.2.1. Загальний вигляд форми

1. Меню

   В верхньому фреймі є дві кнопки, перша відповідає за загальну статистику. Інша відкриває випадаючий список з усіма товарами, натискаючи по якомусь елементу відкривається статитистика по цьому товару.

2. Управління обліковим записом.

   В верхньому правому фреймі знаходиться надпис з ПІБ директора та кнопка для виходу із облікового запису.

3. Форми виведення інформації

   3.1. Загальна статистика:

   Загальна статистика містить статистику по таким пунктам:

   - Загальна сума продаж товару
   - Кількість клієнтів
   - Середній чек
   - Сума не реалізованого товару
   - Прибуток по місячно за поточний рік

- Дані щодо загальної кількості доставок
- Дані щодо компанії, які надають послуги з доставки замовлень

3.2. Статистика по кожному товару окремо:

Відображення статистичних даних по кожному товару окремо:

- Дані щодо кількості продажу щомісячно в поточному році
- Дані щодо змін цін на товар
- Дані про загальну кількість реалізованого товару
- Дані про суму продаж вибраного товару
- Дані про відсоток наявності товару у всіх замовлень
- Дані про відсоток клієнтів, які купили поточний товар
- Дані про кількість та суму нереалізованого товару

Загальний вигляд форм відображено в додатках для цієї роботи.

2.8.3. Кабінет адміністратора ІС

Після авторизації під правами адміністратора вас направить на відповідне вікно, яке містить меню та фрейм для відображення інформації.

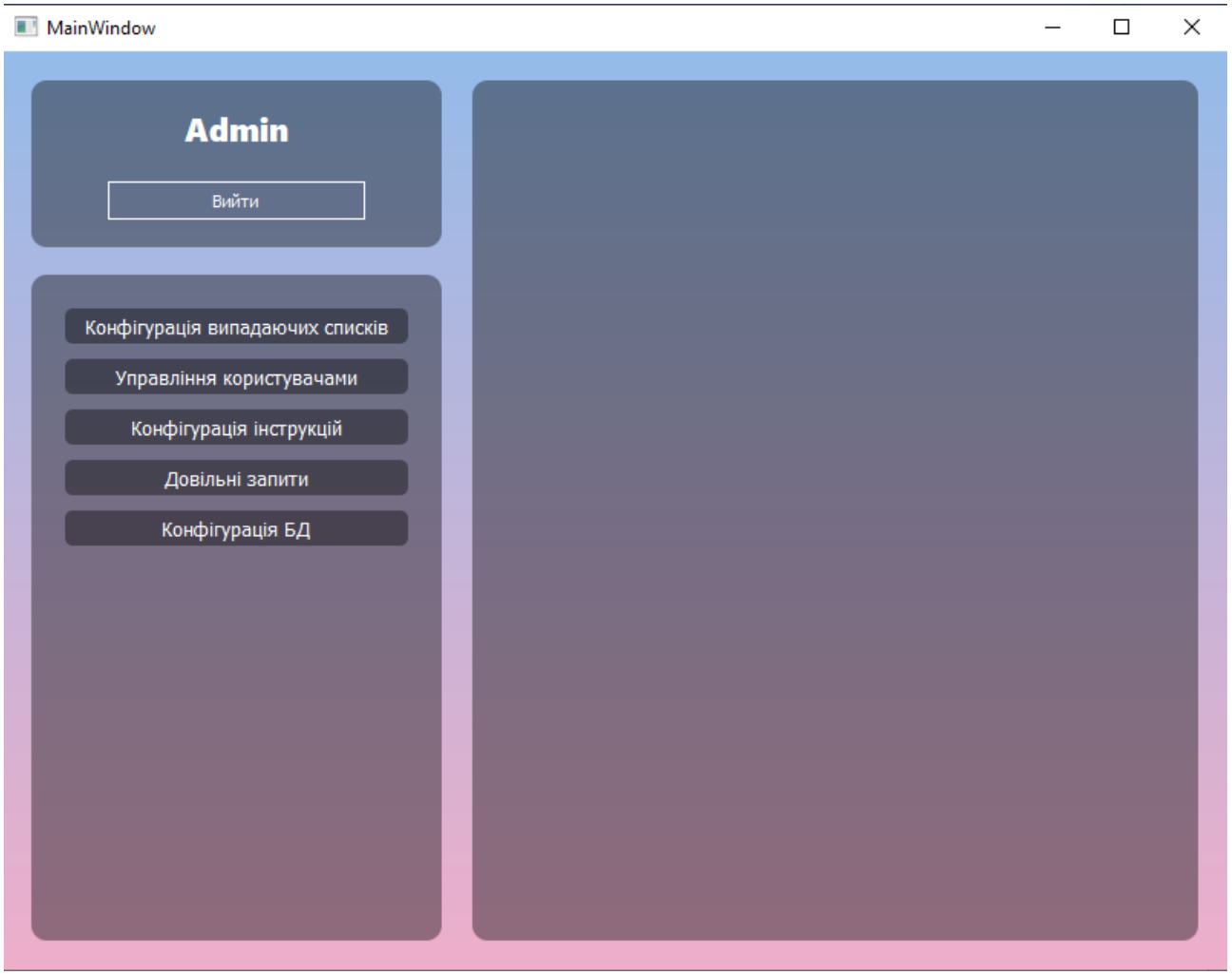


Рис. 2.8.3.1. Загальний вигляд форми

Меню знаходиться зліва та містить наступні блоки:

1. Конфігурація випадаючих списків

Рис. 2.8.3.2. Загальний вигляд конфігурації випадаючих списків

Тут відображається таблиця в якій можливо змінювати дані інструкцій, котрі відповідають за конфігурацію випадаючих списків. Конфігурація заключається в тому, що можливо вказувати порядковий номер стовбця, який буде відображатись в випадаючому списку замість ключа запису.

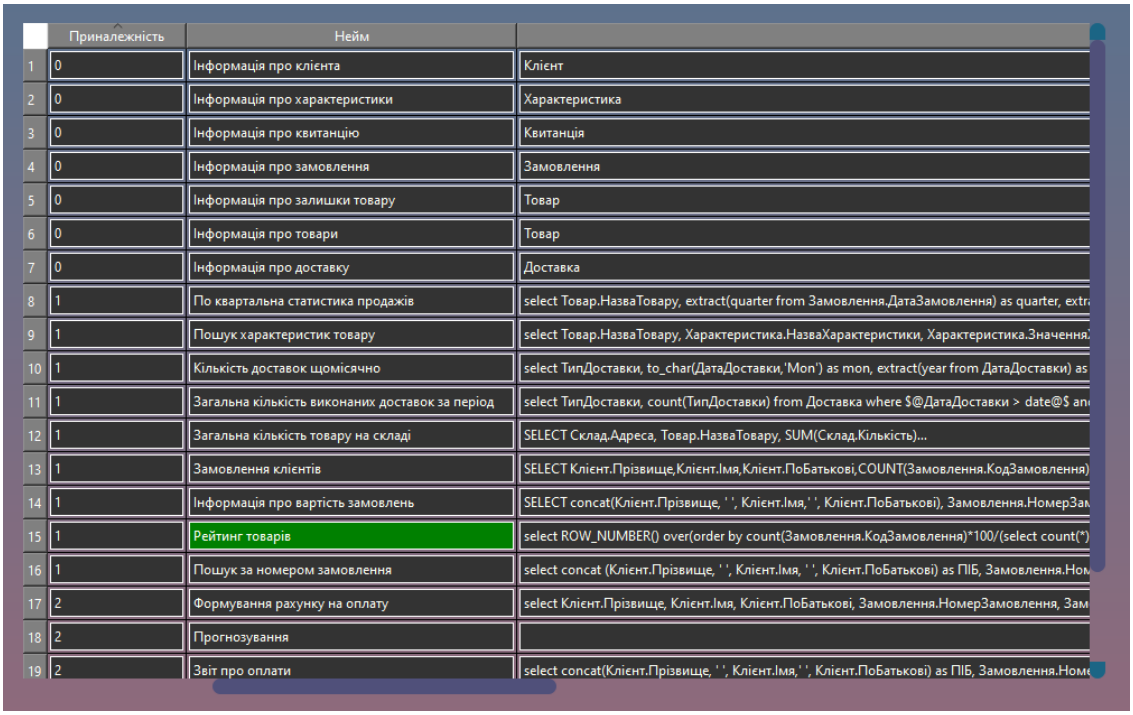
2. Управління користувача

Рис. 2.8.3.3. Загальний вигляд функції управління користувача

Тут відображається таблиця в якій можливо змінювати дані користувачів. Поле статус містить різні статуси:

- Enable – в обліковим записом все гарзд
- Disable – авторизація вимкнена під таким записом
- NeedNewPassword – це означає що користувач надіслав запит на зміну паролю

3. Конфігурація інструкцій



	Приналежність	Нейм	
1	0	Інформація про клієнта	Клієнт
2	0	Інформація про характеристики	Характеристика
3	0	Інформація про квитанцію	Квитанція
4	0	Інформація про замовлення	Замовлення
5	0	Інформація про залишки товару	Товар
6	0	Інформація про товари	Товар
7	0	Інформація про доставку	Доставка
8	1	По квартальна статистика продажів	select Товар.НазваТовару, extract(quarter from Замовлення.ДатаЗамовлення) as quarter, extr
9	1	Пошук характеристик товару	select Товар.НазваТовару, Характеристика.НазваХарактеристики, Характеристика.Значення
10	1	Кількість доставок щомісячно	select ТипДоставки, to_char(ДатаДоставки,'Mon') as mon, extract(year from ДатаДоставки) as
11	1	Загальна кількість виконаних доставок за період	select ТипДоставки, count(ТипДоставки) from Доставка where $@ДатаДоставки > date@$ an
12	1	Загальна кількість товару на складі	SELECT Склад.Адреса, Товар.НазваТовару, SUM(Склад.Кількість)...
13	1	Замовлення клієнтів	SELECT Клієнт.Прізвище,Клієнт.Імя,Клієнт.ПоБатькові,COUNT(Замовлення.КодЗамовлення)
14	1	Інформація про вартість замовлень	SELECT concat(Клієнт.Прізвище, ' ', Клієнт.Імя,' ', Клієнт.ПоБатькові), Замовлення.НомерЗам
15	1	Рейтинг товарів	select ROW_NUMBER() over(order by count(Замовлення.КодЗамовлення)*100/(select count(*)
16	1	Пошук за номером замовлення	select concat (Клієнт.Прізвище, ' ', Клієнт.Імя, ' ', Клієнт.ПоБатькові) as ПІБ, Замовлення.Ном
17	2	Формування рахунку на оплату	select Клієнт.Прізвище, Клієнт.Імя, Клієнт.ПоБатькові, Замовлення.НомерЗамовлення, Зам
18	2	Прогнозування	
19	2	Звіт про оплати	select concat(Клієнт.Прізвище, ' ', Клієнт.Імя,' ', Клієнт.ПоБатькові) as ПІБ, Замовлення.Номе

Рис. 2.8.3.4. Загальний вигляд форми для управління інструкціями

Тут вказані інструкції для забезпечення різноманітними функціями користувачів.

Стовбець «приналежність» відповідає за порядковий номер блоку в меню.

Стовбець «нейм» відповідає за назву кнопки в меню.

Якщо приналежність = 0:

- Стовбець «Функція» містить інформацію, щодо назви таблиці, дані яких будуть відображатись

- Стовбець «ДодатковаІнформація» містить інформацію, щодо назви дочірньої таблиці, яку потрібно виводити разом із основною таблицею.
- Якщо таку таблицю не потрібно виводити, то залишаємо поле пустим.
- Стовбець «Параметри» залишається порожнім.

Якщо приналежність дорівнює 1 або 2:

- Стовбець «Функція» містить інформацію, щодо запиту на мові SQL.
- Стовбець «ДодатковаІнформація» містить інформацію, щодо назв заголовків, які потрібно відображати разом з даними.
- Стовбець «Параметри» містить інформацію щодо типу даних в якому потрібно відображати дані.

4. Довільні запити

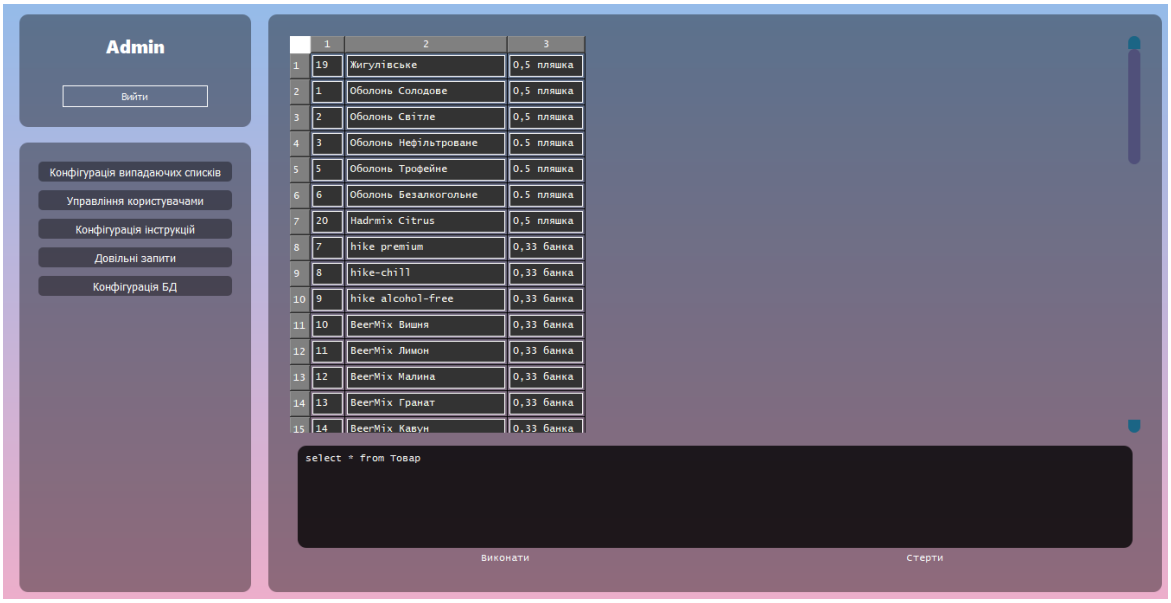


Рис. 2.8.3.5. Загальний вигляд форми «Довільні запити»

Містить текстове поле для введення запиту на мові SQL, та дві кнопки «Виконати» та «Стерти»

Якщо потрібно виконати введений запит – натискаєте кнопку «Виконати»

Якщо потрібно стерти запит в текстовому полі – натискаєте кнопку «Стерти»

Якщо це запит на видалення, оновлення, вставку – таблиця отриманих даних не відображається

Якщо це запит на вибірку та в БД присутні дані, які отримуються після виконання запиту, то відображається таблиця, але обмежена можливість їх редагування безпосередньо в таблиці.

2.9. Технічне та системне забезпечення розробки

2.9.1. Обґрунтування вибору технічних засобів

Для розробки програмного забезпечення було використано багато різних програмних засобів.

Для створення моделі бази даних Erwin Data Modeler.

AllFusion ERWin Data Modeler (ERWin) – CASE-засіб проектування баз даних від фірми Computer Associates. ERWin поєднує графічний інтерфейс Windows, інструменти для побудови ER-діаграм, редактори для створення логічного та фізичного опису моделі даних і прозору підтримку провідних реляційних СУБД. ERWin не прив’язаний до технології будь-якої конкретної фірми, що поставляє СУБД або засобу розробки. Він підтримує різні сервери баз даних та настільні СУБД, а також може звертатися до бази даних через інтерфейс ODBC (мова йде тільки про реляційні СУБД).

Систему управління базами даних обрано PostgreSQL.

PostgreSQL, також відомий як Postgres - це об'єктно-реляційна система управління базами даних, і вона використовує SQL (мова структурованих запитів) як основну мову запитів. Ми можемо інтегрувати PostgreSQL з будь-якою мовою програмування, таким як Java, C, C++ і т. д. Postgres має багато функцій, які ми не знайти в інших базах даних. Він підтримує складні структури та широкий спектр вбудованих та визначених користувачем типів даних. Також

забезпечує розширену ємність даних і заслужив довіру дбайливим ставленням до цілісності даних.

Python - це інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією.[3] Python розроблений під ліцензією з відкритим вихідним кодом. Тобто його можна використовувати безкоштовно навіть у комерційних проектах. Один з головних плюсів цієї мови програмування – це об'єм коду для реалізації того або іншого завдання. Об’єм значно менше, порівняно з іншими мовами. Також плюсом цієї мови є велика кількість уже створених бібліотек, існують бібліотеки для управління базами даних, тестуванням системи, та навіть бібліотека для створення QR кодів, яку було застосовано в цій роботі. І ще безліч різноманітних бібліотек.

PyQT5 – оболонка на мові програмування Python для бібліотеки Qt. Бібліотека реалізована в Python-модулях, та охоплює близько 1000 класів.[4] Головним плюсом є наявність середовища QT Designer для створення UI – дизайну.

PyCharm – середовище для розробки додатків мовою програмування Python, яке має простий інтерфейс та якісний рефакторінг та багато інших функцій потрібних для розробки різноманітних систем.

2.9.2. Визначення топології комп’ютерної мережі

Під час розробки системи використовувався локальний сервер, але оскільки це інтернет магазин, то сервер разом з базою даних повинні бути доступними кожному користувачу, тому що інформація про замовлення клієнтів надходить безпосередньо з магазину. Для цього було забезпечено можливість міграції сервера без втрати розробленого додатку, оскільки з’єднання з БД встановляється згідно параметрів, які прописані в окремому файлі для зручності їх зміни.

2.9.3. Обґрунтування вибору ОС

Python – кросплатформна мова програмування. А PyQT5 - це мультиплатформна бібліотека, яка працює на всіх основних операційних системах. Тестування системи відбувалось під операційною системою Windows, але на Linux, Unix та Mac OS все повинно працювати, але не без нюансів, які можуть виникнути під час інсталювання. Головний пріоритет при розробці випав на операційну систему Windows, а саме Windows 10. Тому що згідно статистики від компанії «Net Applications» саме Windows 10 є лідером серед операційних систем встановлених на персональних комп’ютерах по всьому світу.

2.9.4. Заходи захисту від несанкціонованого доступу до системи

Для забезпечення захисту від несанкціонованого доступу до ІС розроблена авторизація користувачів системи. Отже, щоб маніпулювати даними в базі потрібно мати обліковий запис зареєстрований в системі. Облікові записи надаються безпосередньо адміністраторами системи.

ВИСНОВКИ

Даний проект розроблявся для інформаційної підтримки діяльності інтернет-крамниці. Для аналізу предметної області використовувалось середовище Erwin Process Modeler. Для проектування БД використовувався Erwin Data Modeler. Проект виконано у середовищі PyCharm з використанням мови програмування Python, PyQT5, та СУБД PostgreSQL. Розроблена інформаційна система спрямована на підвищення ефективності контролю за продажами продукції інтернет-крамниці та відображення агрегованих та статистичних даних для формування звітності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. М’якшило О.М. CASE-технології у проектуванні інформаційних систем: [електронний ресурс] навч. посібник для студентів вищих навчальних закладів / О.М. М’якшило, Л.Г. Загоровська,– К.: НУХТ, 2017. – 190 с.

2. Проектування інформаційних систем: лабораторний практикум для студ. освіт. ступ. "Бакалавр" спец. 122 "Комп'ютерні науки " ден. і заоч. форм навч. Частина 2 "Проектування клієнтського додатку" / уклад. : О. М. М'якшило, О. В. Харкянен; Нац. ун-т харч. технол. - Київ : НУХТ, 2017. - 33 с.

3. Методичні рекомендації до виконання кваліфікаційної роботи на здобуття освітнього ступеня «Бакалавр» спеціальності 122 «Комп’ютерні науки» освітньо-професійної програми «Комп’ютерні науки» денної та заочної форм навчання / уклад. : Л. Г. Загоровська, О. М. М’якшило, М. П. Костіков. – К. : НУХТ, 2020. – 30 с.

4. Теорія прийняття рішень [Електронний ресурс]: лаборат. практикум для студентів освітнього ступеня «бакалавр» спеціальності 122 «Комп’ютерні науки та інформаційні технології» денної та заочної форм навчання / уклад. Л.Г. Загоровська, С.В. Грибков, Т.В. Ярова – К.: НУХТ, 2016. – 43с.

5. Оболонь. Статья з новостного блогу. Доступ: https://obolon.kyivcity.gov.ua/news/15556.html

6. Рейтинг 25 найбільших алкогольних компаній України. Доступ: https://landlord.ua/reytingi/reiting-25-krupneyshin-alkogolnih-kompaniy-ukraini/

7. Django. Доступ: https://uk.wikipedia.org/wiki/Django

8. Оболонь. Вікіпедія. Доступ: https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BE%D0%BB%D0%BE%D0%BD%D1%8C_(%D0%BA%D0%BE%D0%BC%D0%BF%D0%B0%D0%BD%D1%96%D1%8F)

9. Оболонь. Офіційний сайт. Доступ: https://obolon.ua/

10. Положення про відділ маркетингу. Доступ: http://trudova-ohrana.ru/primery-dokumentov/zrazki-polozhennja-pro-vddli/3935-polozhennja-pro-vddl-marketingu.html

11. PostgreSQL. Доступ: https://uk.education-wiki.com/5154595-what-is-postgresql

12. Моделювання бізнес процесів. Доступ: https://kd43.ru/uk/magazines/modelirovanie-biznes-processov-opisanie-i-analiz-primery-analiz.html

ДОДАТКИ

ДОДАТОК А - ФУНКЦІОНАЛЬНІ МОДЕЛІ

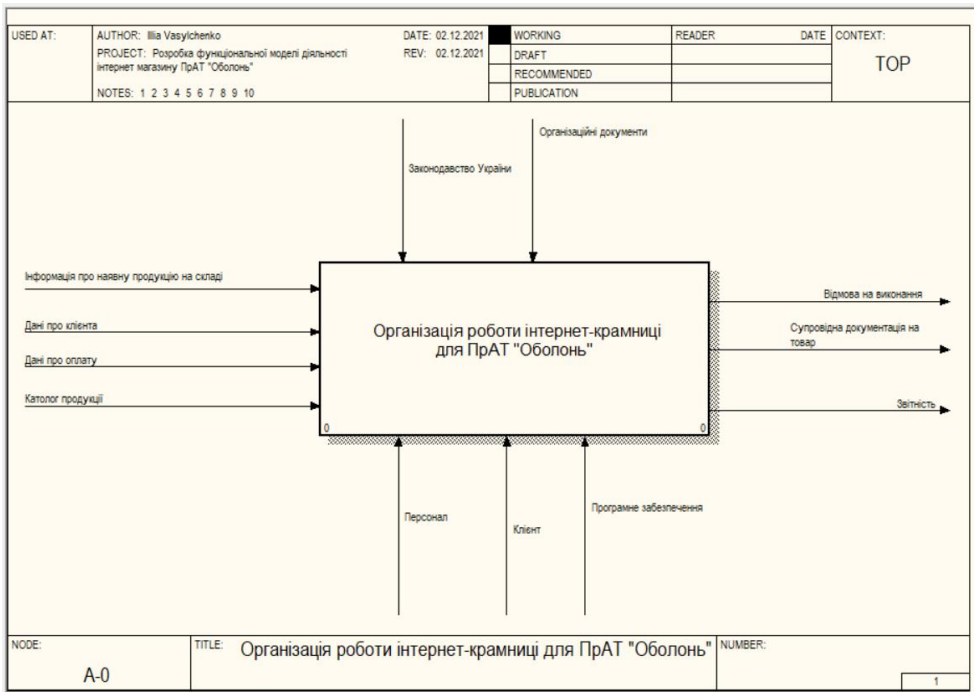


Рисунок А.1 - Контекстна діаграма AS-IS для ПрАТ «Оболонь»

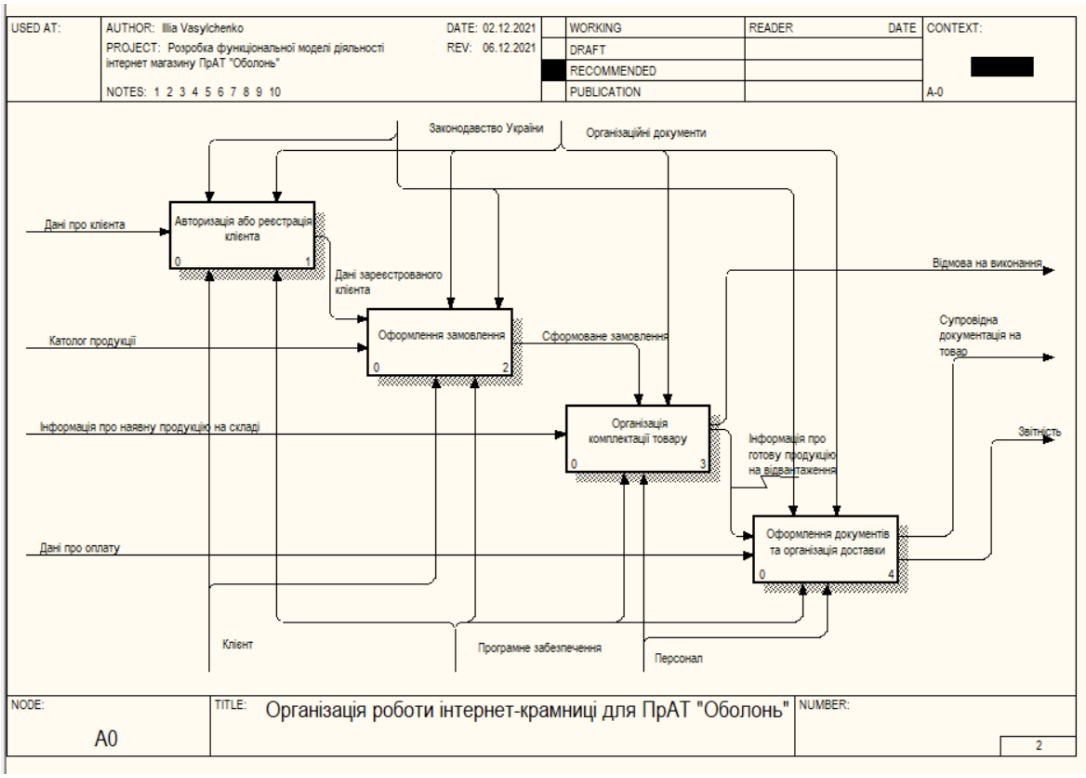


Рисунок А.2 - Декомпозиція першого рівня

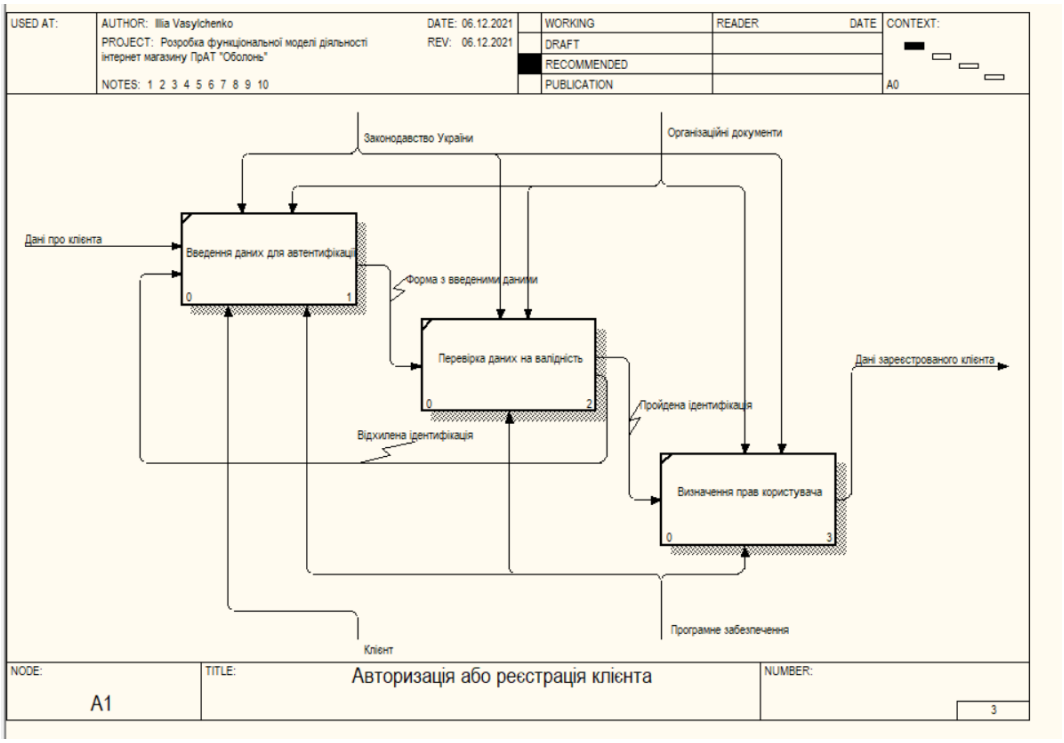


Рисунок А.3 - Декомпозиція другого рівня «Авторизація або реєстрація клієнта»

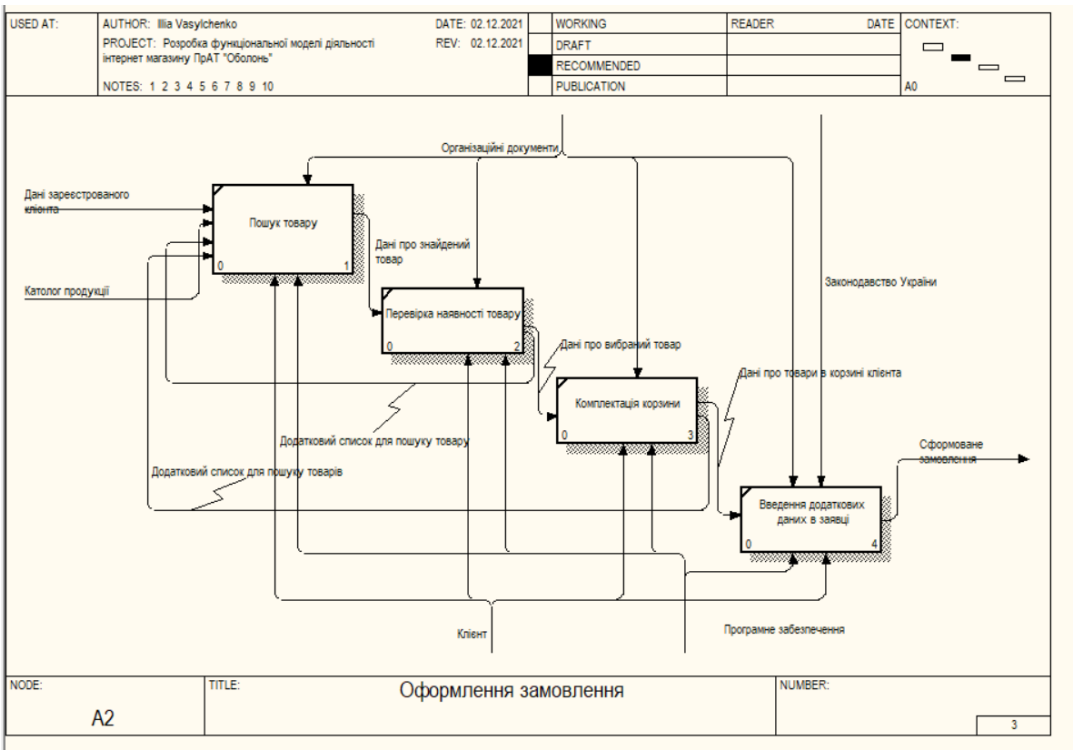


Рисунок А.4 - Декомпозиція другого рівня «Оформлення замовлення»

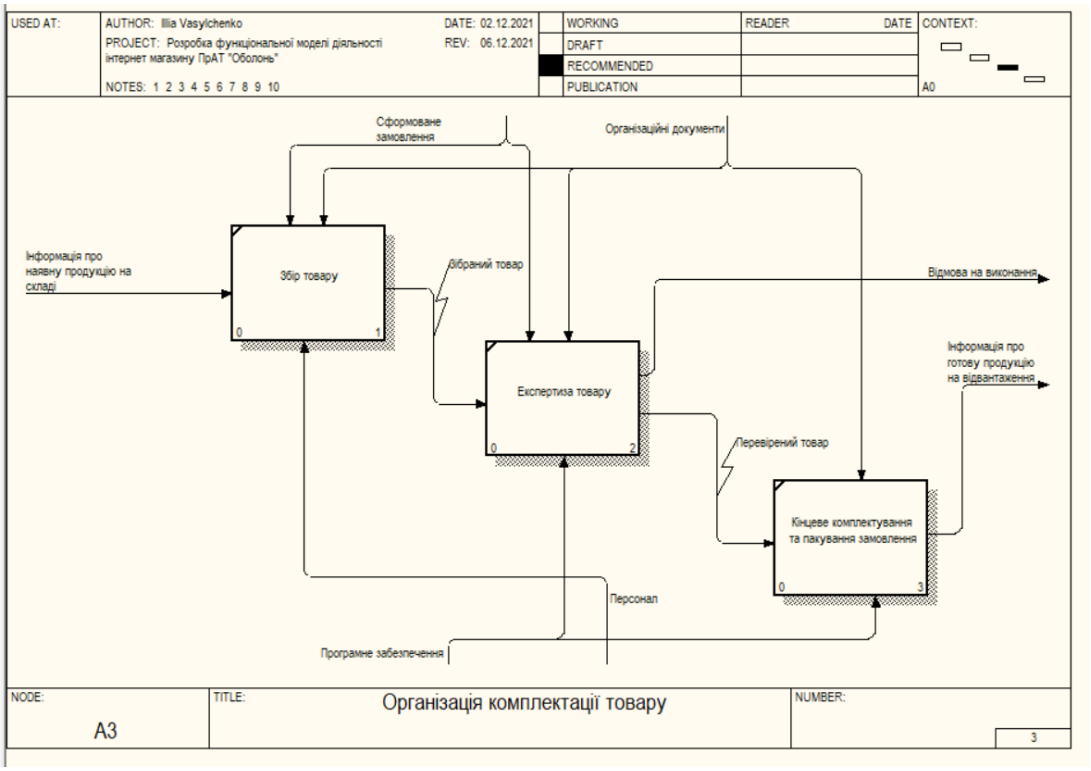


Рисунок А.5 - Декомпозиція другого рівня «Організація комплектації товару»

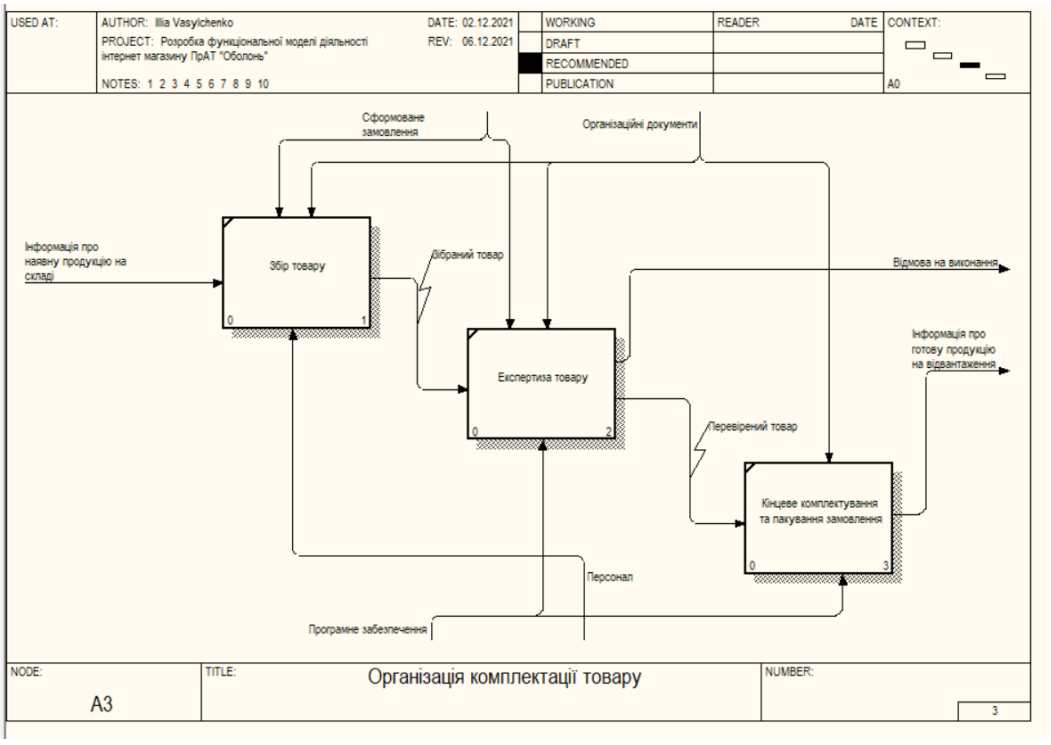


Рисунок А.6 - Декомпозиція другого рівня «Організація комплектації товару»

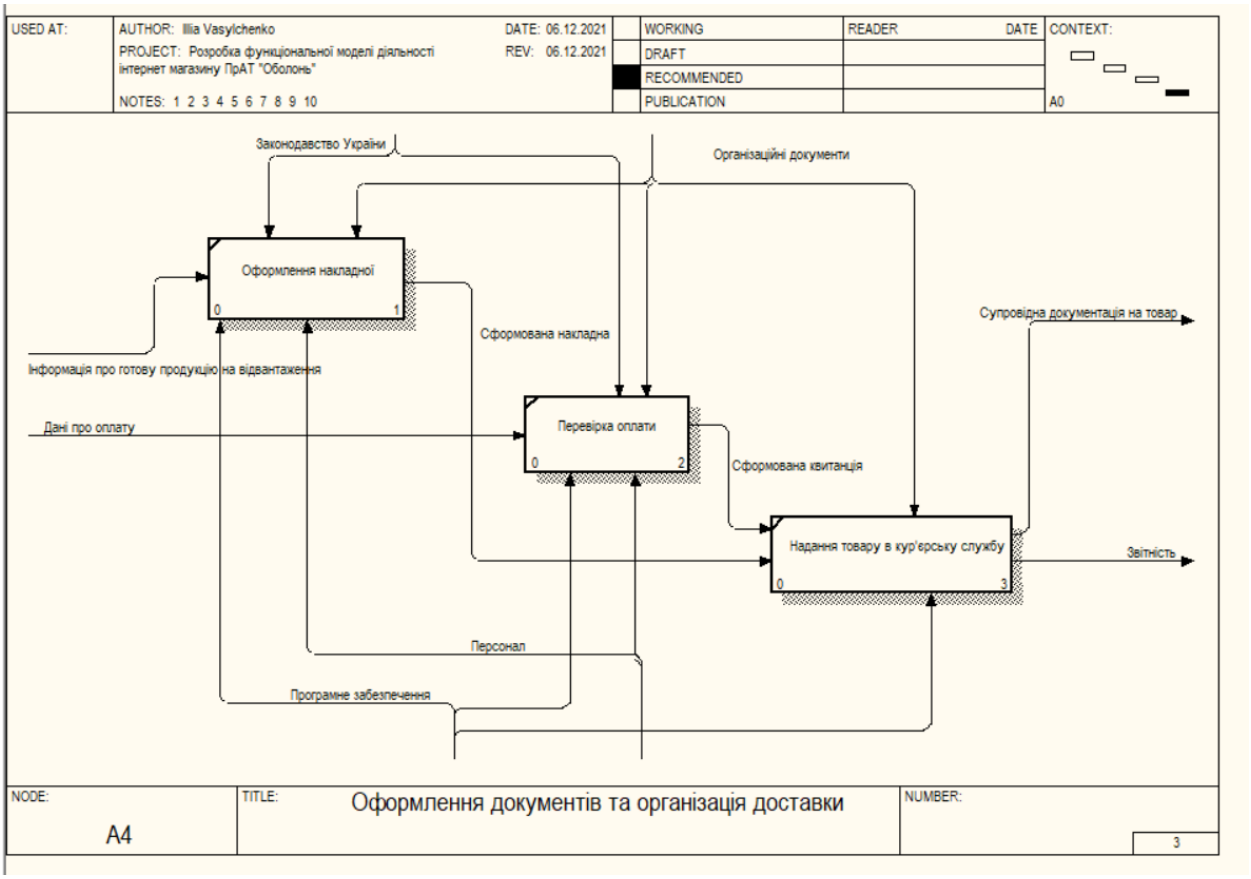


Рисунок А.7 - Декомпозиція другого рівня «Оформлення документів та організація доставки»

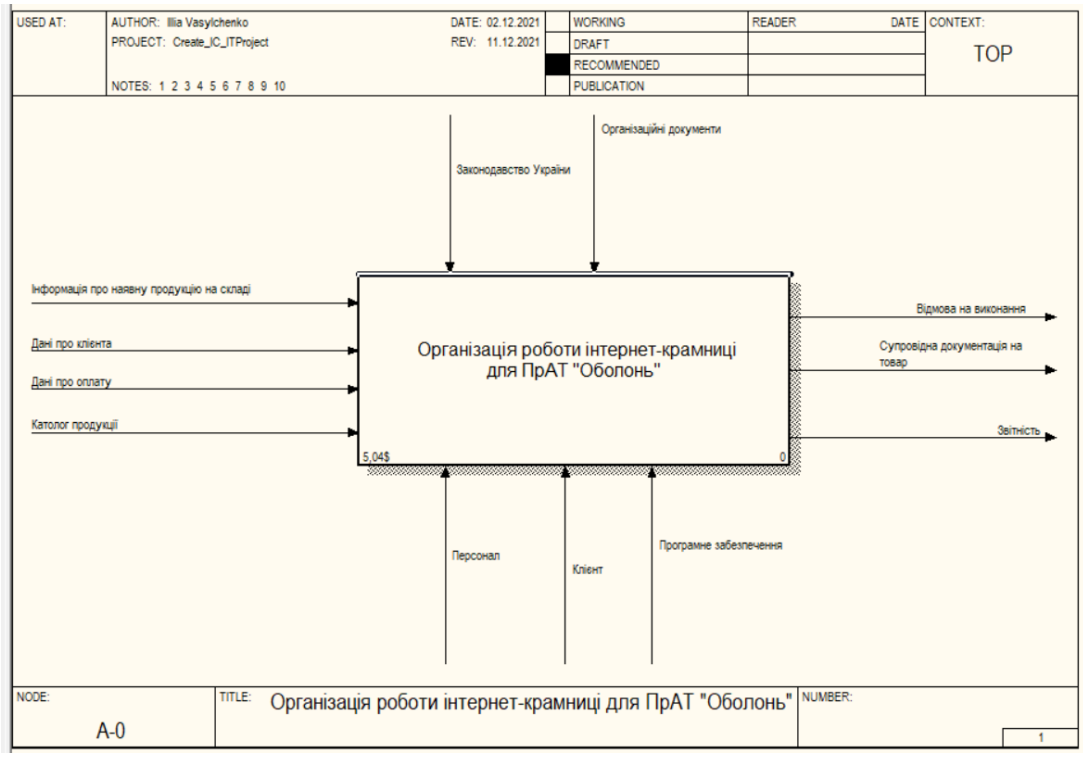


Рисунок А.8 - Контекстна діаграма в нотації IDEF0 (TO BE)

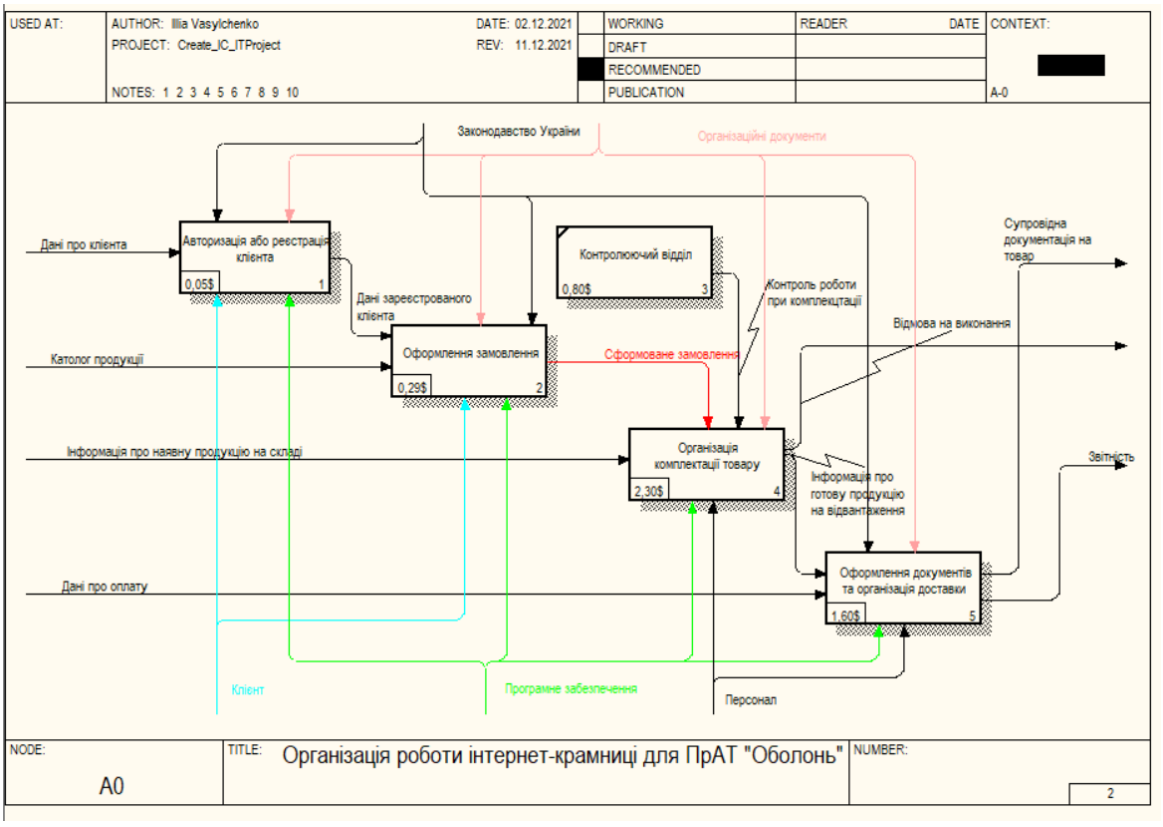


Рисунок А.9 - Декомпозиція першого рівня в нотації IDEF0 (TO BE)

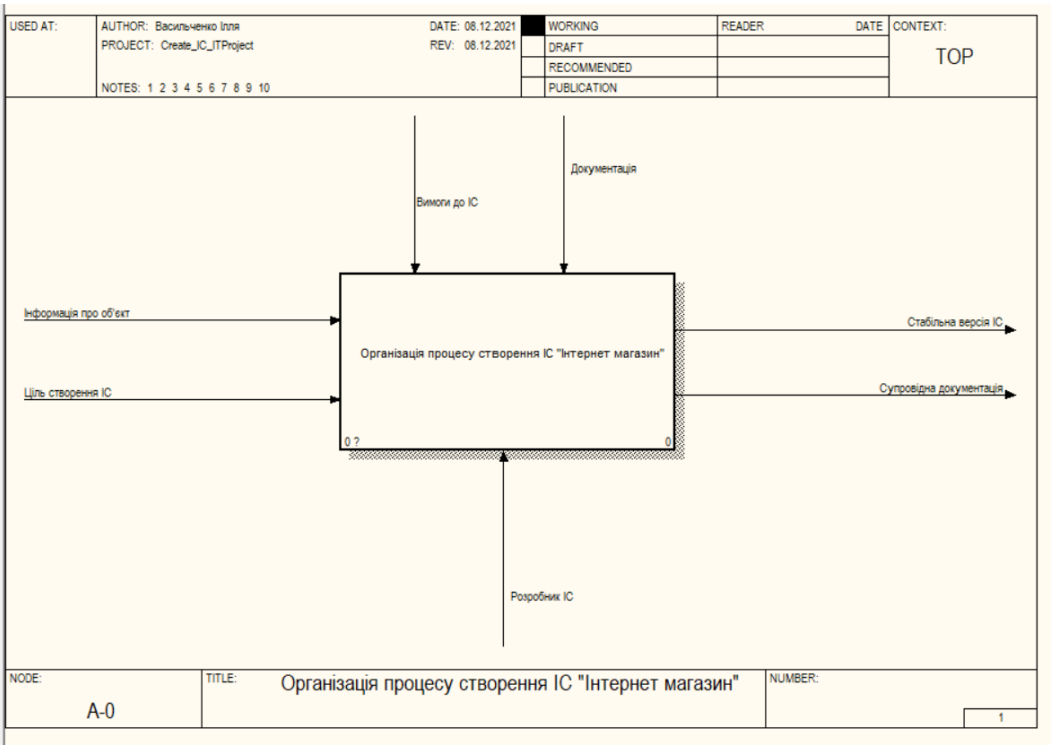


Рисунок А.10 – Контекстна діаграма «Організація процесу створення ІС для інтернет магазину»

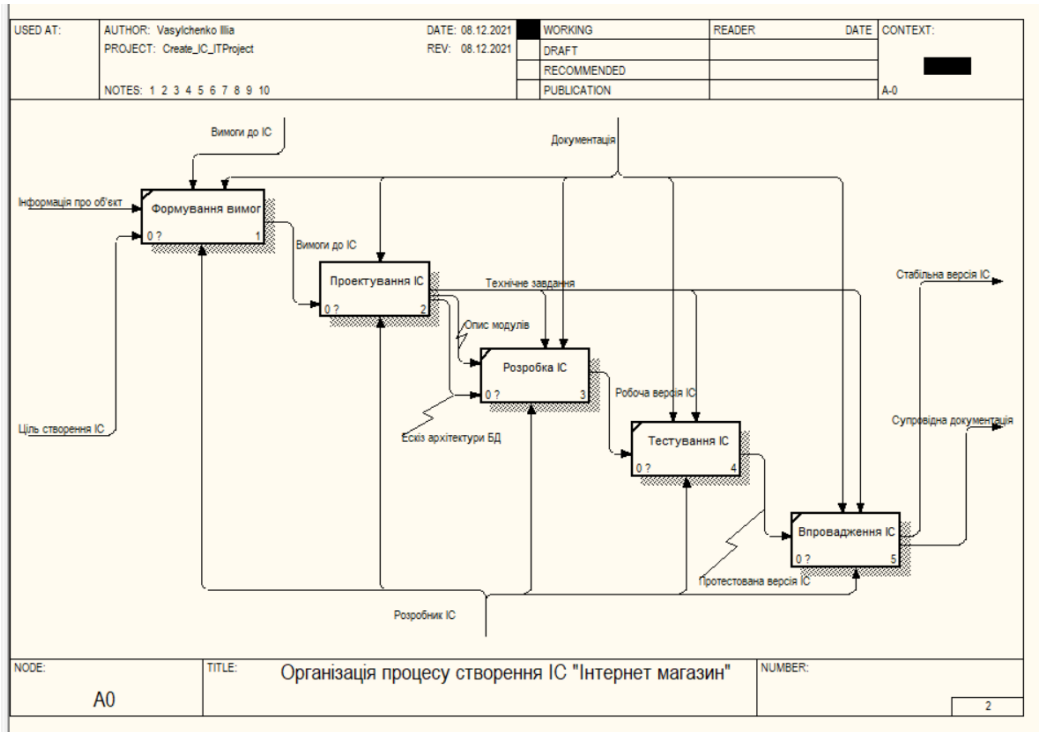


Рисунок А.11 – Декомпозиція першого рівня «Організація процесу створення ІС для інтернет магазину»

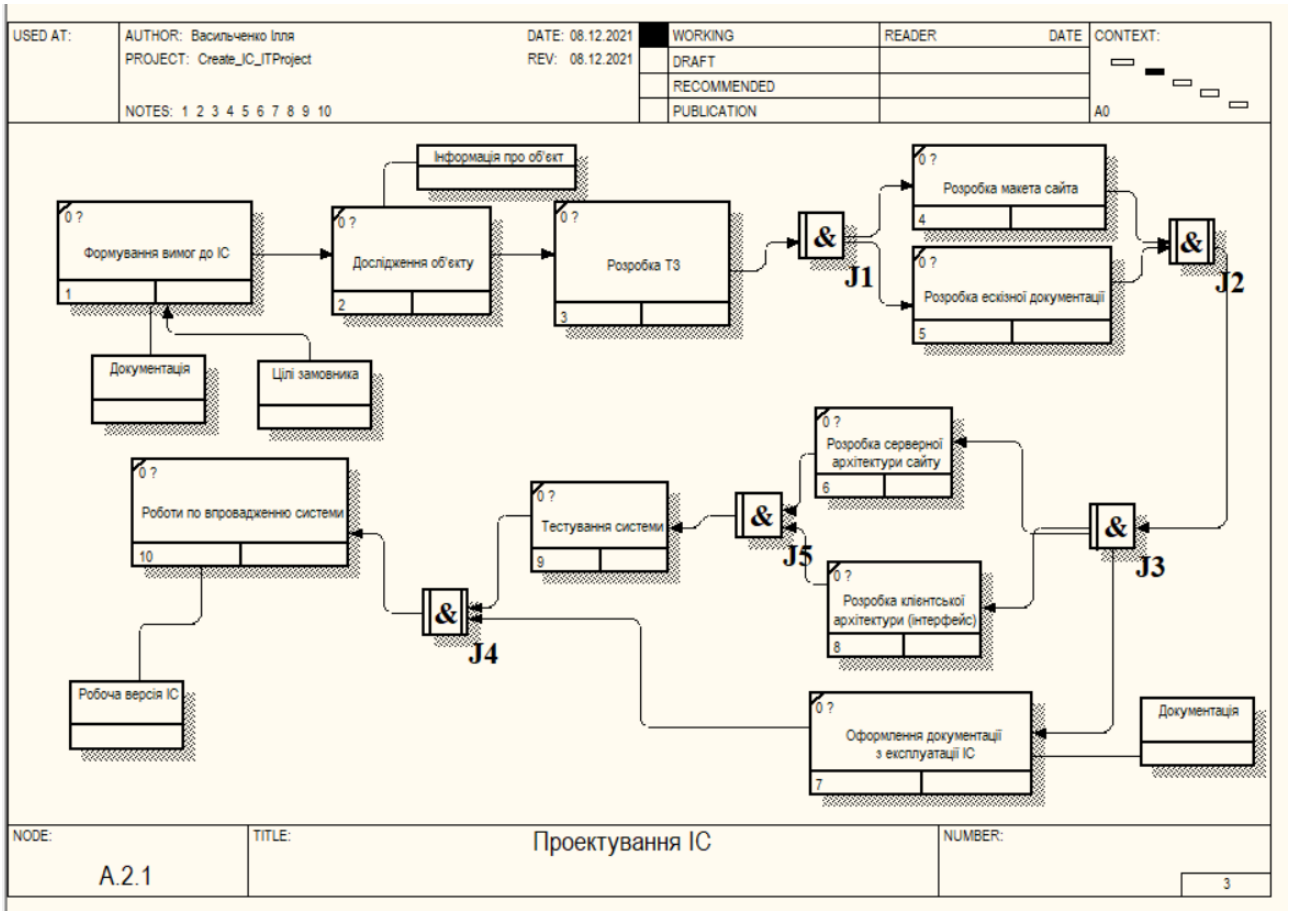


Рисунок А.13 - Модель процесу проектування ІС в нотації IDEF3

ДОДАТОК Б – МОДЕЛІ БАЗИ ДАНИХ

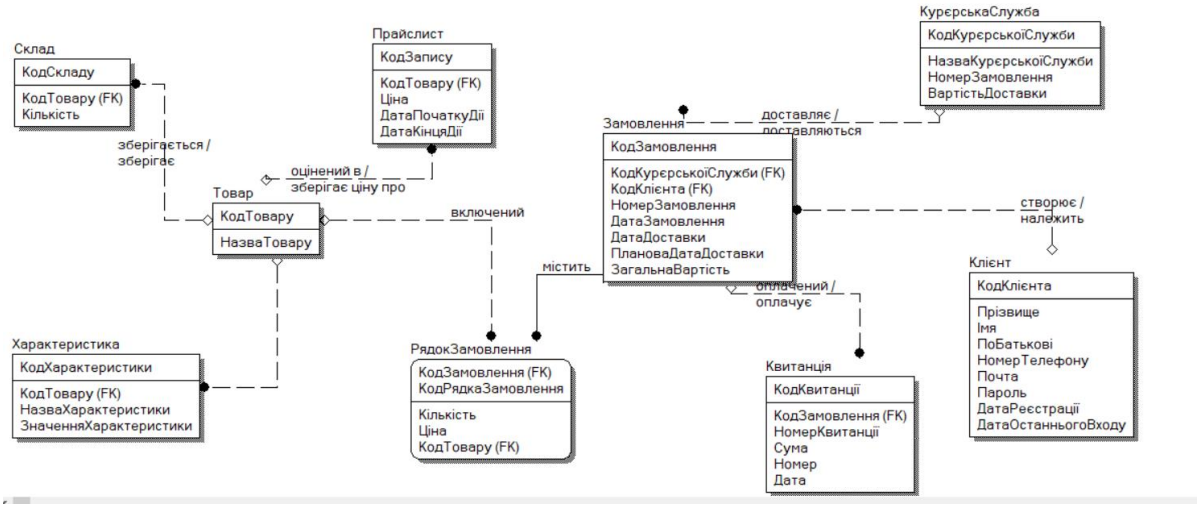


Рис. Б1. Логічна модель бази даних

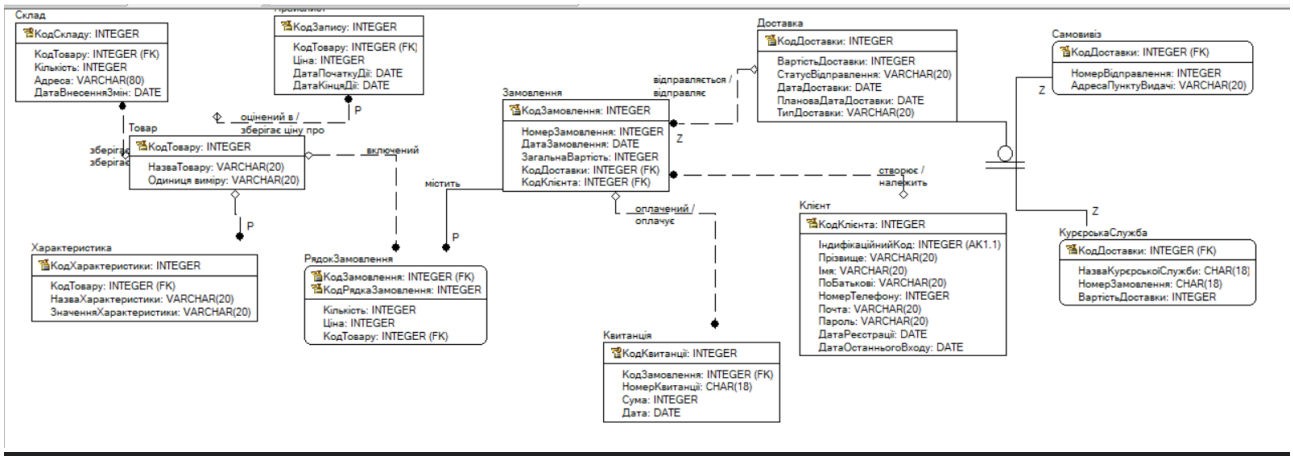


Рис. Б2. Фізична модель бази даних

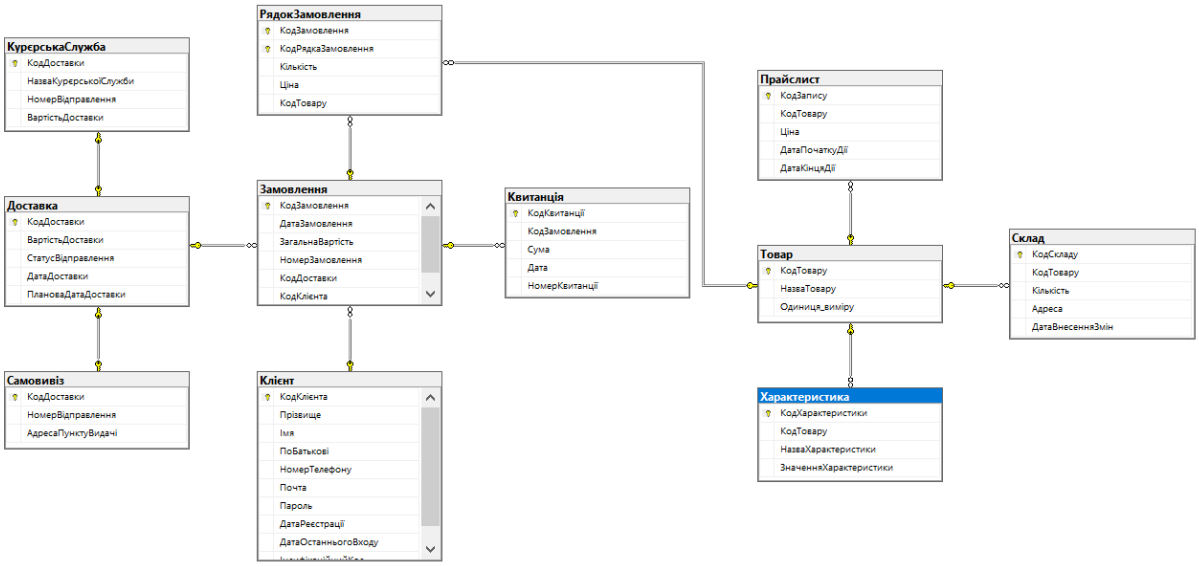
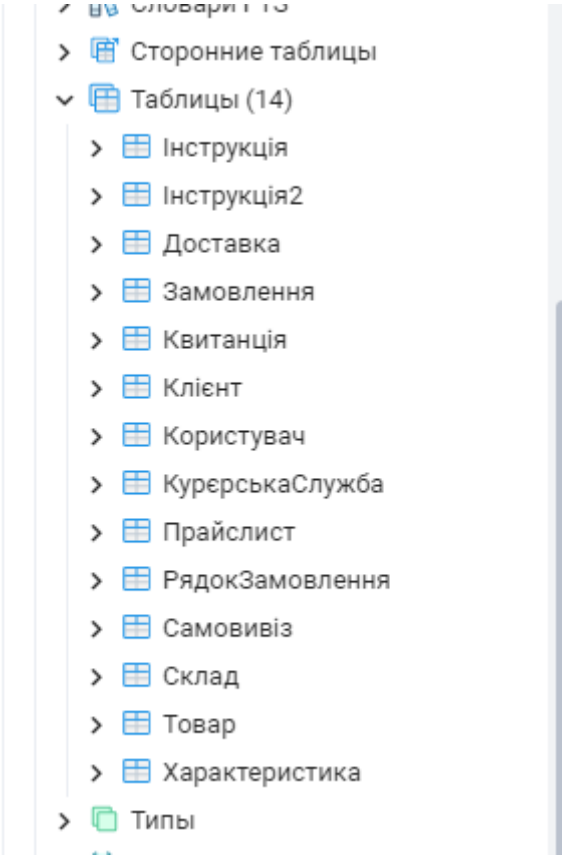


Рис. Б3. Модель бази даних

ДОДАТОК В – ЗГЕНЕРОВАНА БАЗА ДАНИХ У POSTGRESQL



ДОДАТОК Г – Макети UI-дизайну

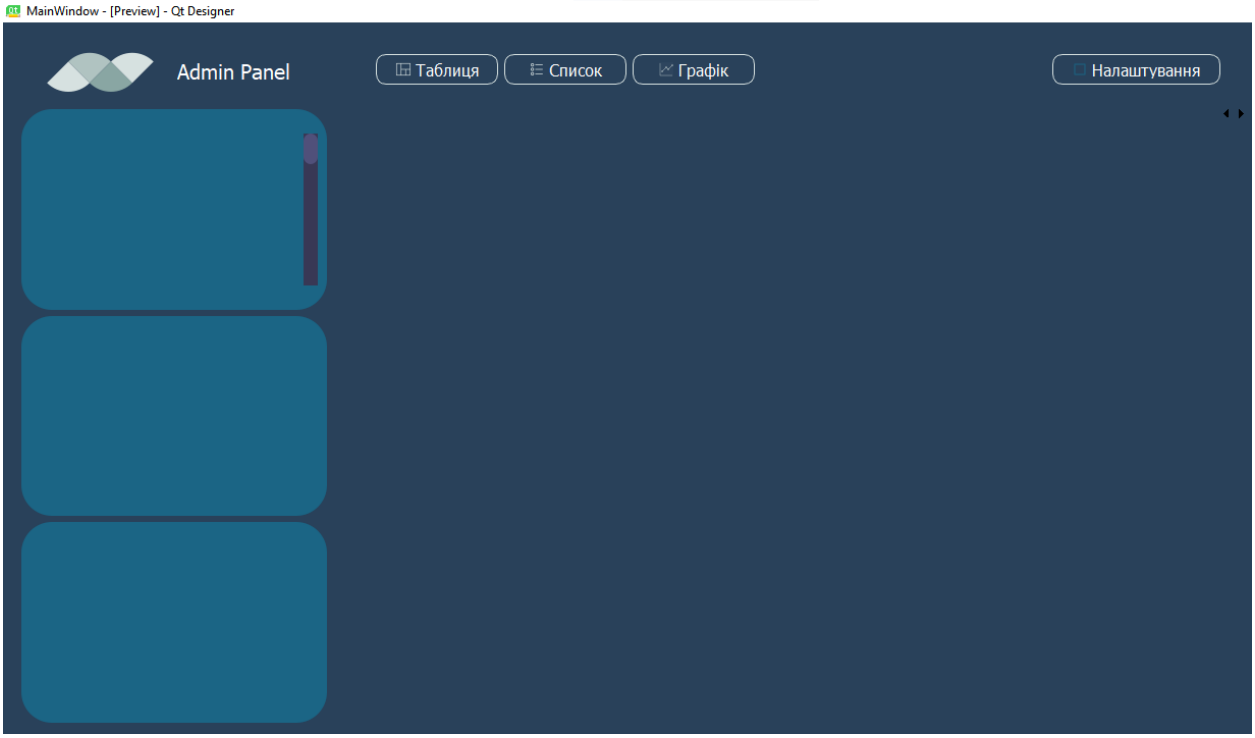


Рис. Д1. Макет ІС для операторів БД та менеджерів

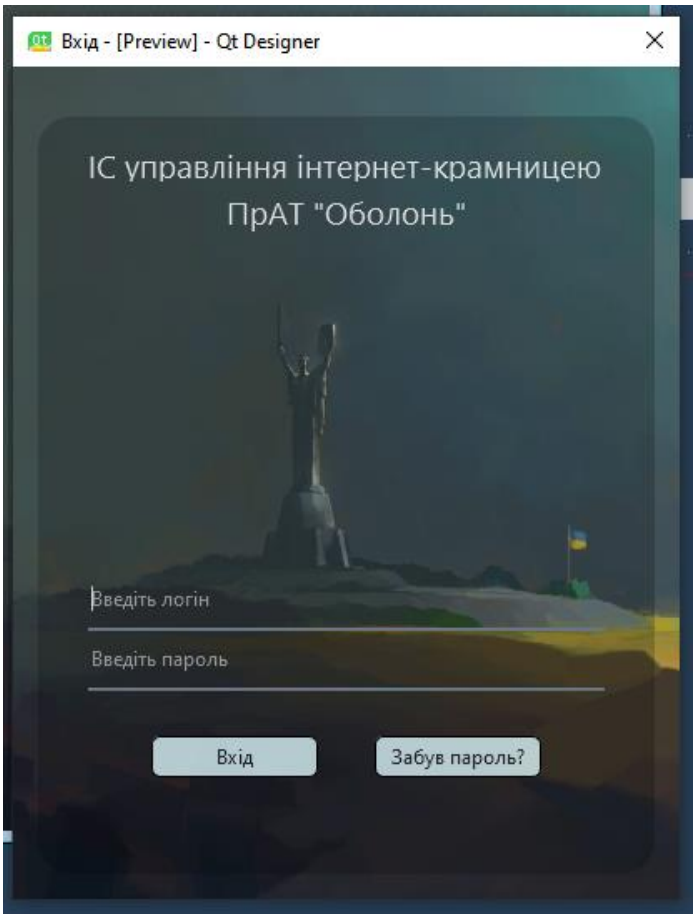


Рис. Д2. Макет для форми авторизації

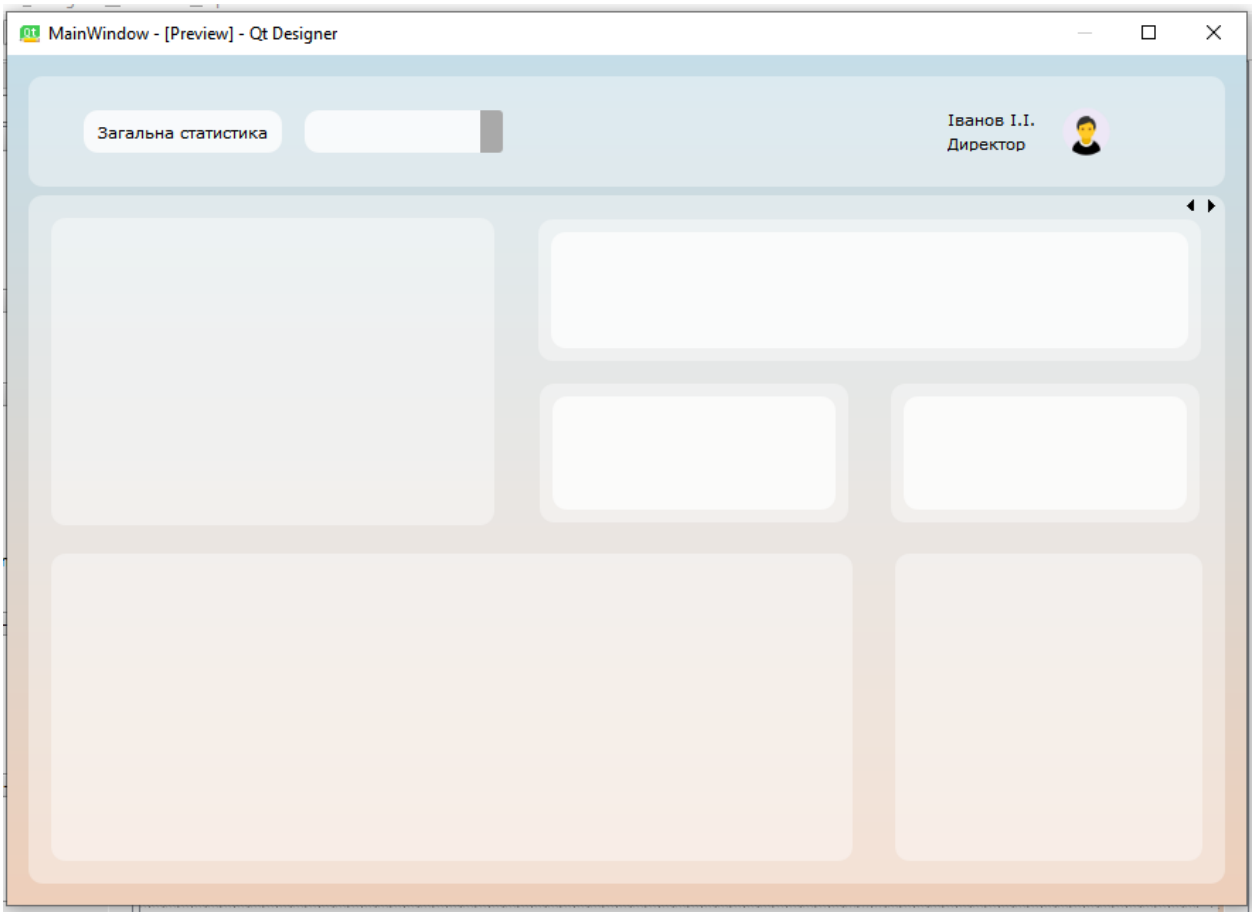


Рис. Д3. Макет форми директора

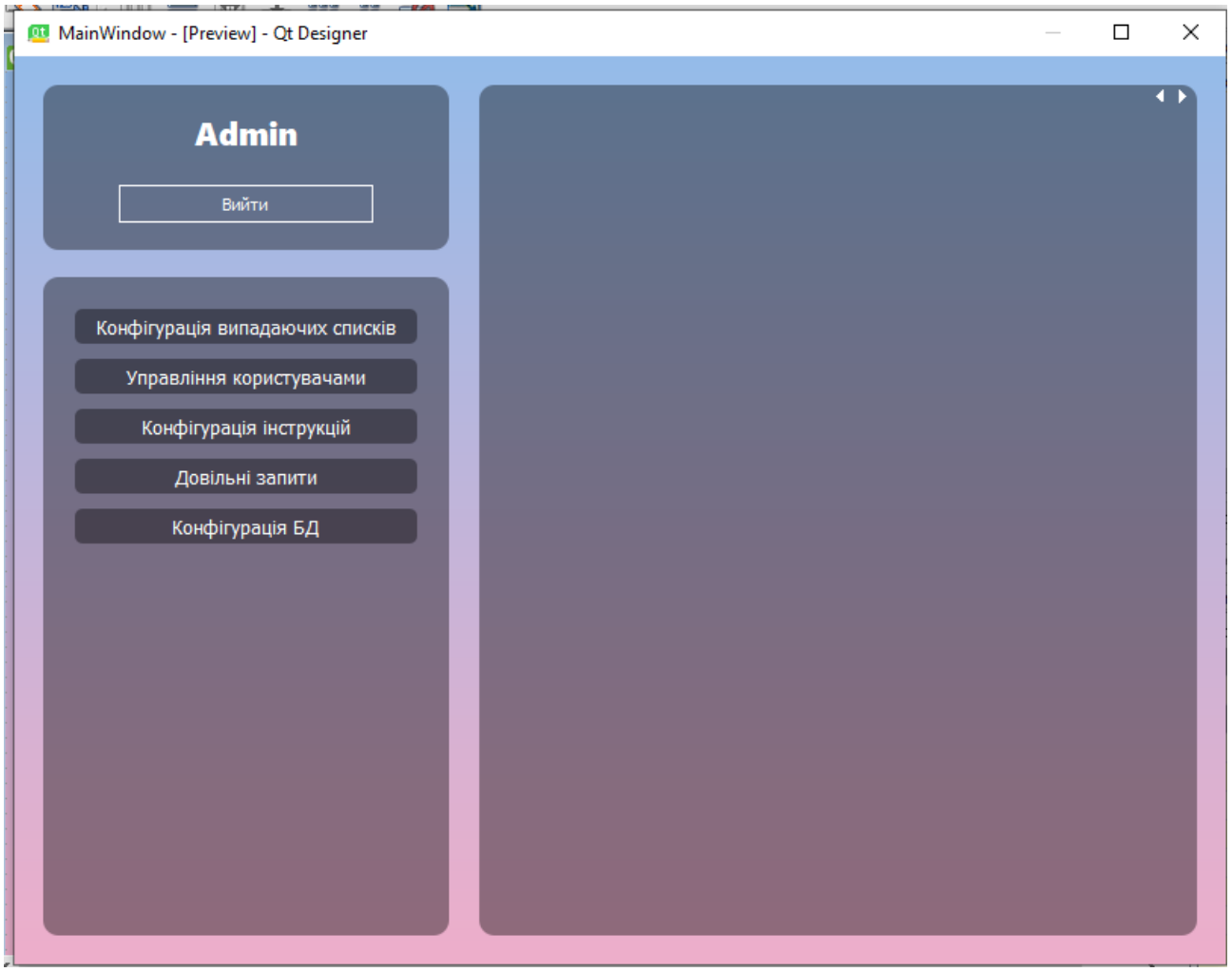


Рис. Д4. Макет форми адміністратора

ДОДАТОК Д – Приклад інтерфейсу користувача

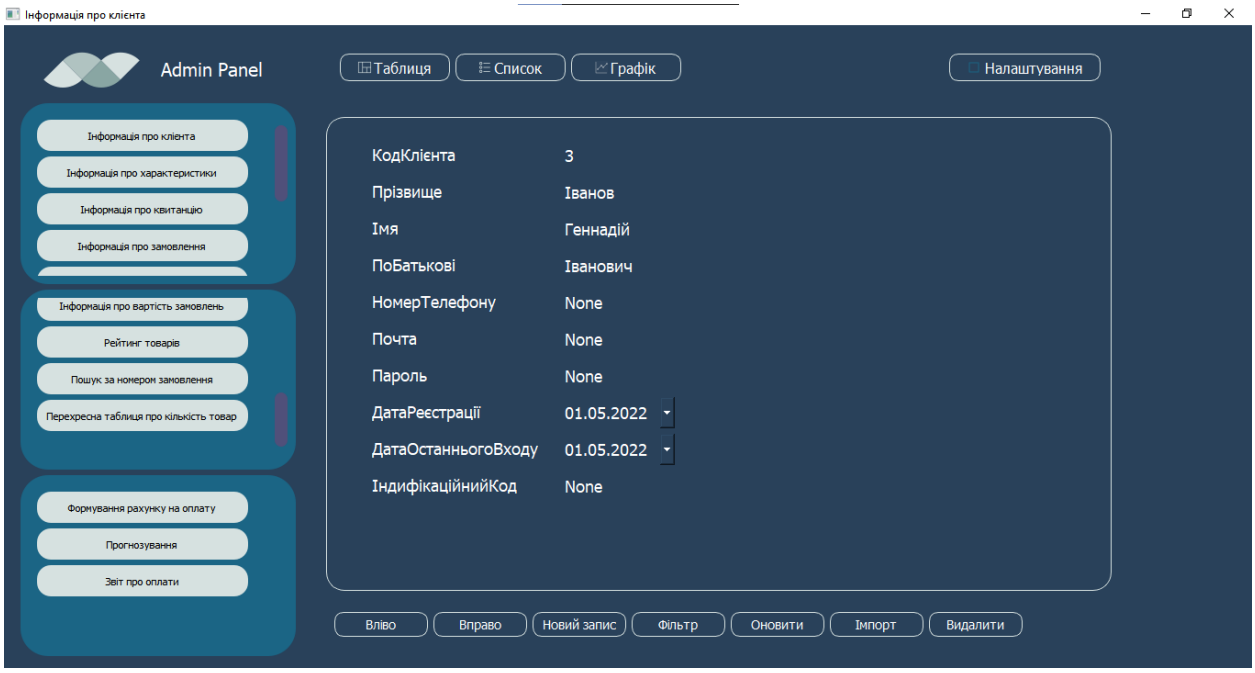


Рис. Е1. Форма відображення таблиці «Клієнт»

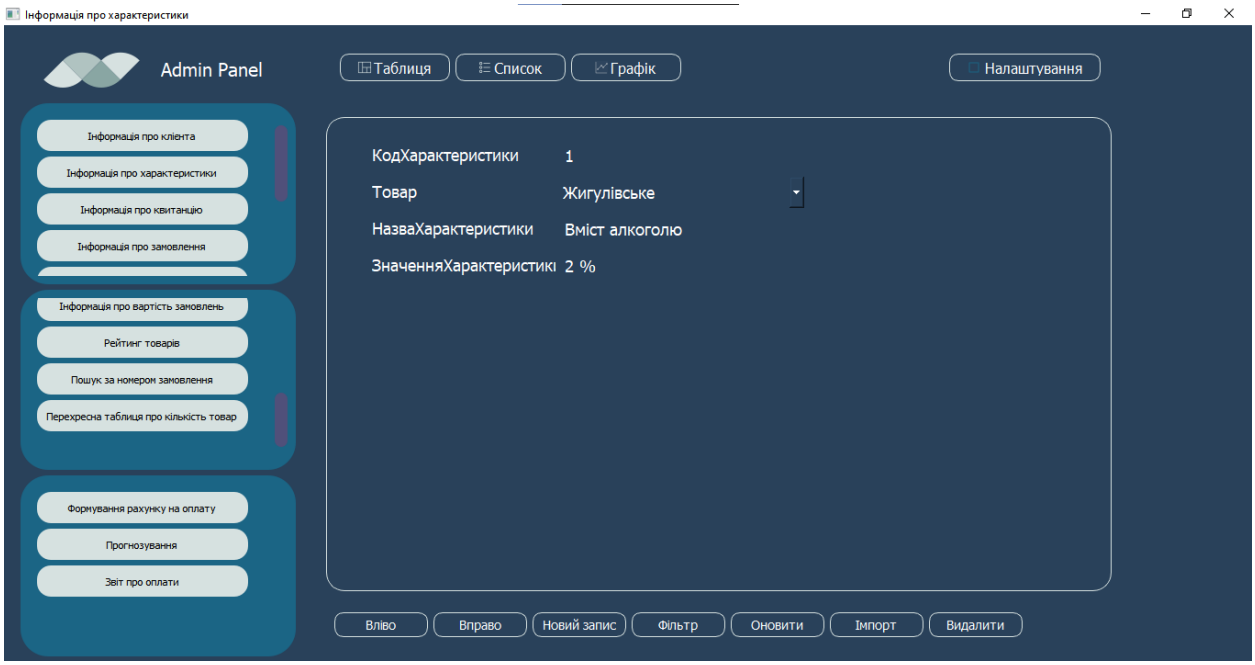


Рис. Е2. Форма відображення таблиці «Характеристика»

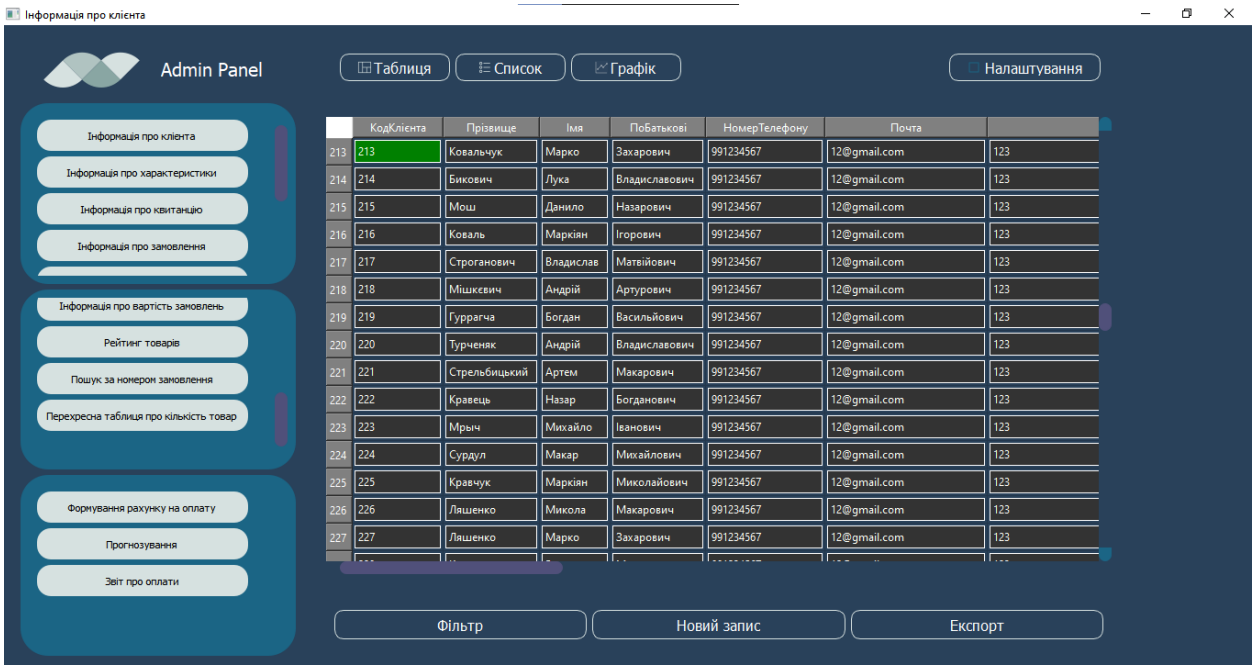


Рис. Е3. Форма відображення таблиці «Клієнт» в режимі «Таблиця»

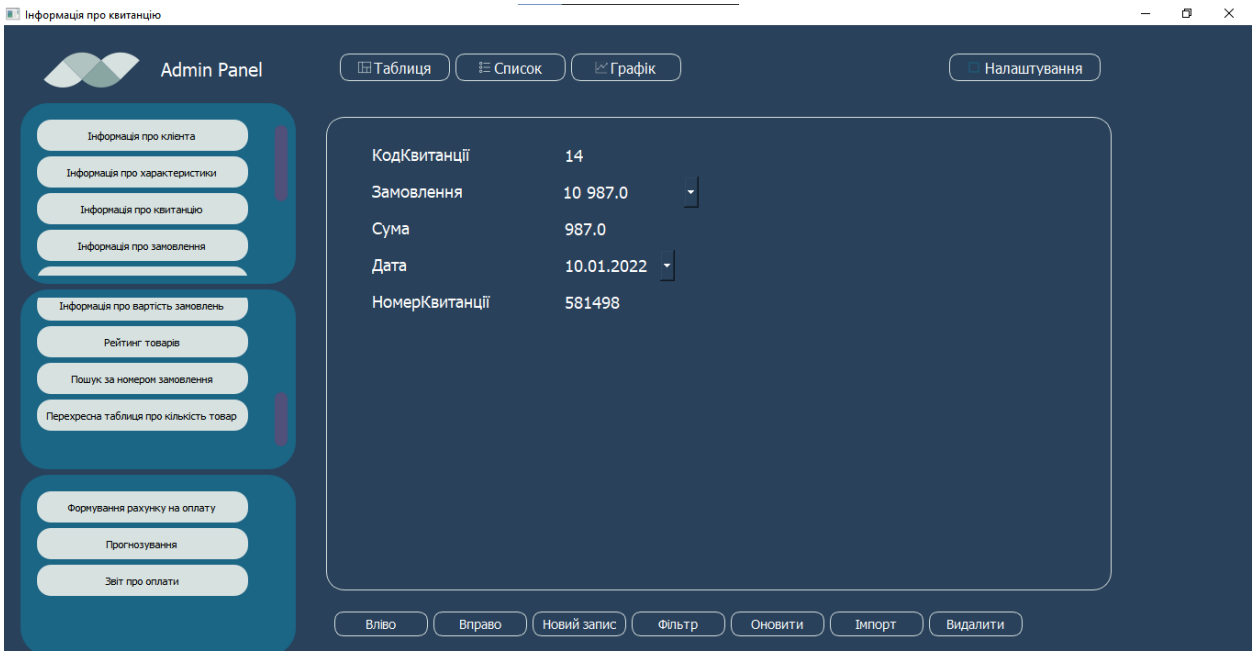


Рис. Е4. Форма відображення таблиці «Квитанція»

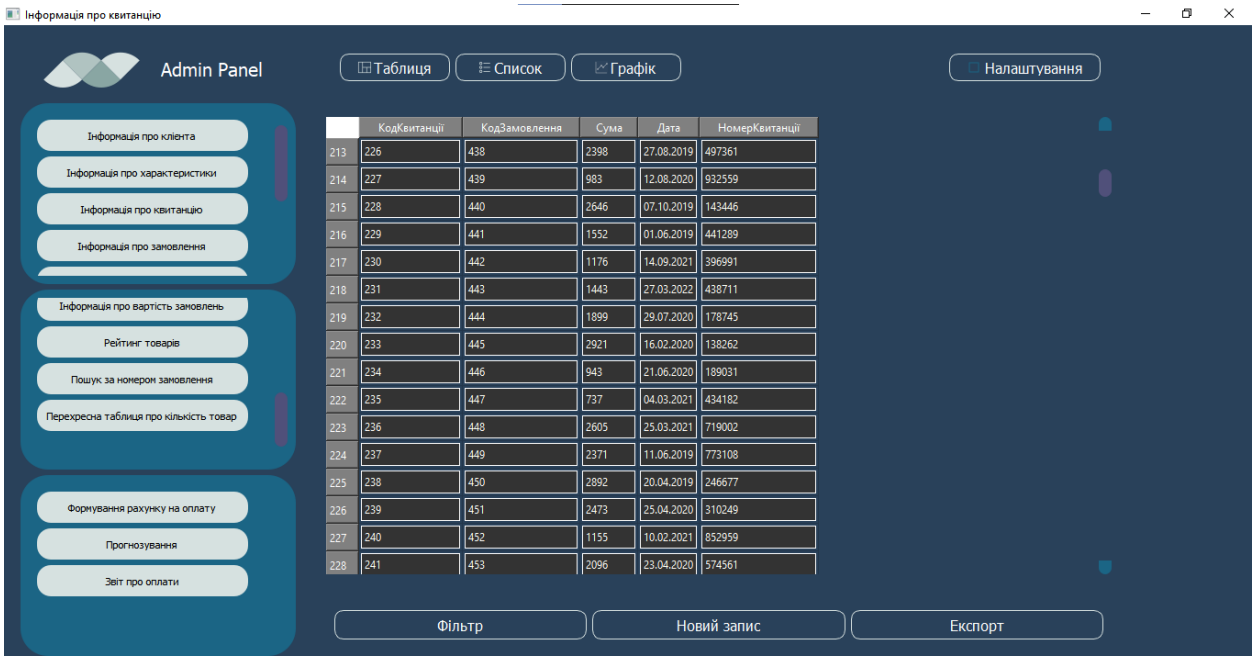


Рис. Е5. Форма відображення таблиці «Квитанція» в режимі «Таблиця»

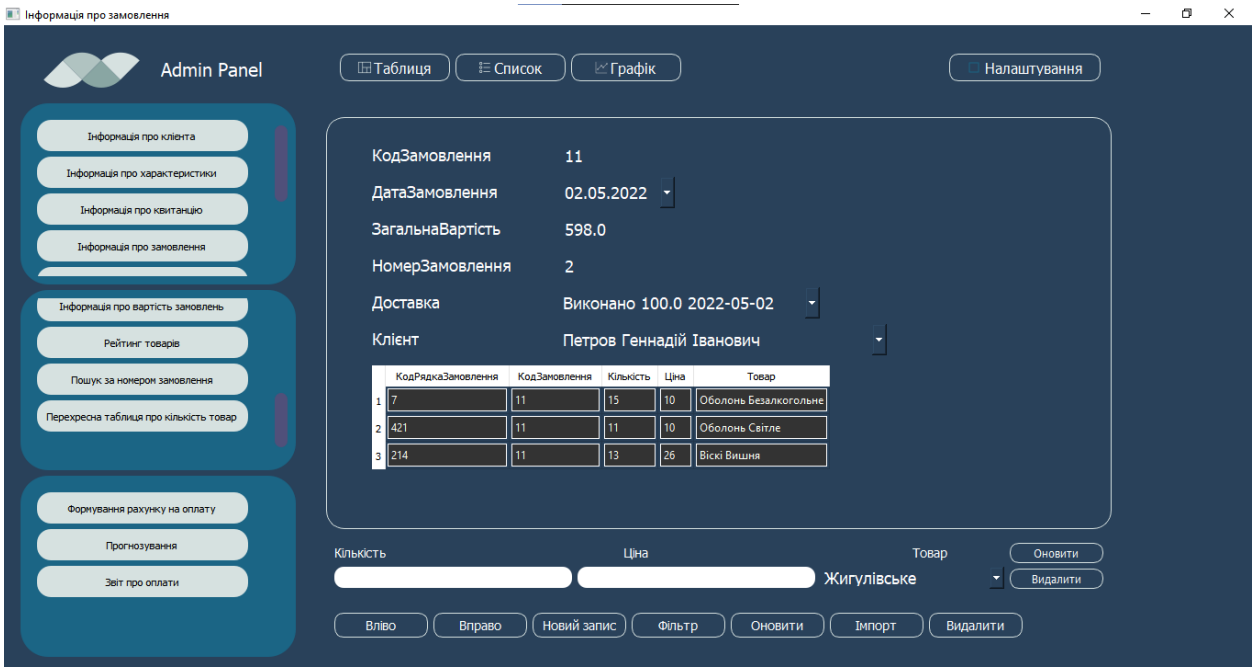
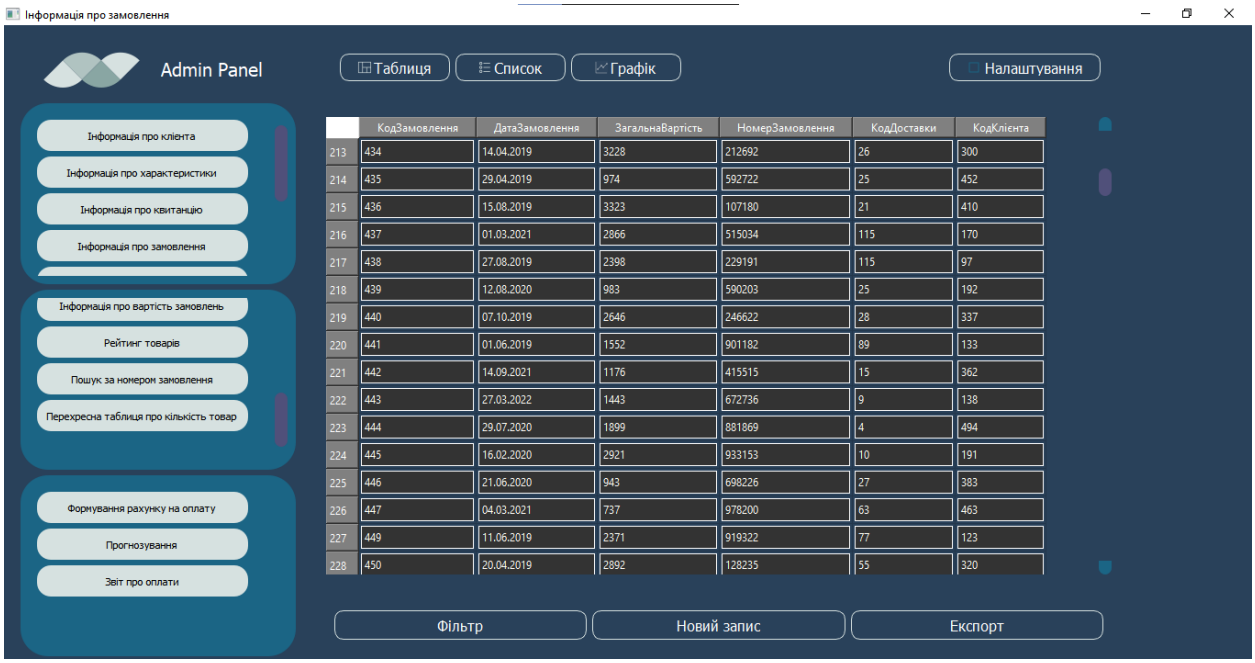
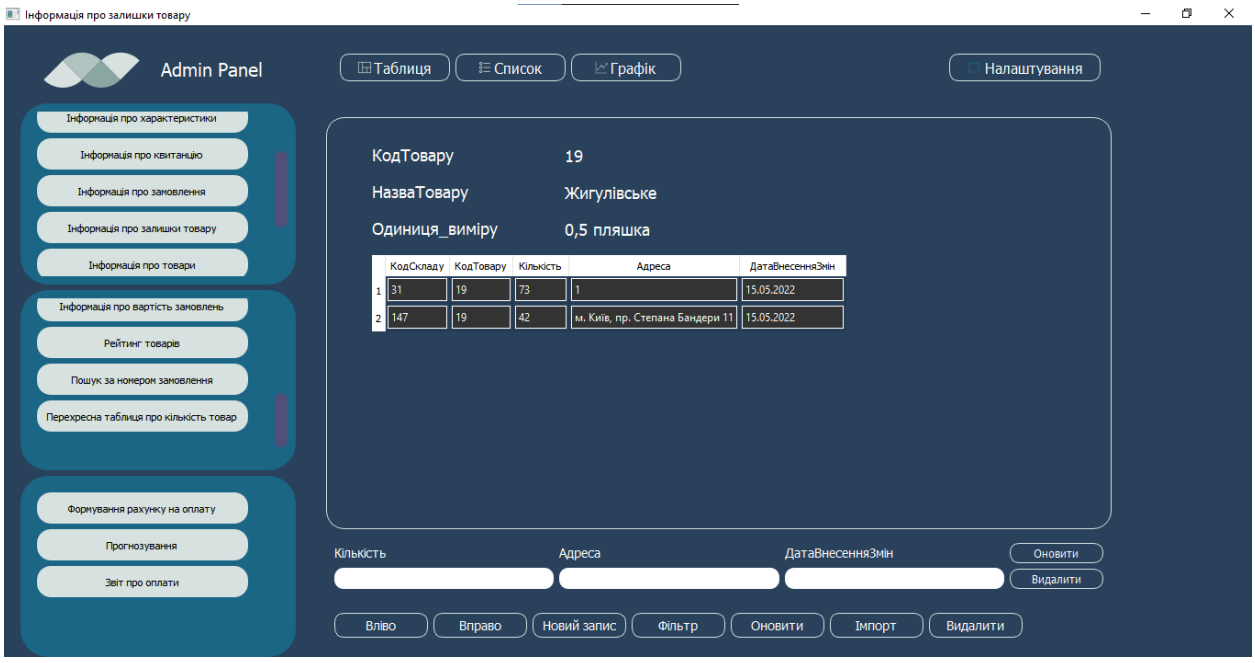


Рис. Е6. Форма відображення таблиці «Замовлення»



	КодЗамовлення	ДатаЗамовлення	ЗагальнаВартість	НомерЗамовлення	КодДоставки	КодКлієнта
213	434	14.04.2019	3228	212692	26	300
214	435	29.04.2019	974	592722	25	452
215	436	15.08.2019	3323	107180	21	410
216	437	01.03.2021	2866	515034	115	170
217	438	27.08.2019	2398	229191	115	97
218	439	12.08.2020	983	590203	25	192
219	440	07.10.2019	2646	246622	28	337
220	441	01.06.2019	1552	901182	89	133
221	442	14.09.2021	1176	415515	15	362
222	443	27.03.2022	1443	672736	9	138
223	444	29.07.2020	1899	881869	4	494
224	445	16.02.2020	2921	933153	10	191
225	446	21.06.2020	943	698226	27	383
226	447	04.03.2021	737	978200	63	463
227	449	11.06.2019	2371	919322	77	123
228	450	20.04.2019	2892	128235	55	320

Рис. Е7. Форма відображення таблиці «Замовлення» в режимі «Таблиця»



	КодСкладу	КодТовару	Кількість	Адреса	ДатаВнесенняЗмін
1	31	19	73	1	15.05.2022
2	147	19	42	м. Київ, пр. Степана Бандери 11	15.05.2022

Рис. Е8. Форма відображення таблиці «Товар»

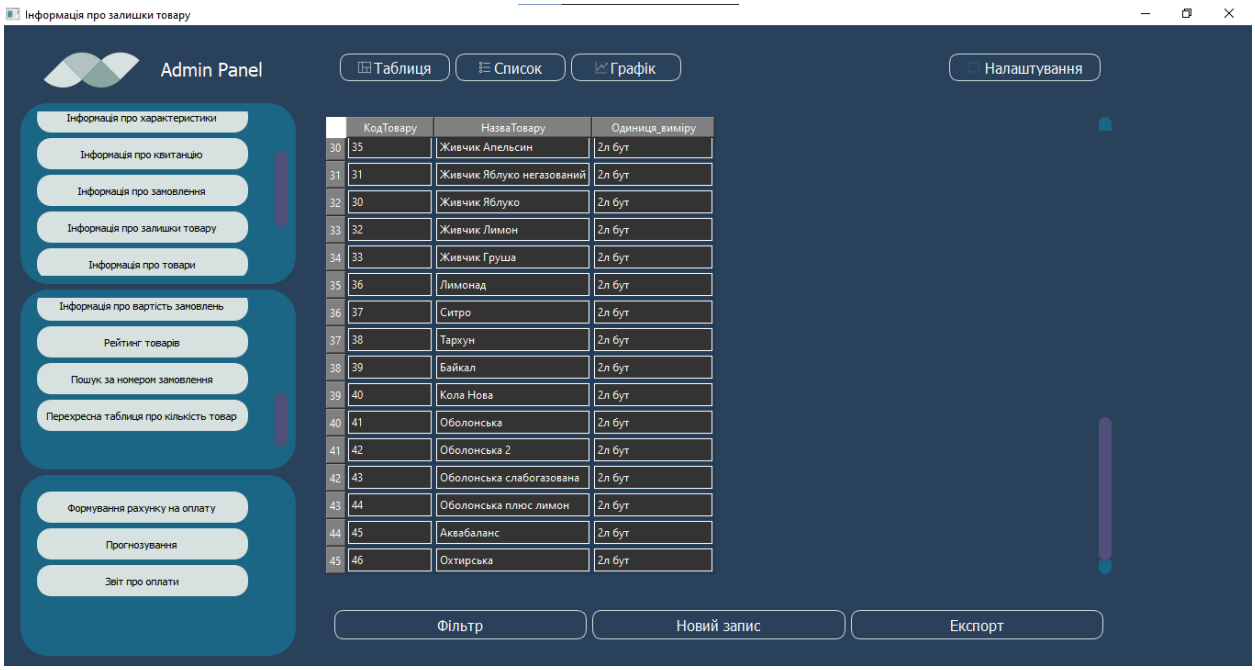


Рис. Е9. Форма відображення таблиці «Товар» в режимі «Таблиця»

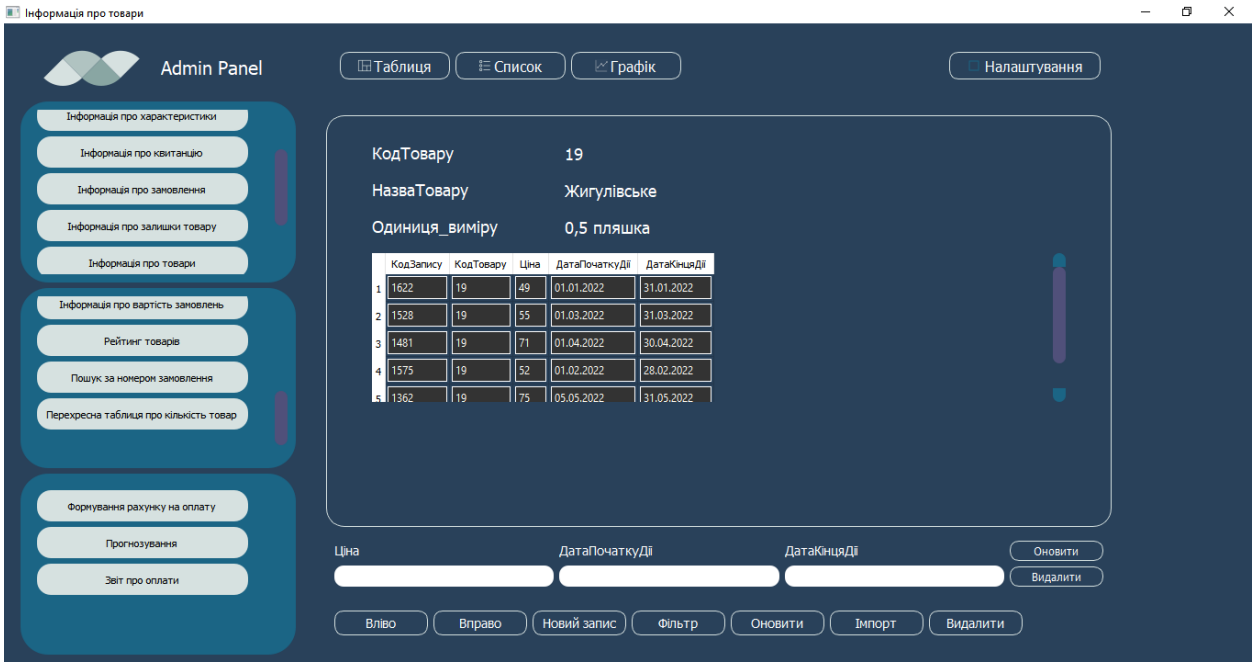


Рис. Е10. Форма відображення таблиці «Товар»

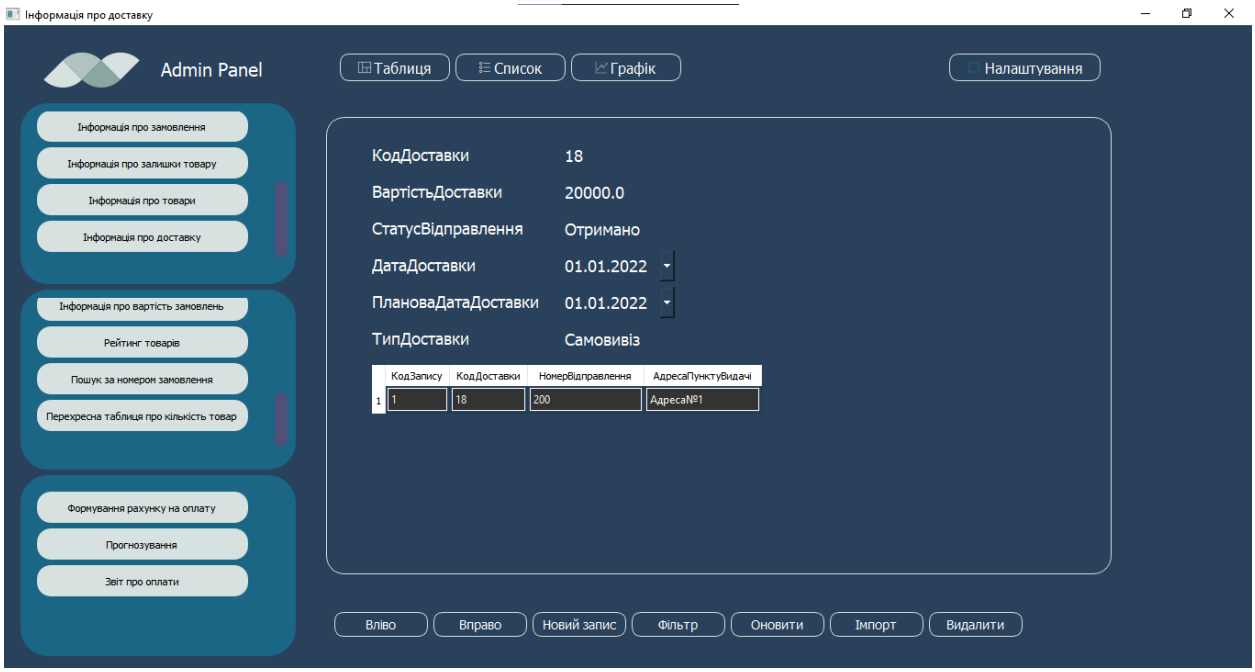


Рис. Е11. Форма відображення таблиці «Доставка»

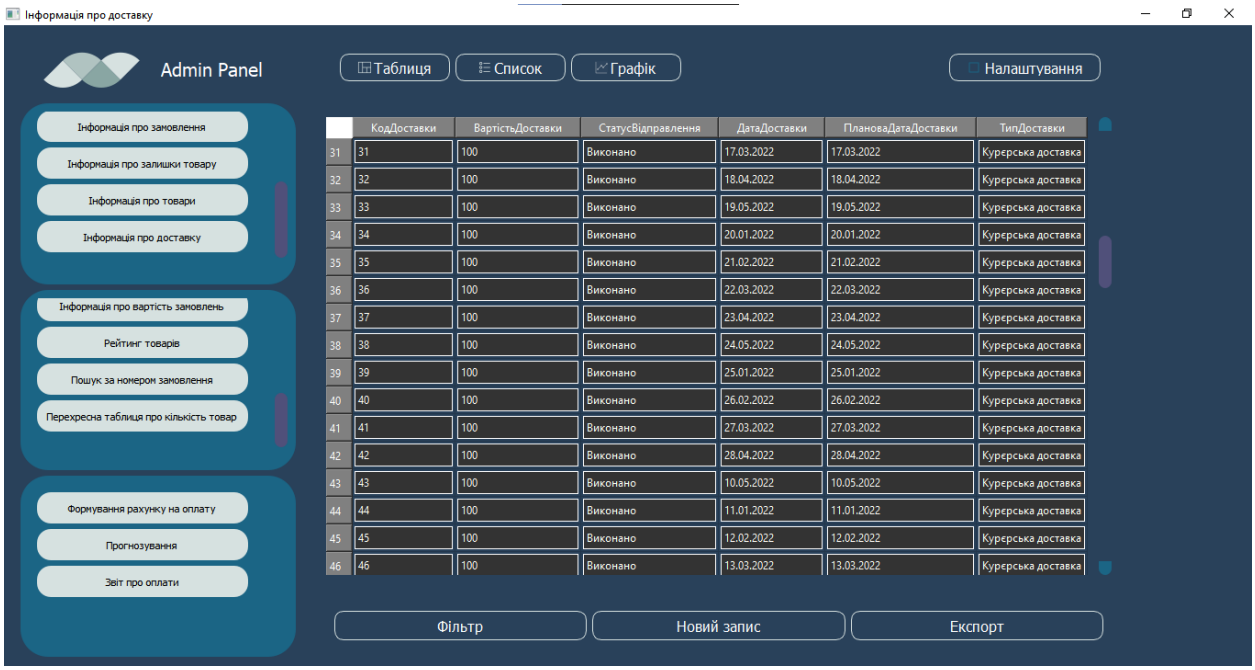
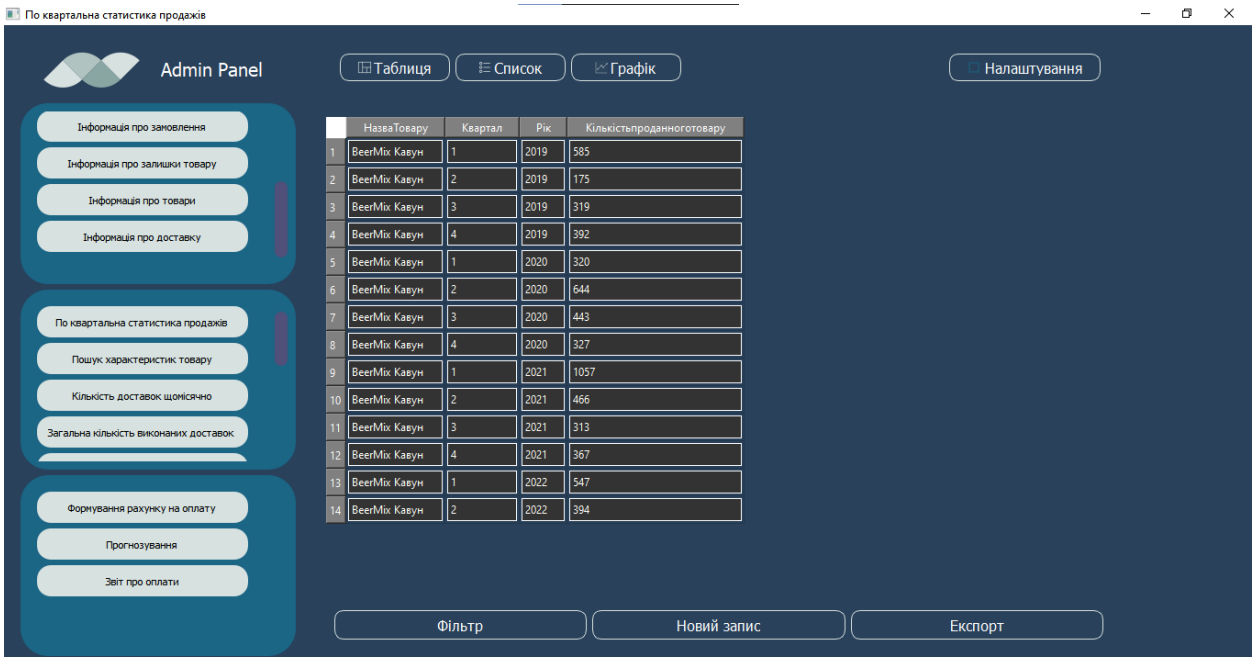


Рис. Е12. Форма відображення таблиці «Доставка» в режимі «Таблиця»



	НазваТовару	Квартал	Рік	Кількістьпроданноготовару
1	BeerMix Кавун	1	2019	585
2	BeerMix Кавун	2	2019	175
3	BeerMix Кавун	3	2019	319
4	BeerMix Кавун	4	2019	392
5	BeerMix Кавун	1	2020	320
6	BeerMix Кавун	2	2020	644
7	BeerMix Кавун	3	2020	443
8	BeerMix Кавун	4	2020	327
9	BeerMix Кавун	1	2021	1057
10	BeerMix Кавун	2	2021	466
11	BeerMix Кавун	3	2021	313
12	BeerMix Кавун	4	2021	367
13	BeerMix Кавун	1	2022	547
14	BeerMix Кавун	2	2022	394

Рис. Е13. Форма відображення запиту «По квартальна статистика продажів»

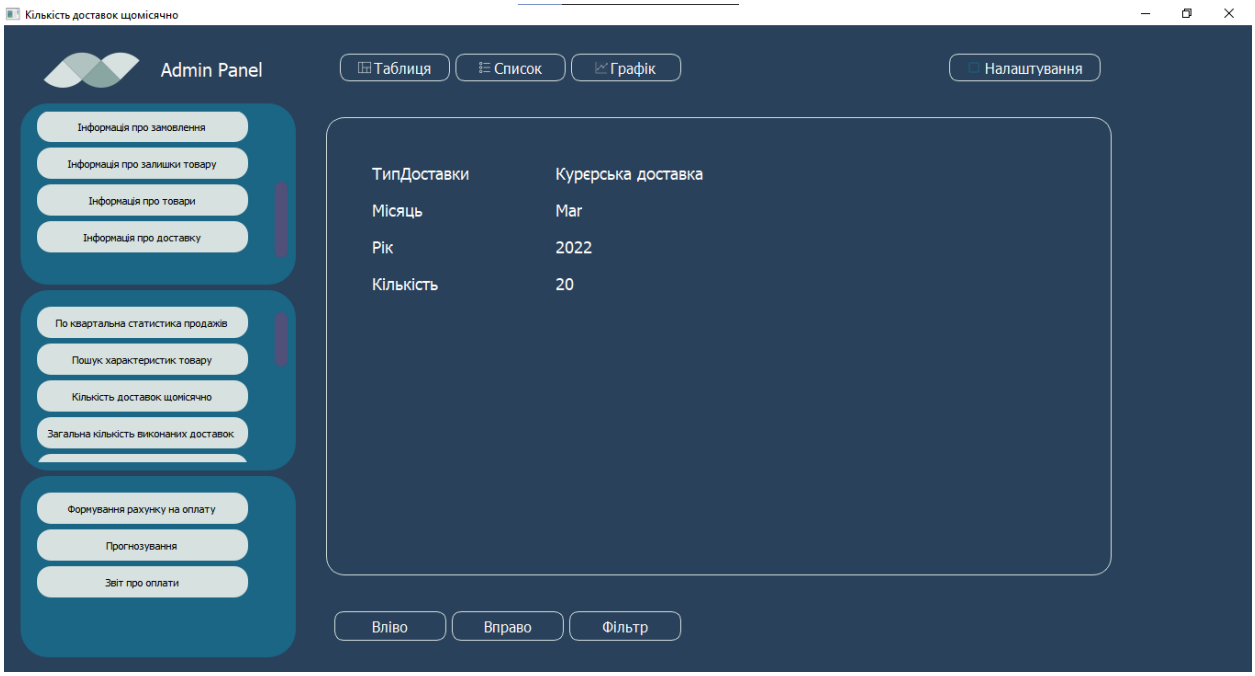


Рис. Е14. Форма відображення запиту про доставку

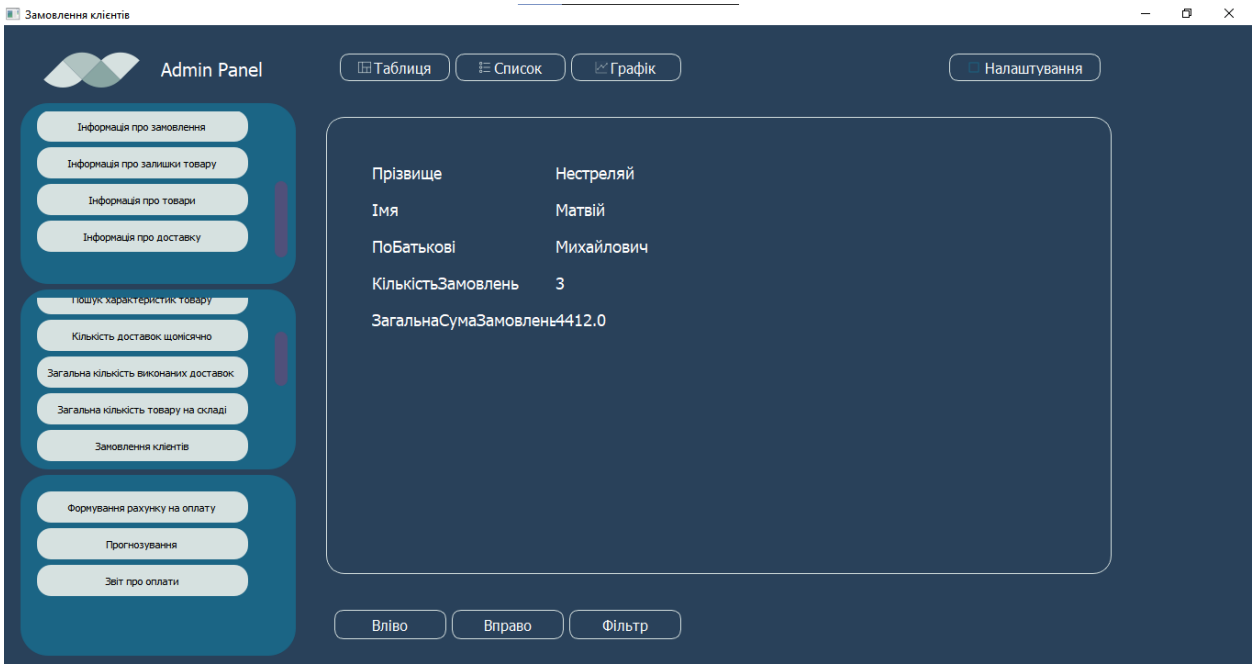


Рис. Е15. Форма відображення запиту замовлення клієнтів

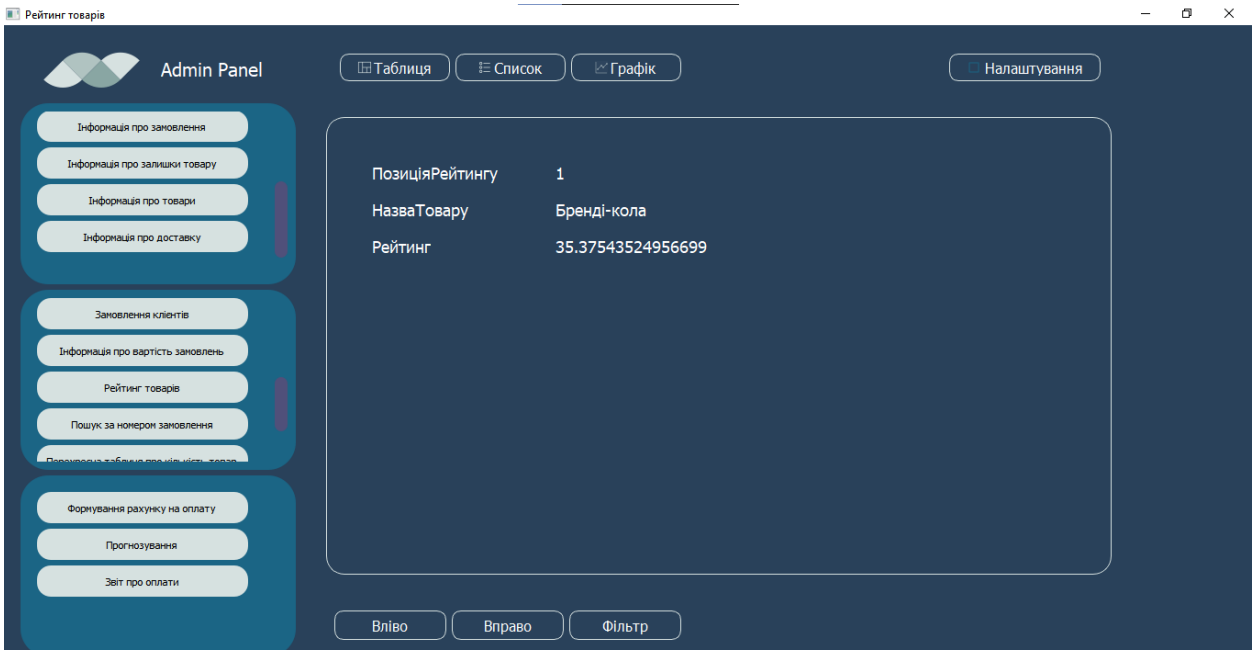


Рис. Е16. Форма відображення запиту «Рейтинг товарів»

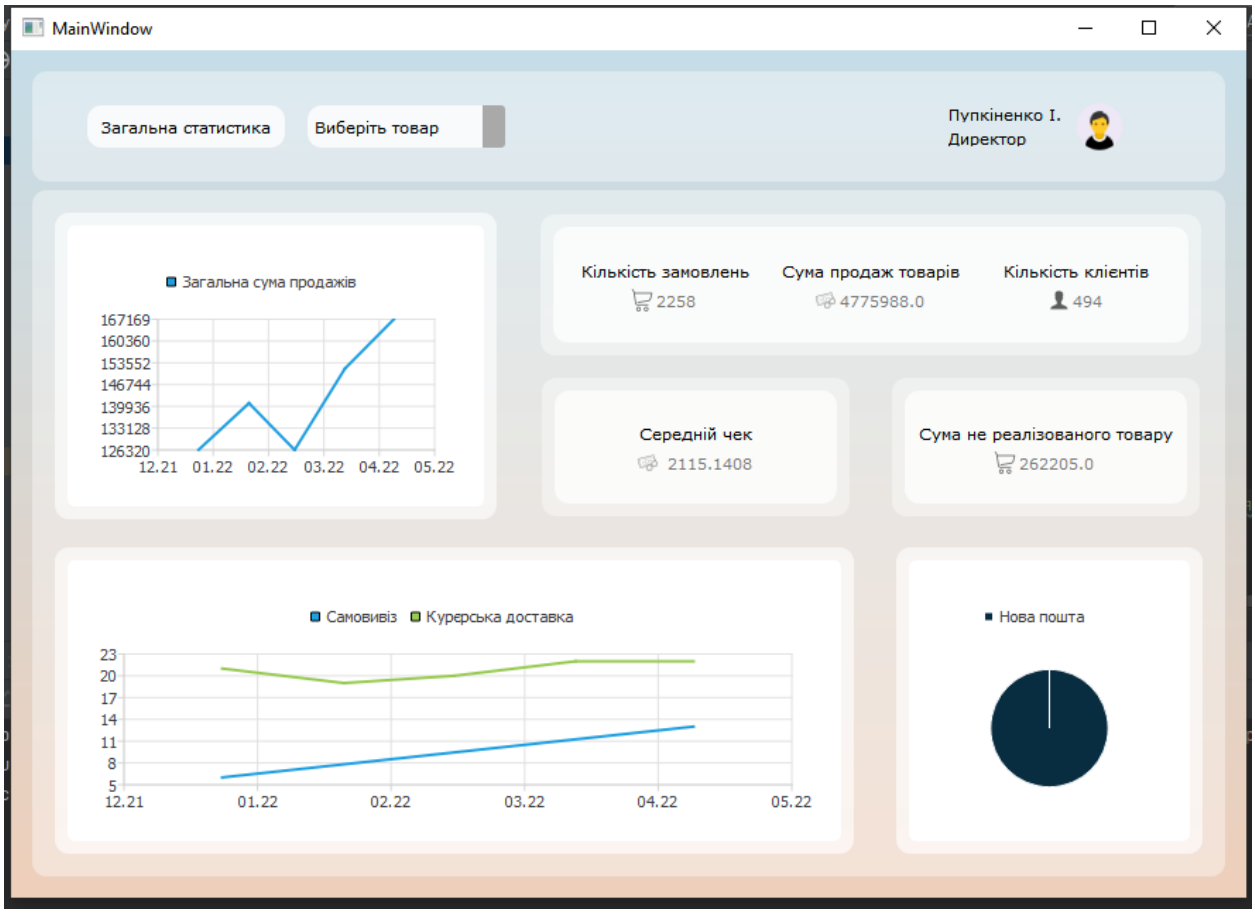


Рис. Е17. Загальний вигляд кабінету директора

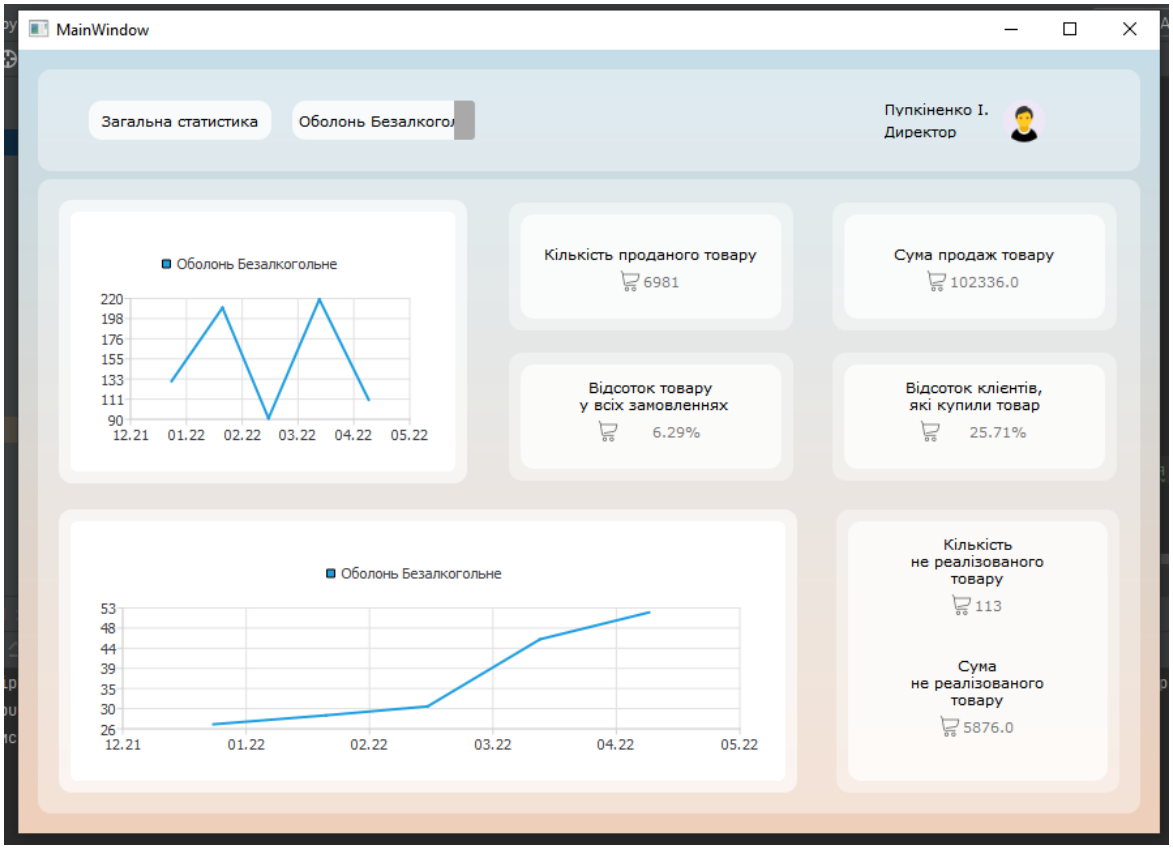


Рис. Е18. Загальний вигляд кабінету директора

Рис. Е19. Загальний вигляд кабінету директора

ДОДАТОК Е - SQL Запити

1. select Товар.НазваТовару, extract(quarter from Замовлення.ДатаЗамовлення) as quarter, extract(year from Замовлення.ДатаЗамовлення) as yyyy, sum(РядокЗамовлення.Кількість) from Замовлення inner join (РядокЗамовлення inner join Товар on Товар.КодТовару = РядокЗамовлення.КодТовару) on Замовлення.КодЗамовлення = РядокЗамовлення.КодЗамовлення group by quarter, yyyy, Товар.КодТовару having $@Товар.НазваТовару = TABLE(Товар)@$ order by Товар.КодТовару, yyyy, quarter
2. select Товар.НазваТовару, Характеристика.НазваХарактеристики, Характеристика.ЗначенняХарактеристики from Характеристика inner join Товар on Товар.КодТовару = Характеристика.КодТовару where $@Товар.НазваТовару = TABLE(Товар)@$
3. select Клієнт.Прізвище, Клієнт.Імя, Клієнт.ПоБатькові, Замовлення.НомерЗамовлення, Замовлення.ДатаЗамовлення, Замовлення.ЗагальнаВартість from Замовлення inner join Клієнт on Замовлення.КодКлієнта = Клієнт.КодКлієнта where $@Замовлення.НомерЗамовлення = TABLE(Замовлення)@$
4. select Товар.НазваТовару, SUM(Склад.Кількість) as Сума, Склад.Адреса from Товар inner join Склад on Товар.КодТовару = Склад.КодТовару GROUP BY Товар.НазваТовару, Склад.Адреса
5. select ТипДоставки, to_char(ДатаДоставки,'Mon') as mon, extract(year from ДатаДоставки) as yyyy, count(ТипДоставки) from Доставка group by ТипДоставки, mon, yyyy
6. select ТипДоставки, count(ТипДоставки) from Доставка where $@ДатаДоставки > date@$ and $@ДатаДоставки < date@$ group by ТипДоставки
7. SELECT Склад.Адреса, Товар.НазваТовару, SUM(Склад.Кількість)

FROM Склад, Товар
WHERE Товар.КодТовару = Склад.КодТовару
GROUP BY Склад.Адреса, Товар.КодТовару;

8. select concat(Клієнт.Прізвище, ' ', Клієнт.Імя,' ', Клієнт.ПоБатькові) as ПІБ, Замовлення.НомерЗамовлення, Квитанція.Дата, Квитанція.Сума, sum(Квитанція.Сума) over(partition by Клієнт.КодКлієнта) as ЗагальнаСума
from Клієнт inner join (Замовлення inner join Квитанція on Замовлення.КодЗамовлення = Квитанція.КодЗамовлення) on Замовлення.КодКлієнта = Клієнт.КодКлієнта
order by Клієнт.КодКлієнта, Дата, Сума desc
9. SELECT Клієнт.Прізвище,Клієнт.Імя,Клієнт.ПоБатькові,COUNT(Замовлення.КодЗамовлення),SUM(Замовлення.ЗагальнаВартість)
FROM Доставка, Замовлення, Клієнт
WHERE Клієнт.КодКлієнта = Замовлення.КодКлієнта AND Замовлення.КодДоставки = Доставка.КодДоставки
GROUP BY Клієнт.КодКлієнта, Доставка.СтатусВідправлення
HAVING Доставка.СтатусВідправлення = 'Виконано'
10. SELECT concat(Клієнт.Прізвище, ' ', Клієнт.Імя,' ', Клієнт.ПоБатькові), Замовлення.НомерЗамовлення, Замовлення.ЗагальнаВартість, (sum(Замовлення.ЗагальнаВартість) OVER (PARTITION BY Клієнт.КодКлієнта)) as "Сума замовлень клієнта", (Замовлення.ЗагальнаВартість / (sum(Замовлення.ЗагальнаВартість) OVER (PARTITION BY Клієнт.КодКлієнта)) *100 ::float) as "Відсоток від суми"
FROM Замовлення, Клієнт
WHERE Клієнт.КодКлієнта = Замовлення.КодКлієнта
Order by Клієнт.КодКлієнта, Замовлення.ЗагальнаВартість desc;

11. select ROW_NUMBER() over(order by count(Замовлення.КодЗамовлення)*100/(select count(*) from Замовлення)::float + count(DISTINCT Замовлення.КодКлієнта)*100/(select count(*) from Клієнт)::float desc) as "Позиція в рейтингу", Товар.НазваТовару, count(Замовлення.КодЗамовлення)*100/(select count(*) from Замовлення)::float + count(DISTINCT Замовлення.КодКлієнта)*100/(select count(*) from Клієнт)::float as Рейтинг
from Замовлення INNER JOIN (РядокЗамовлення inner join Товар on РядокЗамовлення.КодТовару = Товар.КодТовару) on Замовлення.КодЗамовлення = РядокЗамовлення.КодЗамовлення
group by Товар.КодТовару
12. select concat (Клієнт.Прізвище, ' ', Клієнт.Ім'я, ' ', Клієнт.ПоБатькові) as ПІБ, Замовлення.НомерЗамовлення, Товар.НазваТовару, РядокЗамовлення.Кількість, РядокЗамовлення.Ціна, РядокЗамовлення.Кількість*РядокЗамовлення.Ціна as Сума
from Квитанція inner join ((Замовлення inner join (РядокЗамовлення inner join Товар on Товар.КодТовару = РядокЗамовлення.КодТовару) on Замовлення.КодЗамовлення = РядокЗамовлення.КодЗамовлення) inner join Клієнт on Замовлення.КодКлієнта = Клієнт.КодКлієнта) on Квитанція.КодЗамовлення = Замовлення.КодЗамовлення
where $@Замовлення.НомерЗамовлення = TABLE(Замовлення)@$
order by Сума desc
13. select НазваКурєрськоїСлужби, count(НазваКурєрськоїСлужби) from КурєрськаСлужба group by НазваКурєрськоїСлужби
14. select concat(to_char(Замовлення.ДатаЗамовлення,'Mon'), ' ', extract(year from Замовлення.ДатаЗамовлення)) as dat, 'Загальна сума продажів' as \"Загальна сума продажів\", sum(Замовлення.ЗагальнаВартість) from Замовлення where Замовлення.ДатаЗамовлення >= '01.01.2022' group by dat

15. select ТипДоставки, concat(to_char(ДатаДоставки,'Mon'), ' ', extract(year from ДатаДоставки)) as dat, count(ТипДоставки) from Доставка group by ТипДоставки, dat'')
16. select concat(to_char(Замовлення.ДатаЗамовлення,'Mon'), ' ', extract(year from Замовлення.ДатаЗамовлення)) as dat, Товар.НазваТовару, sum(РядокЗамовлення.Кількість) from Товар INNER JOIN (РядокЗамовлення INNER JOIN Замовлення ON Замовлення.КодЗамовлення = РядокЗамовлення.КодЗамовлення) ON Товар.КодТовару = РядокЗамовлення.КодТовару where Замовлення.ДатаЗамовлення >= '01.01.2022' group by Товар.НазваТовару, dat having Товар.НазваТовару = '" + str(lastValue)
17. select Товар.НазваТовару, concat(to_char(Прайслист.ДатаКінцяДії,'Mon'), ' ', extract(year from Прайслист.ДатаКінцяДії)) as dat, Прайслист.Ціна from Товар INNER JOIN Прайслист ON Товар.КодТовару = Прайслист.КодТовару where Товар.НазваТовару = '" + str(lastValue)
18. select count(КодЗамовлення) From Замовлення
19. select sum(ЗагальнаВартість) From Замовлення
20. select count(КодКлієнта) From Клієнт
21. select sum(Склад.Кількість * Прайслист.Ціна) from Склад INNER JOIN Прайслист on Склад.КодТовару = Прайслист.КодТовару where extract(month from Прайслист.ДатаПочаткуДії) = 5

ДОДАТОК Є – Код програми

Відправка запитів до БД:

```
import psycopg2
from config import host, user, password, db_name

from messageinformation import *

import threading

from locker import *

def getcon(name):
    try:
        connection = psycopg2.connect(
            host=host,
            user=user,
            password=password,
            database=db_name
        )
        connection.autocommit = True

        with connection.cursor() as cursor:
            string = "SELECT * FROM " + name
            cursor.execute(
                string
            )
            print(string)
            Arr = []
            for i in cursor:
                Arr.append(i)
            return Arr

    except Exception as _ex:
        print("[INFO] Error with DB:", _ex)
        s = "Помилка в БД:\n" + str(_ex)
        startMessage(s)
    finally:
        if connection:
            cursor.close()
            connection.close()
            print("[INFO] DB was closed after select all")


def gethead(name):
```

```
    try:
        connection = psycopg2.connect(
            host=host,
            user=user,
            password=password,
            database=db_name
        )
        connection.autocommit = True

        with connection.cursor() as cursor:
            string = "SELECT * FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME= '" + name + "'ORDER BY ordinal_position"
            cursor.execute(
                string
            )
            Arr = []
            for i in cursor:
                Arr.append(i[3])
            return Arr

    except Exception as _ex:
        print("[INFO] Error with DB:", _ex)
        s = "Помилка в БД:\n" + str(_ex)
        startMessage(s)
    finally:
        if connection:
            cursor.close()
            connection.close()
            print("[INFO] DB was closed after select instrustion")


def insert(Array, table, DataType):
    try:
        connection = psycopg2.connect(
            host=host,
            user=user,
            password=password,
            database=db_name
        )
        connection.autocommit = True

        with connection.cursor() as cursor:
            string = "INSERT INTO " + table + " VALUES ("
            i = 0
```

```
        for a in Array:
            if i == 0:
                string += "default"
            elif DataType[i] == "character varying":
                string += "'"
                string += str(a)
                string += "'"
            elif DataType[i] == 'date':
                string += "'#"
                string += str(a)
                string += "#'"
            else:
                string += str(a)
            i = i + 1
            if i == len(Array):
                string += ")"
            else:
                string += ", "
        print(string)
        cursor.execute(
            string
        )
        if threading.currentThread().getName() != "Other":
            startMessage("Дані в БД внесено успішно.")
        else:
            pass


except Exception as _ex:
    print("[INFO] Error with DB:", _ex)
    s = "Помилка в БД:\n"
    gg = False
    for i in str(_ex):
        if i == 'D' or i == 'L':
            gg = True
            s += i
        elif gg:
            s += i

    startMessage(s)
finally:
    if connection:
        cursor.close()
        connection.close()
```

```
        print("[INFO] DB was closed after insert")



def update(Array, table, DataType):
    try:
        connection = psycopg2.connect(
            host=host,
            user=user,
            password=password,
            database=db_name
        )
        connection.autocommit = True

        with connection.cursor() as cursor:
            string = "SELECT column_name FROM
INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = '" + table +
"'ORDER BY ordinal_position"
            cursor.execute(
                string
            )
            Name = []
            for i in cursor:
                Name.append(i[0])
            string = "UPDATE " + table + " SET "
            i = 0
            for a in Array:
                if i == 0:
                    i = i + 1
                    continue

                string += Name[i]
                string += " = "
                if DataType[i] == "character varying":
                    string += "'"
                    string += str(a)
                    string += "'"
                elif DataType[i] == 'date':
                    string += "'#"
                    string += str(a)
                    string += "#'"
                else:
                    string += str(a)
                i = i + 1
```

```
            if i == len(Array):
                string += " WHERE "
                string += Name[0]
                string += " = "
                string += str(Array[0])
            else:
                string += ", "
        print(string)
        cursor.execute(
            string
        )

    except Exception as _ex:
        print("[INFO] Error with DB:", _ex)
        s = "Помилка в БД:\n" + str(_ex)
        startMessage(s)
    finally:
        if connection:
            cursor.close()
            connection.close()
            print("[INFO] DB was closed after update")
            startMessage("Дані в БД внесено успішно.")




def getType(name):
    try:
        connection = psycopg2.connect(
            host=host,
            user=user,
            password=password,
            database=db_name
        )
        connection.autocommit = True

        with connection.cursor() as cursor:
            string = "SELECT data_type FROM
INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME= '" + name +
"'ORDER BY ordinal_position"
            cursor.execute(
                string
            )
            Arr = []
```

```
            for i in cursor:
                Arr.append(i[0])
            return Arr

    except Exception as _ex:
        print("[INFO] Error with DB:", _ex)
        s = "Помилка в БД:\n" + str(_ex)
        startMessage(s)
    finally:
        if connection:
            cursor.close()
            connection.close()
            print("[INFO] DB was closed after select type columns")


def getother(query):
    try:
        connection = psycopg2.connect(
            host=host,
            user=user,
            password=password,
            database=db_name
        )
        connection.autocommit = True

        with connection.cursor() as cursor:
            cursor.execute(
                query
            )
            Arr = []
            for i in cursor:
                Arr.append(i)
            return Arr

    except Exception as _ex:
        print("[INFO] Error with DB:", _ex)
        s = "Помилка в БД:\n" + str(_ex)
        #startMessage(s)
    finally:
        if connection:
            cursor.close()
            connection.close()
            #print("[INFO] DB was closed after select all")
```

```
def getKey(Table):
    try:
        connection = psycopg2.connect(
            host=host,
            user=user,
            password=password,
            database=db_name
        )
        connection.autocommit = True

        with connection.cursor() as cursor:
            query = "select sequence_name from information_schema.sequences where
sequence_name LIKE '"
            query += Table
            query += "%'"
            cursor.execute(
                query
            )
            Arr = []
            for i in cursor:
                Arr.append(i[0])
            query = "select last_value from "
            query += str(Arr[0])
            cursor.execute(
                query
            )
            Arr.clear()
            for i in cursor:
                Arr.append(i[0])
            return Arr

    except Exception as _ex:
        print("[INFO] Error with DB after getKey:", _ex)
        s = "Помилка в БД:\n" + str(_ex)
        startMessage(s)
    finally:
        if connection:
            cursor.close()
            connection.close()
            print("[INFO] DB was closed after select all")
```

```
def delete(column, value, table, DataType):
    try:
        connection = psycopg2.connect(
            host=host,
            user=user,
            password=password,
            database=db_name
        )
        connection.autocommit = True

        with connection.cursor() as cursor:
            string = "DELETE FROM " + table + " WHERE " + column + " = "
            if DataType == "character varying":
                string += """
                string += str(value)
                string += """
            elif DataType == 'date':
                string += "'#"
                string += str(value)
                string += "#'"
            else:
                string += str(value)
            print(string)
            cursor.execute(
                string
            )
            startMessage("Дані в БД видалено успішно.")
            return True

    except Exception as _ex:
        print("[INFO] Error with DB after delete:", _ex)
        s = "Помилка в БД:\n" + str(_ex)
        startMessage(s)

    finally:
        if connection:
            cursor.close()
            connection.close()
            print("[INFO] DB was closed after delete")


def getUserStatus(login, passw):
    try:
```

```
        connection = psycopg2.connect(
            host=host,
            user=user,
            password=password,
            database=db_name
        )
        connection.autocommit = True

        with connection.cursor() as cursor:
            query = "select Права from Користувач where \"Логін\" = '"
            query += login
            query += "' and \"Пароль\" = '"
            query += passw
            query += "'"
            print(query)
            cursor.execute(
                query
            )
            Arr = []
            for i in cursor:
                Arr.append(i[0])
            return Arr

    except Exception as _ex:
        print("[INFO] Error with DB after getKey:", _ex)
        s = "Помилка в БД:\n" + str(_ex)
        startMessage(s)
    finally:
        if connection:
            cursor.close()
            connection.close()
```

Створення графіку:

```
import random
from datetime import datetime

from PyQt5 import QtCore, QtGui, QtWidgets, Qt
from PyQt5.QtChart import *
from PyQt5.QtCore import QEvent, QPointF
from qr import qrGenerate

import messageinformation
```

```
# boxes = []
q = 0


def clear():
    global q
    q = 0



class LineSeries(QLineSeries):
    def __init__(self, *args, **kwargs):
        QLineSeries.__init__(self, *args, **kwargs)
        self.pressed.connect(self.on_pressed)
        global q
        self.name = q
        q += 1

    def on_pressed(self, point):
        qrGenerate.setData(self.name)


class TimeChart():
    def __init__(self, ui, titleText):
        super(TimeChart, self).__init__()

        self.ui = ui
        self.seriesS = []
        self.plot = QChart()
        self.chart_view = QChartView(self.plot)
        self.chart_view.setRenderHint(QtGui.QPainter.Antialiasing)
        if titleText is not None:
            self.plot.setTitle(str(titleText))



        # Setting X-axis
        self.axis_x = QDateTimeAxis()
        self.axis_x.setTickCount(11)
        self.axis_x.setFormat("MM.yy")
        self.axis_x.setMax(datetime.strptime('202206', '%Y%m'))
        self.axis_x.setMin(datetime.strptime('202105', '%Y%m'))

        # Setting Y-axis
```

```
        self.axis_y = QValueAxis()
        self.axis_y.setTickCount(7)
        self.axis_y.setLabelFormat("%i")
        self.axis_y.setMax(60)
        self.axis_y.setMin(0)

        if self.ui.layout() is None:
            vbox = QtWidgets.QVBoxLayout()
            vbox.addWidget(self.chart_view)
            vbox.setContentsMargins(0, 0, 0, 0)
            self.ui.setLayout(vbox)
        else:
            item = self.ui.layout().takeAt(0)
            widget = item.widget()
            if widget is not None:
                widget.setParent(None)
            self.ui.layout().addWidget(self.chart_view)

    def add(self, x, format, y, index):
        self.seriesS[index].append(float(QtCore.QDateTime.fromString(x,
format).toMSecsSinceEpoch()), y, )



    def addLine(self, label):
        self.series2 = LineSeries()
        self.seriesS.append(self.series2)
        self.series2.setName(label)
        self.plot.addSeries(self.series2)


        self.plot.setAxisX(self.axis_x, self.series2)
        self.plot.setAxisY(self.axis_y, self.series2)


    def setX(self, x1, x2, form, tick):
        self.axis_x.setMax(datetime.strptime(x2, form))
        self.axis_x.setMin(datetime.strptime(x1, form))
        self.axis_x.setTickCount(int(tick))


    def setY(self, y1, y2):
        self.axis_y.setMax(y2)
        self.axis_y.setMin(y1)
```

```
    def save(self):
        from PyQt5.QtGui import QPixmap
        q = QPixmap(self.chart_view.grab())
        q.save("a.png", "PNG")
```

Генерація QR:

```
import qrcode
import sys

from PIL.ImageQt import ImageQt
from PyQt5.QtGui import QPixmap
from PyQt5.QtWidgets import QMainWindow, QApplication, QWidget
from PyQt5 import QtWidgets, QtGui, QtCore

import messageinformation
import prognoz.start
import startApp
import test2
from directorForm import startWindow as dirForm
from currentWindow import currentWindow

data = ""

img = qrcode.make(data)

def setData(number):
    global data
    if currentWindow() == "DirectorForm":
        if dirForm.ui.stackedWidget.currentIndex() == 0:
            if number == 0:
                data = ""
                for i in dirForm.data1:
                    for j in i:
                        data += str(j)
                        data += " "
                    data += "\n"
                showImg()
            elif number == 'QBar':
                data = ""
                for i in dirForm.data3:
                    for j in i:
                        data += str(j)
```

```
                data += " "
            data += "\n"
        showImg()
    else:
        data = ""
        for i in dirForm.data2:
            for j in i:
                data += str(j)
                data += " "
            data += "\n"
        showImg()
else:
    if number == 0:
        data = ""
        for i in dirForm.data4:
            for j in i:
                data += str(j)
                data += " "
            data += "\n"
        showImg()
    else:
        data = ""
        for i in dirForm.data5:
            for j in i:
                data += str(j)
                data += " "
            data += "\n"
        showImg()
elif currentWindow() == "DBForm":
    if test2.ui.stackedWidget.currentIndex() == 1:
        if number == 0:
            d2 = prognoz.start.convertToQuarter(prognoz.start.prog)
            data = str(d2[0][0])
            data += " \n"
            for i in d2:
                for j in range(1, len(i)):
                    data += str(i[j])
                    data += " "
                data += "\n"
            showImg()
        elif number == 1:
            d2 = prognoz.start.convertToQuarter(prognoz.start.trend)
            data = str(d2[0][0])
            data += " \n"
```

```
            for i in d2:
                for j in range(1, len(i)):
                    data += str(i[j])
                    data += " "
                data += "\n"
            showImg()
        elif number == 2:
            d2 = prognoz.start.convertToQuarter(prognoz.start.dataSe)
            data = str(d2[0][0])
            data += " \n"
            for i in d2:
                for j in range(1, len(i)):
                    data += str(i[j])
                    data += " "
                data += "\n"
            showImg()
    else:
        pass


class myGrid(QtWidgets.QWidget):
    popuphidden = QtCore.pyqtSignal()

    def __init__(self):
        super(myGrid, self).__init__()
        self.setWindowFlags(QtCore.Qt.SplashScreen |
QtCore.Qt.FramelessWindowHint | QtCore.Qt.WindowStaysOnTopHint)
        self.setMinimumSize(QtCore.QSize(300, 300))
        self.animation = QtCore.QPropertyAnimation(self, b"windowOpacity", self)
        self.timer = QtCore.QTimer()
        self.setupUi()
        vbox = QtWidgets.QVBoxLayout()

        im = ImageQt(img).copy()
        pixmap = QtGui.QPixmap.fromImage(im)
        pixmap_resized = pixmap.scaled(300, 300, QtCore.Qt.KeepAspectRatio)
        label = QtWidgets.QLabel()
        label.setPixmap(pixmap_resized)
        vbox.addWidget(label)
        self.setStyleSheet("background: white")



        resetBtn = QtWidgets.QPushButton()
```

```
        resetBtn.setText("Закрити")
        resetBtn.clicked.connect(self.hideAnimation)
        resetBtn.setMinimumHeight(30)
        resetBtn.setMaximumHeight(30)
        vbox.addWidget(resetBtn)
        resetBtn.setStyleSheet("QPushButton {background-color: rgba(0, 0, 0, 0);
border: 1px solid black; border-radius: 10px;} QPushButton:hover{background-
color: black; color: white;}")

        self.setLayout(vbox)


    def setupUi(self):
        pass


    def mousePressEvent(self, event):
        if event.button() == QtCore.Qt.LeftButton:
            self.dragPosition = event.globalPos() - self.frameGeometry().topLeft()
            event.accept()

    def mouseMoveEvent(self, event):
        if event.buttons() == QtCore.Qt.LeftButton:
            self.move(event.globalPos() - self.dragPosition)
            event.accept()

    def show(self):
        self.setWindowOpacity(0.0)
        self.animation.setDuration(1500)
        self.animation.setStartValue(0.0)
        self.animation.setEndValue(1.0)
        QtWidgets.QWidget.show(self)
        self.animation.start()
        self.timer.start(5000)

    def hideAnimation(self):
        self.timer.stop()
        self.animation.setDuration(200)
        self.animation.setStartValue(1.0)
        self.animation.setEndValue(0.0)
        self.animation.start()
        self.hide()
```

```
    def hide(self):
        if self.windowOpacity() == 0:
            QtWidgets.QWidget.hide(self)
            self.popuphidden.emit()


def showImg():
    global img
    img = qrcode.make(data)
    w = myGrid()
    w.show()
```

Фільтрація:

```
import datetime


def EditData(onedata, types, inst, filterStyle, FromBefore):
    data = []
    if not FromBefore:
        if filterStyle: #якщо точно копія
            #print("From 1")
            for i in onedata:
                addToData = True
                for j in i:
                    if str(j) != str(inst[i.index(j)][0]) and inst[i.index(j)][0] is not None:
                        addToData = False
                        print(str(inst[i.index(j)][0]))
                if addToData:
                    data.append(i)
        else:
            #print("From 2")
            for i in onedata:
                addToData = False
                for j in i:
                    if str(j) == str(inst[i.index(j)][0]) and inst[i.index(j)][0] is not None:
                        addToData = True
                if addToData:
                    data.append(i)
    else:
        if filterStyle: #якщо точно копія
            #print("From 3")
            for i in onedata:
                addToData = True
```

```
            jpos = 0
            for j in i:
                #print("0 = ", str(inst[jpos][0]), " jpos = ", jpos)
                #print("1 = ", str(inst[jpos][1]))
                if inst[jpos][1] is not None and inst[jpos][0] is not None:
                    if str(type(j)) == "<class 'int'>" or str(type(j)) == "<class
'decimal.Decimal'>":
                        if j >= int(inst[jpos][0]) and j <= int(inst[jpos][1]):
                            pass
                        else:
                            addToData = False
                    elif str(type(j)) == "<class 'float'>":
                        if j >= float(inst[jpos][0]) and j <= float(inst[jpos][1]):
                            pass
                        else:
                            addToData = False
                    elif str(type(j)) == "<class 'datetime.date'>":
                        dt1 = datetime.datetime.strptime(inst[jpos][0], '%Y-%m-%d')
                        dt2 = datetime.datetime.strptime(inst[jpos][1], '%Y-%m-%d')
                        if j >= dt1.date() and j <= dt2.date():
                            pass
                        else:
                            addToData = False
                    elif str(j) != str(inst[jpos][0]):
                        addToData = False
                elif inst[jpos][1] is not None and inst[jpos][0] is None:
                    #print("here2")
                    #print(str(type(j)))
                    if str(type(j)) == "<class 'int'>" or str(type(j)) == "<class
'decimal.Decimal'>":
                        if j > int(inst[jpos][1]):
                            addToData = False
                    elif str(type(j)) == "<class 'float'>":
                        if j > float(inst[jpos][1]):
                            addToData = False
                    elif str(type(j)) == "<class 'datetime.date'>":
                        dt1 = datetime.datetime.strptime(inst[jpos][1], '%Y-%m-%d')
                        if j > dt1.date():
                            addToData = False
                elif inst[jpos][1] is None and inst[jpos][0] is not None:
                    #print("here3 ", str(type(j)))
                    if str(type(j)) == "<class 'int'>" or str(type(j)) == "<class
'decimal.Decimal'>":
                        if j < int(inst[jpos][0]):
```

```
                        addToData = False
                    elif str(type(j)) == "<class 'float'>":
                        if j < float(inst[jpos][0]):
                            addToData = False
                    elif str(type(j)) == "<class 'datetime.date'>":
                        dt1 = datetime.datetime.strptime(inst[jpos][0], '%Y-%m-%d')
                        if j < dt1.date():
                            addToData = False
                    elif str(j) != str(inst[i.index(j)][0]):
                        #print("here5")
                        addToData = False
                elif inst[jpos][1] is None and inst[jpos][0] is None:
                    #print('here5')
                    jpos += 1
                    continue
                else:
                    #print("here4")
                    addToData = False
                jpos += 1
            if addToData:
                data.append(i)

        else:
            #print("From 4")
            for i in onedata:
                addToData = False
                jpos = 0
                for j in i:
                    if inst[jpos][1] is not None and inst[jpos][0] is not None:
                        #print("here1")
                        if str(type(j)) == "<class 'int'>" or str(type(j)) == "<class
'decimal.Decimal'>":
                            if j >= int(inst[jpos][0]) and j <= int(inst[jpos][1]):
                                addToData = True
                        elif str(type(j)) == "<class 'float'>":
                            if j >= float(inst[jpos][0]) and j <= float(inst[jpos][1]):
                                addToData = True
                        elif str(type(j)) == "<class 'datetime.date'>":
                            dt1 = datetime.datetime.strptime(inst[jpos][0], '%Y-%m-%d')
                            dt2 = datetime.datetime.strptime(inst[jpos][1], '%Y-%m-%d')
                            if j >= dt1.date() and j <= dt2.date():
                                addToData = True
                        elif str(j) == str(inst[jpos][0]):
                            addToData = True
```

```
                elif inst[jpos][1] is not None and inst[jpos][0] is None:
                    #print("here2")
                    if str(type(j)) == "<class 'int'>" or str(type(j)) == "<class
'decimal.Decimal'>":
                        if j <= int(inst[jpos][1]):
                            addToData = True
                    elif str(type(j)) == "<class 'float'>":
                        if j <= float(inst[jpos][1]):
                            addToData = True
                    elif str(type(j)) == "<class 'datetime.date'>":
                        dt1 = datetime.datetime.strptime(inst[jpos][1], '%Y-%m-%d')
                        if j <= dt1.date():
                            addToData = True
                        # ????? str
                elif inst[jpos][1] is None and inst[jpos][0] is not None:
                    #print("here3 ", str(inst[jpos][0]))
                    if str(type(j)) == "<class 'int'>" or str(type(j)) == "<class
'decimal.Decimal'>":
                        if j >= int(inst[jpos][0]):
                            addToData = True
                    elif str(type(j)) == "<class 'float'>":
                        if j >= float(inst[jpos][0]):
                            addToData = True
                    elif str(type(j)) == "<class 'datetime.date'>":
                        dt1 = datetime.datetime.strptime(inst[jpos][0], '%Y-%m-%d')
                        if j >= dt1.date():
                            addToData = True
                    elif str(j) == str(inst[i.index(j)][0]):
                        #print("here5")
                        addToData = True
                elif inst[jpos][1] is None and inst[jpos][0] is None:
                    #print('here5')
                    jpos += 1
                    continue
                else:
                    #print("here4")
                    addToData = True
                jpos += 1
            if addToData:
                data.append(i)
    return data
```