Форма № 24

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)
бакалавра
(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СЕРВІСУ ЕКСПРЕС-ДОСТАВКИ
EXPRESS DELIVERY SERVICE SOFTWARE

Виконав: студент(ка) 4 курсу, групи КНТ-117
Спеціальності 121 Інженерія програмного забезпечення
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

Кодак І.В.
(прізвище та ініціали)

Керівник Льовкін В.М.
(прізвище та ініціали)

Рецензент Зеленьова І.Я.
(прізвище та ініціали)

2021

Форма № 25

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ІІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації)

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
_____С.О. Субботін
“____” _____2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Кодака Івана Васильовича
(прізвище, ім’я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення сервісу експрес-доставки Express Delivery Service Software

керівник проєкту (роботи) Льовкін Валерій Миколайович, к.т.н., доцент ,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від “ 30 ” березня 2021 року № 103
2. Строк подання студентом проєкту (роботи) травень 2021 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Проєктування програмного забезпечення. 3. Основні рішення щодо реалізації компонентів системи. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Льовкін В.М., доцент		
Нормоконтролер	Камінська Ж.К., асистент		

7. Дата видачі завдання “ 10 ” березня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2-3 тижні	Розділ 1
3	Розробка архітектури програми	4 тиждень	Розділ 2
4	Розробка програми	5-6 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми	7 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	8 тиждень	Додатки
7	Захист роботи	9 тиждень	

Студент(ка) _____ Кодак І.В.
(підпис) (прізвище та ініціали)

Керівник проєкту (роботи) _____ Льовкін В.М.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 87 с., 25 рис., 1 табл., 3 дод., 14 джерел.

СЛУЖБИ ДОСТАВКИ, ЕКСПРЕС-ДОСТАВКА, КУР'ЄРСЬКА ДОСТАВКА, ВІДСЛІДКОВУВАННЯ ПАКУНКІВ, MODEL-VIEW-VIEWMODEL, MICROSOFT SQL SERVER.

Об'єкт роботи – процес експрес-доставки. Предмет роботи – програмне забезпечення сервісу експрес-доставки.

Мета роботи – розроблення програмного забезпечення сервісу експрес-доставки для забезпечення моніторингу даних про стан доставки зі сторони всіх учасників процесу та підтримки реалізації даного процесу.

Матеріали, методи та технічні засоби: мова програмування C#, патерн проєктування Model-View-ViewModel, система Microsoft SQL Server.

Результати. Виконано огляд проблеми експрес-доставки, проєктування програмного забезпечення зі створенням діаграми прецедентів, схеми база даних, структурної схеми і обґрунтуванням проєктних рішень. Реалізовано програмний застосунок і представлено прийняті під час цього рішення.

Висновки. Створене програмне забезпечення надає можливість керування процесом екпрес-доставки шляхом відслідковування стану доставки пакунку, перегляду всіх власних відправлених пакунків і пакунків для отримання, визначення очікуваної дати прибуття пакунку, підтвердження доставки кур'єром, видачі пакунку, визначення прибуття пакунків, відправлення пакунку, внесення попередніх даних відправлення, перегляду пакунків за відділенням, розподілу пакунків за кур'єрами.

Галузь використання. Сервіси експрес-доставки з потенційною мережею відділень у різних містах країни та кур'єрською службою.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 87 pages, 25 figures, 1 table, 3 appendixes, 14 sources.

DELIVERY SERVICE, EXPRESS DELIVERY, COURIER DELIVERY, PACKAGE TRACKING, MODEL-VIEW-VIEWMODEL, MICROSOFT SQL SERVER.

The object of the work is express delivery process. The subject of the work is express delivery service software.

The purpose of the work is to develop express delivery service software to ensure the monitoring of data on the state of delivery by all participants of the process and to support the implementation of this process.

Materials, methods and technical means: C# programming language, Model-View-ViewModel design pattern, Microsoft SQL Server system.

Results. An overview of the problem of express delivery was performed. Software design was presented using use case diagram, database schema, functional diagram and justification of design decisions. The software application is implemented and the decisions made during this process are presented.

Conclusions. The created software allows users to control the express delivery process by tracking the delivery status of the package, viewing all packages sent by user and to be received by user, determining the expected date of arrival of the package, confirming delivery by courier, issuing the package, determining the arrival of packages, sending packages, making preliminary sending data, viewing packages by department, making distribution of packages by couriers.

Field of use. Express delivery services with a potential department network in different cities of the country and courier service.

ЗМІСТ

С.

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

MVVM – Movel-View-VIewModel;

БД – база даних.

ВСТУП

Зростання роздрібних продажів в Інтернет трансформувалося і вплинуло на логістичні фірми протягом останніх десяти років, і ця тенденція продовжуватиметься на тому ж або подібному рівні протягом наступних декількох років, відповідно оскільки зростає кількість товарів, що їх купують компанії та споживачі в Інтернет, зростає попит на послуги експрес-доставки [1]. До даних тенденцій додалася також ситуація з пандемією, адже в період локдаунів сервіси експрес-доставки стали виконувати додатковий обсяг роботи, пов’язаний зі створенням каналу доставки товарів до споживачів з магазинів. Стаючи на певний період по суті єдиним каналом фізичної комунікації між не тільки інтернет-магазинами, але і звичайними магазинами і споживачами, сервіси експрес-доставки отримали додаткові можливості для розвитку.

Таким чином, не зважаючи на популярність даного сектору в світі, значні обсяги роботи і активний розвиток, можна говорити про актуальність розширення сервісу і на даний момент. Відповідно даний сектор потребує якісного програмного забезпечення для підтримки роботи, адже без існування такого програмного забезпечення досягти якісної реалізації послуг неможливо.

Під час розробки програмного забезпечення необхідно враховувати те, що потенційно таке програмне забезпечення буде розширюватися, адже подібні сервіси весь час вимагають збільшення кількості послуг, функціональних особливостей програмного забезпечення через високу конкурентність у цій сфері, а також потребу підтримки потенційно більшого охоплення ринку компанією. Сервіс експрес-доставки – це відносно велика або велика компанія, яка має представництва в різних містах, має логістичні центри і дана конструкція не є статичною, а розростається з часом.

Метою роботи є розроблення програмного забезпечення сервісу експрес-доставки для забезпечення моніторингу даних про стан доставки зі сторони всіх учасників процесу та підтримки реалізації даного процесу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка проблеми організації експрес-доставки

Суть експрес-доставки заснована на організації логістичного процесу таким чином, що відправлення отримується з будь-якого місця відвантаження та доставляється до будь-якого пункту призначення, на практиці це означає, що ні відправник, ні одержувач не повинні переїжджати зі свого постійного місця проживання для обміну товарами, в даний час спектр додаткових послуг розвивається надзвичайно широко, так відправнику слід очікувати:

– доставку конкретній особі;

– доставку до зазначеного часу, зазвичай близько 9:00 або 10:00 та 12:00 ранку (на даний момент найпопулярніша додаткова послуга);

– повернення підтвердженого документа, що додається до вантажу;

– страхування вмісту вантажу;

– підтвердження доставки електронною поштою, телефоном або текстовим повідомленням;

– копії списку з підписом одержувача [2].

Швидкість зростання популярності електронної комерції очевидна зокрема і завдяки швидкості збільшення кількості транспортних засобів у житлових районах, важливими тенденціями на даному ринку з розвитком сервісів є збільшення кількості невдалих спроб доставки, оскільки доставки додому стають все частішими: для того, щоб доставити посилку (пакунок) до замовника, кур'єрським компаніям потрібен підпис власника перед завершенням фізичної доставки, і більшу частину часу споживачі не завжди бувають вдома, відповідно це призводить до завищених витрат на доставку через необхідність повертати вантаж відправнику, а також можливо доставляти посилку повторно [1]. Окрім того останнім часом в Україні збільшується кількість шахрайських дій, пов’язаних з сервісами доставки, які також впливають на даний процес.

Основна мета компаній, що здійснюють доставку посилок, полягає в тому, щоб посилки, які відправляються для доставки клієнту їх кур’єром (водієм), доставлялися для досягнення задоволення клієнта, для того щоб компанії-постачальники посилок залишалися конкурентоспроможними у сфері електронної комерції, вони повинні мати можливість забезпечити високий рівень обслуговування та підтримку економічної ефективності, як результат, усі постачальники послуг у експрес-компаніях, що здійснюють доставку, зосереджуються на мінімізації повернення (повторного відправлення) та задоволенні нових вимог споживачів, які роблять конкуренцію жорсткішою [1].

На відміну від інших логістичних служб, служби експрес-доставки більше зосереджуються на швидкості, безпеці та точності виконання операцій, для досягнення цих цілей весь логістичний процес повинен містити наземну, морську та повітряну мережі, це означає, що необхідна співпраця з іншими компаніями та іншими країнами, відповідно поширюючи спектр послуг, компанії експрес-доставки повинні розвивати більше видів бізнесу, тому щоб заощадити час замовника, послуга може починатися не обов’язково з відділення, але і від дверей замовника, ці послуги збільшують ширину та глибину експрес-послуг, але також пред'являють високі вимоги до логістичної системи, тому система логістики у службах експрес-доставки має такі особливості:

– кожна посилка має свій процес і різну швидкість, отже є декілька вузлів для сортування цих різних посилок у процесі логістики;

– через різні вимоги ряду клієнтів і до різних товарів система повинна обробляти ці справи з великим обсягом інформації;

– географічне розташування розподільчих центрів, поштових відділень та складів не зосереджене;

– розширення логістичної служби в нових місцях завжди потрібно;

– місцезнаходження кожного філіалу знаходиться далеко від інших, що іноді призводить до труднощів в управлінні;

– щоб скоротити витрати, компанії експрес-доставки не використовують прямий транспорт від відправника до одержувача, вони управляють деяким логістичним центром, в якому зосереджені посилки багатьох клієнтів;

– послуги від дверей до дверей потребують більше персоналу та збільшують навантаження [3].

Швидкість експрес-доставки безпосередньо впливає на задоволення замовників від придбання товарів у інтернет-магазинах, адже чим швидше буде виконуватися доставка, тим конкурентнішим буде ставати у всіх відношеннях інтернет-магазин зі звичайним магазином. Це обумовлено тим, що окрім зручності вибору всього в одному місці (через програмне забезпечення) нівелюється ще фактор часу. Наприклад, якщо доставка буде виконувати в той же день, то потенційно це може призвести до того, що люди будуть використовувати інтернет-магазини вже і для швидкого забезпечення власних потреб.

Основними елементами створення якісного сервісу експрес-доставки як будь-якого сервісу є:

– культура обслуговування побудована на елементах принципів лідерства, норм, трудових звичок і бачення, місії та цінностей, під культурою при цьому розуміється сукупність найважливіших принципів, згідно з якими керівництво контролює, підтримує та розвиває процес, представлений як доставлення послуг та передача цінностей споживачам, після того, як була створена вища система надання послуг та реалістична концепція послуг, немає жодної іншої складової, що є такою фундаментальною для довгострокового успіху організації обслуговування, як її культура;

– залучення співробітників включає діяльність, пов’язану зі ставленням співробітників, лідерство, кероване цілями, та процеси управління персоналом: навіть найкраще розроблені процеси та системи будуть ефективними лише в тому випадку, якщо їх здійснюватимуть люди з

високою залученістю в робочі процеси, залучення є модератором між розробкою та виконанням моделі досконалості послуг;

– якість послуг включає стратегії, процеси та системи управління ефективністю: розробка стратегії та процесу є фундаментальною для проєктування загальної моделі управління послугами, допомога клієнту у виконанні його місії та підтримка у досягненні його організаційних цілей повинні бути основою будь-якого партнерства, що надає послуги;

– досвід клієнтів включає елементи інформації від клієнтів, управління рахунками та постійні вдосконалення: сприйняття є головним і потрібно постійно оцінювати, як сприймається надання послуг як споживачем, так і кінцевим користувачем, що важливо для постійної співпраці, успішне надання реалізується працює на основі того, що клієнт є частиною створення та надання послуги, а потім розробляються процеси, побудовані на цій філософії [4].

Зараз світові інтернет-магазини часто створюють власні логістичні послуги, крім того, підприємства, що працюють у спектрі спільної економіки, вивчають можливості логістики, існують наступні приклади відповідних бізнес-моделей:

– Amazon розширює власні логістичні послуги, щоб стати менш залежними від традиційних вантажних послуг;

– Alibaba вирішила інвестувати в ряд невеликих логістичних компаній з метою створення логістичної мережі;

– Uber (та інші) розробляють «модель перевезення вантажів», яка пропонує приватним громадянам можливість заробляти гроші доставкою посилок, при цьому основна критика описаної бізнес-моделі полягає в тому, що вона потенційно стимулюють сутність «одного пакета, одного автомобіля», нехтуючи тим, щоб використовувати всю місткість транспортного засобу, це не особливо екологічно, дослідження показують, що послуги, подібні до Uber, також спонукають людей їздити в пошуках

роботи, що призводить до збільшення кількості автомобілів на дорозі та додаткового забруднення.

– 3M Europe вдалося зменшити викиди на 50% і витрати на 35%, переключившись на логістичні мережі MIXMOVE, навіть у міських районах з міським трафіком це рішення дало подібні цифри [5].

1.2 Огляд програмних аналогів

Типовий застосунок для експрес-доставки пропонує зручний спосіб відстежувати пакунки в режимі реального часу: з моменту їх реєстрації в системі (перевізником або користувачем) і до моменту, коли вони досягають місця призначення та доставляються за адресою, не зважаючи на таку типовість, існують певні відмінності між, наприклад, застосунком доставки посилок FedEx або DPD та послугою доставки пакетів на замовлення, як Uber, в основному ця різноманітність обумовлюється як раз різними бізнес-моделями, які можна застосувати до відстеження та доставки пакунків, але загалом в програмному забезпеченні, що використовуються для даної основної функціональності, виділяють наступні групи:

– агрегатори відстеження пакунків – такі програми відстеження посилок об’єднують дані з усіх сервісів у одне місце, таким чином, потрібен лише один застосунок для відстеження пакунків, незалежно від оператора, це дуже просто, можна додати пакунок вручну, використовуючи його ідентифікатор, і програма буде повідомляти про стан доставки за допомогою своєчасних повідомлень, але при цьому варто враховувати, що такі програми надають функціональність тільки відстеження пакунку, тобто не можна його створити в системі для відповідної служби, не можна реалізувати ніяких операційних дій, тобто агрегатор корисний саме для питання відстеження пакунків, але програмне забезпечення кожного окремого сервісу може додавати до даної функціональності ще додаткові засоби;

– програми поштової та кур'єрської служби: всі сучасні служби еспрес-доставки повинні мати відповідні застосунки, що можуть охоплювати вебзастосунки, мобільні застосунки, десктопні застосунки, серед прикладів можна привести FedEx (який також має окремий застосунок для локальної доставки в той же день) та програму доставки посилок DPD, що пропонує повноцінну адміністративну панель клієнта, включаючи можливість створювати відправлення та навіть роздруковувати етикетки для своїх пакунків;

– фірмові програми для доставки та інтегровані рішення: хоча деякі інтернет-магазини можуть додавати можливості відстеження замовлень у свої мобільні застосунки або вебсистеми, мало хто з них насправді надаватиме повний обсяг інформації про доставку, у більшості випадків вони знають лише, обробляється чи відправляється замовлення і коли воно буде доставлене, саме тому деякі роздрібні продавці використовують фірмові програми доставки пакунків як конкурентну перевагу, забезпечуючи тим самим кращу прозорість та зручність для своїх клієнтів, у той же час для маленького місцевого продавця немає сенсу створювати власний мобільний застосунок для доставки, тому в такому випадку корисними можуть бути сторонні рішення, такі як Doorman або Deliv, вони інтегруються з бізнесом, включаючи вебсайт, застосунок чи систему управління запасами, і надають можливості відстеження посилок;

– Uber-подібні засоби для доставки пакунків: існують десятки подібних до Uber застосунків та підприємств, які порушують традиційні бізнес-моделі у багатьох галузях, деякі стартапи зосереджуються на їжі (наприклад, GrubHub), продуктах (Instacart), інші мають на меті трансформувати традиційні кур'єрські послуги (Quiqup), Roadie застосовує дещо інший підхід, він поєднує в собі поїздки та послуги доставки на вимогу, пов’язуючи людей, яким потрібно щось доставити, з людьми, які можуть це зробити за них, і роблять це під час подорожі або просто як допомогу [6].

В якості основної моделі для розробки програмного забезпечення сервісу експрес-доставки використано модель програми поштової та кур'єрської служби.

Серед програмних засобів експрес-доставки першим виділемо застосунок FedEx [7] (рис. 1.1).

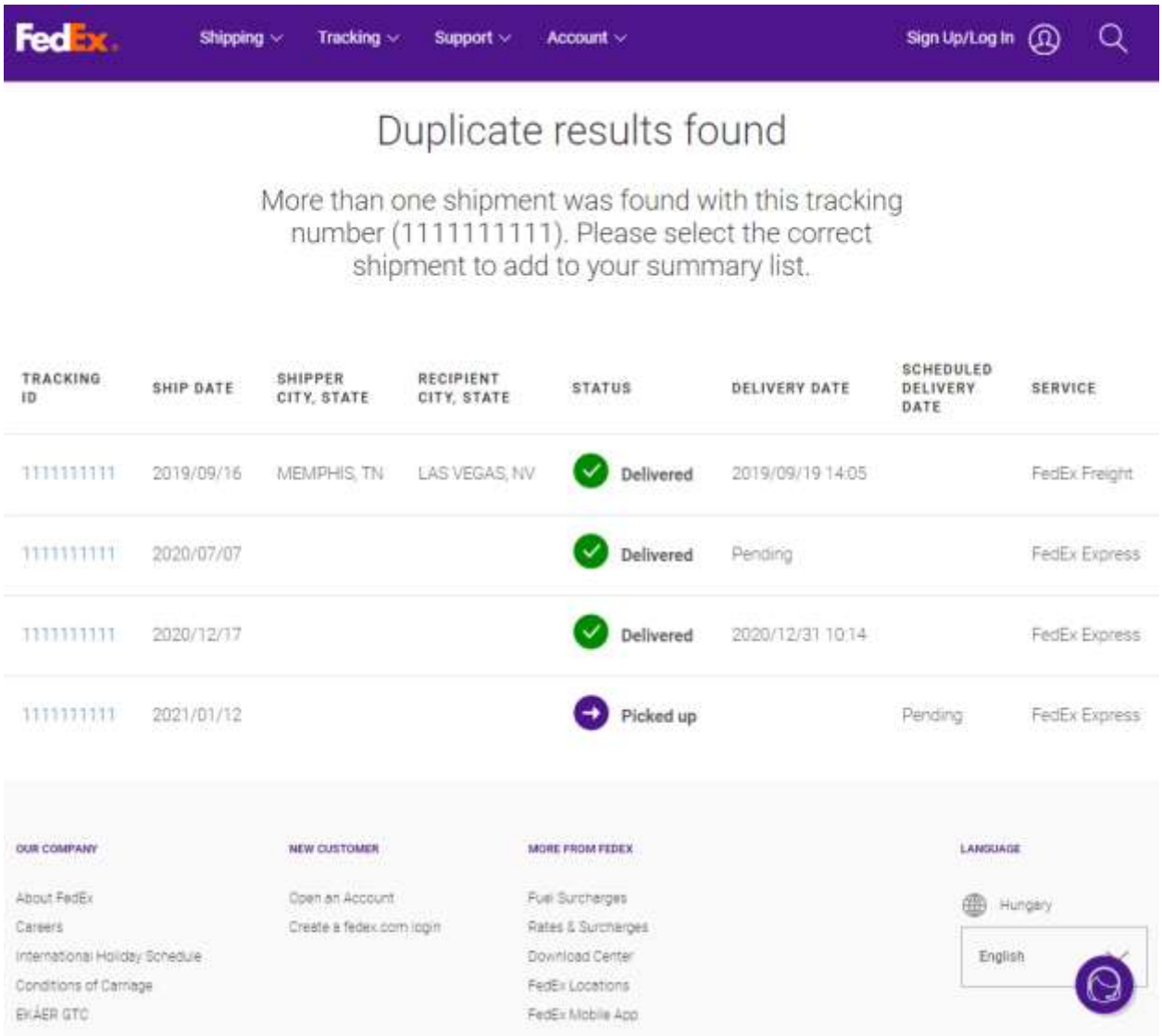

Рисунок 1.1 – Застосунок FedEx

Застосунок FedEx характеризується наданням наступних функціональних можливостей:

– відстеження стану усіх вхідних або вихідних пакунків;

– створення етикетки мобільного відвантаження та подача запиту на відправлення;

– увімкнення push-сповіщення, щоб отримувати оновлення щодо відправлень;

– сканування штрих-коду для зручності відстеження;

– пошук поблизу місцезнаходження FedEx;

– отримання кошторису витрат на доставку та термінів доставки;

– отримання доступу до адресної книги вантажовідправника;

– пошук відправлень за номером відстеження, посиланнями, постачальниками тощо [8].

В якості додаткової альтернативи розглянемо програмний застосунок компанії DHL [9] (рис. 1.2).

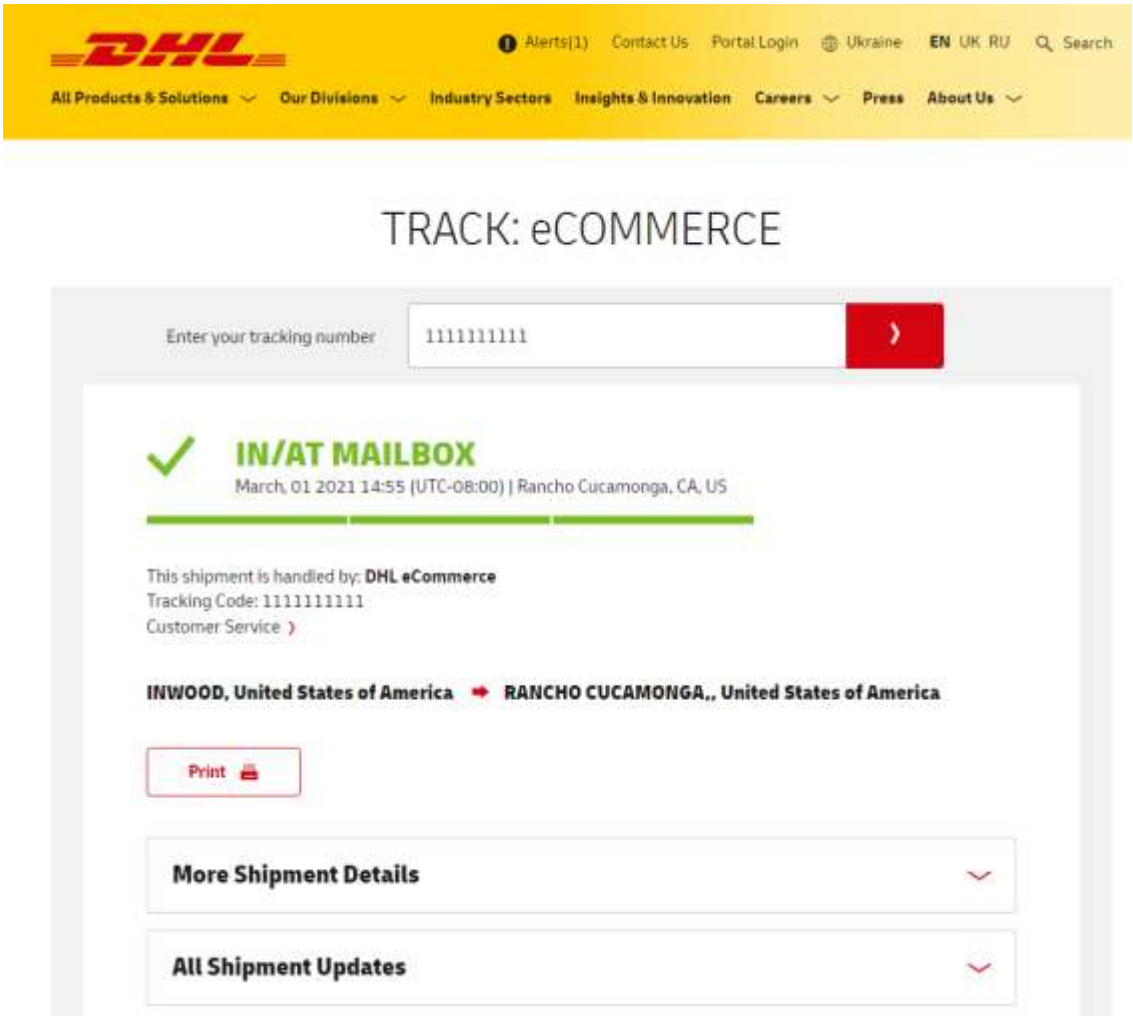

Рисунок 1.2 – Застосунок DHL

Застосунок DHL надає можливість виконувати зокрема наступні функції:

– пошук вантажів, що відображає не лише поточний стан відвантаження, а також дозволяє клієнтам відстежувати додаткові деталі процесу відвантаження, такі як час початку відправлення, поточне місцезнаходження та передбачуваний час прибуття;

– збереження пошукових запитів;

– перегляд детальної історії доставки за останні шість місяців: індивідуальний звіт за останні півроку допомагає користувачам перевіряти економічну ефективність своїх відправлень;

– пошук місця, який показує найближчий офіс DHL Global Forwarding та його контактні дані [10].

Базовою функцією обох застосунків є відслідковування етапів доставки.

1.3 Висновки за розділом 1

Проаналізовано проблему експрес-доставки:

– розглянуто основні можливості сервісів експрес-доставки;

– розглянуто особливості експрес-доставки порівняно зі звичайною доставкою пакунків, що включають швидкість, безпеку та точність виконання операцій;

– представлено моделі організації експрес-доставки та програмного забезпечення з даної сфери;

– обрано модель програм поштової та кур'єрської служби в якості основи для розробки програмного забезпечення в роботі.

У даній дипломній кваліфікаційній роботі має бути розв’язано наступні завдання:

– визначення особливостей проблеми експрес-доставки та використовуваного програмного забезпечення;

– визначення вимог до програмного забезпечення сервісу експрес-доставки;

– вибір мови і засобів проєктування і розробки;

– проєктування програмного забезпечення;

– реалізація і налагодження програмного забезпечення.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Вибір мови програмування

Програмне забезпечення сервісу експрес-доставки потребує загалом розроблення певної екосистеми, яка може включати і десктопне рішення, зручне для роботи працівників служби, і вебрішення, зручне для універсальної роботи, і мобільне рішення для різних операційних систем, зручне для мобільного доступу. У межах даної роботи було прийнято рішення розробити саме десктопне рішення. Для реалізації програми було порівняно між собою мови програмування C# [11]-[12] та C++ [13]-[14] (табл. 2.1), в результаті за сукупністю переваг обрано мову C#.

Таблиця 2.1 – Вибір мови програмування

Критерій	C#	C++
Парадигма програмування	Об’єктно-орієнтована, мультипарадигмальна	Об’єктно-орієнтована, мультипарадигмальна
Створення настіль-них застосунків	Так	Так
Абстракція моделей	Вищий рівень	Високий рівень
Легкість написання коду	Вимагає менше часу	Вимагає більше часу

2.2 Вимоги до програмного забезпечення

Для підтримки роботи сервісу експрес-доставки програмне забезпечення повинно виконувати функції, необхідні для роботи як клієнтів, при чому як відправників, так і отримувачів, так і операторів відділень служби, менеджерів служби, а також кур’єрів служби.

Відслідковування стану доставки пакунку за номером є базовим засобом отримання інформації про процес доставки конкретної посилки для

будь-якого користувача.

Перегляд всіх власних відправлених пакунків є акумулюючим засобом роботи з власними відправленнями, оскільки в такому випадку для перевірки стану не потрібно вводити номер, тобто акумуляція відбувається за авторизованим обліковим записом. Окрім того звідси доступ має бути організований як до попередньо внесених даних відправлень, тобто коли сформоване замовлення, але пакунок ще не відправлено, так і до відправлень, які знаходяться в процесі доставки, так і до історії виконаних відправлень, тобто в процесі об’єднуються одразу три складові.

Перегляд всіх пакунків для отримання передбачає той же самий підхід, але вже до отримань, а не відправлень. За рахунок цих двох функцій без необхідності використання окремих облікових записів користувач може отримувати доступ до засобів роботи як відправника, так і отримувача.

Перегляд очікуваної дати прибуття пакунку має надаватися під час перегляду деталізованої інформації про посилку.

Підтвердження доставки кур’єром є основною функцію комунікації клієнта та кур’єра під час отримання клієнтом пакунку. Задля уникнення підписування документів під час отримування пакунків клієнт має скористатися відповідною функцією в програмі і підтвердити, що він дійсно фактично отримав пакунок.

Видача пакунку призначена для виконання завершення роботи з клієнтом зі сторони кур’єра або оператора відділення. У даному випадку кур’єр або оператор вносить результат взаємодії, що полягає в успішному виконанні та сплаті потрібної суми коштів або відмовлення від пакунку.

Прибуття пакунків – це функція, що дозволяє виходячи з номеру відділення та початкового пункту відправлення пакунків (сортувального терміналу) визначити факт прибуття посилок з переліку.

Відправлення пакунку передбачає внесення в систему інформації про фактичне отримання пакунку представником сервісу експрес-доставки, тобто кур’єром або оператором.

Внесення попередніх даних відправлення передбачає підготовку даних для виконання попередньої функції, коли клієнт-відправник вносить інформацію про відправлення, а потім кур’єр або оператор відповідно підтверджує або змінює внесені дані.

Пошук пакунку складовою перегляду відправлень різними користувачами і виступає певним фільтром під час перегляду всіх даних.

Перегляд пакунків за відділенням дозволяє оператору або менеджеру переглядати всі пакунки, які в даний момент знаходяться у відділенні. Таким чином зокрема реалізується повернення пакунків у випадку, якщо отримувач їх не забрав.

Визначення відділень доставки за параметрами пакунку є частиною внесення попередніх даних відправлення та відправлення пакунку. Дана функція потрібна, оскільки в залежності від параметрів пакунку (розмірів, ваги) не всякий пакунок можна доставити на будь-яке відділення. Тому перелік доступних відділень має формуватися на основі автоматичного визначення.

Реєстрація облікового запису має реалізовуватися тільки для клієнта (відправника або отримувача), оскільки тільки така роль не потребує певної верифікації. Це пов’язано з тим, що після заповнення даних відправлення відправником вони потім підтверджуються оператором або кур’єром, тобто здійснюється додаткова перевірка. Роль отримувача посилки в свою чергу передбачає тільки доступ до даних на читання. При цьому номер телефону є ключовим фактором додаткової перевірки. Тобто якщо користувач фактично знає номер відправлення і номер телефону отримувача, то він може отримати доступ до ширшої інформації про пакунок.

Авторизація в системі виконується для клієнта, менеджера, оператора, кур’єра.

Сформульований набір функціональних вимог до програмного забезпечення дозволив створити діаграму прецедентів (рис. 2.1).

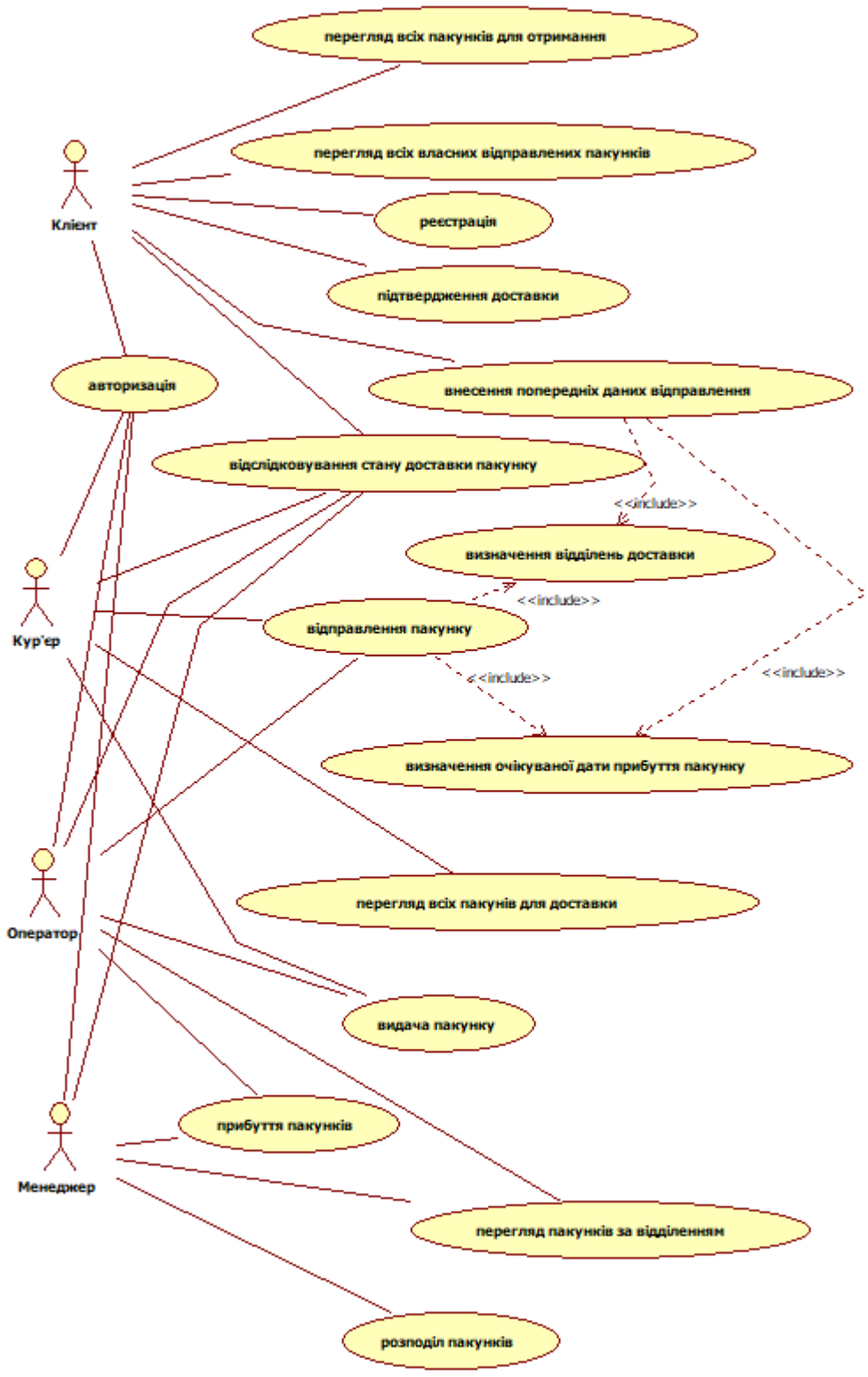

Рисунок 2.1 – Діаграма прецедентів

Перегляд кур’єром всіх пакунків для доставки передбачає отримання в зручному вигляді доступу до всіх пакунків, з яким кур’єр має працювати

протягом дня, тоді з цього вікна кур’єр буде мати швидкий доступ до внесення даних про всі пакунки.

Розподіл пакунків за кур’єрами передбачає роботу менеджера, який визначає, які саме пакунки будуть доставлені яким саме кур’єром.

2.3 Проєктування бази даних

Як було зазначено вище в даному розділі, робота з БД є одним з ключових питань організації застосунку, що розробляється.

Основними сутностями цієї предметної області є:

– Account – обліковий запис;

– Client – клієнт;

– Worker – працівник сервісу експрес-доставки;

– Package – пакунок для доставки;

– Department – відділення сервісу;

– Route – шляхи транспортування;

– Center – найближчий логістичний центр;

– Tariff – вартість доставки.

Сутність Account визначена характеристиками:

– ID – номер;

– Login – логін;

– Password – пароль;

– Status – активний чи неактивний.

Сутність Client визначена характеристиками:

– ID – номер;

– AccountID – номер облікового запису користувача (зовнішній ключ до таблиці Account, який може бути незаповнений, якщо клієнт не має запису в системі);

– Surname – прізвище;

– Name – ім’я;

– MiddleName – по батькові;

– Phone – телефон;

– Address – адреса;

– NearestDepartmentID – найближче відділення (зовнішній ключ до таблиці Department, що може бути незаповнений).

Сутність Package визначена характеристиками:

– ID – номер;

– Track – номер для відслідковування;

– PType – тип пакунку (вантаж, документи, посилка);

– DType – тип доставки (відділення-відділення, дім-відділення, відділення-дім, дім-дім);

– Weight – вага;

– Size1 – перший розмір;

– Size2 – другий розмір;

– Size3 – третій розмір;

– Status – стан;

– Location – поточне розташування;

– Client1ID – номер клієнта-відправника (зовнішній ключ до таблиці Client);

– Date1 – дата і час відправлення пакунку;

– Address1 – адреса відправлення пакунку (порожня для відправлення з відділення);

– Department1ID – відділення відправлення пакунку (відділення, через яке буде організовано доставку кур’єром, або відділення, безпосередньо з якого відправлено пакунок), окрім того це зовнішній ключ до таблиці Department;

– Client2ID – номер клієнта-отримувача (зовнішній ключ до таблиці Client);

– Date2 – дата і час отримання пакунку (очікуваний або фактичний);

– Address2 – адреса отримання пакунку (порожня для отримання відправлення у відділенні);

– Department2ID – відділення отримання пакунку (відділення, через яке буде організовано доставку кур’єром, або відділення, безпосередньо в яке відправлено пакунок), окрім того це зовнішній ключ до таблиці Department;

– Price – вартість;

– Paid – сплачена вартість;

– WorkerID – номер працівника, тобто кур’єра, який відповідає за пакунок (зовнішній ключ до таблиці Worker);

– TariffID – номер тарифу (зовнішній ключ до таблиці Tariff).

Сутність Worker визначена характеристиками:

– ID – номер;

– AccountID – обліковий запис (зовнішній ключ таблиці Account);

– Surname – прізвище;

– Name – ім’я;

– MiddleName – по батькові;

– Phone – телефон;

– StartDate – дата початку роботи;

– DepartmentID – відділення, до якого прив’язаний працівник (зовнішній ключ до таблиці Department);

– Role – роль (оператор, менеджер або кур’єр).

Сутність Department визначена характеристиками:

– ID – номер;

– City – місто;

– Number – номер відділення у місті;

– Address – адреса;

– WType – тип за вагою (максимальна допустима вага);

– DType – тип відділення (логістичний центр, стандартне відділення, невелике відділення).

На рис. 2.2 представлена схема БД.

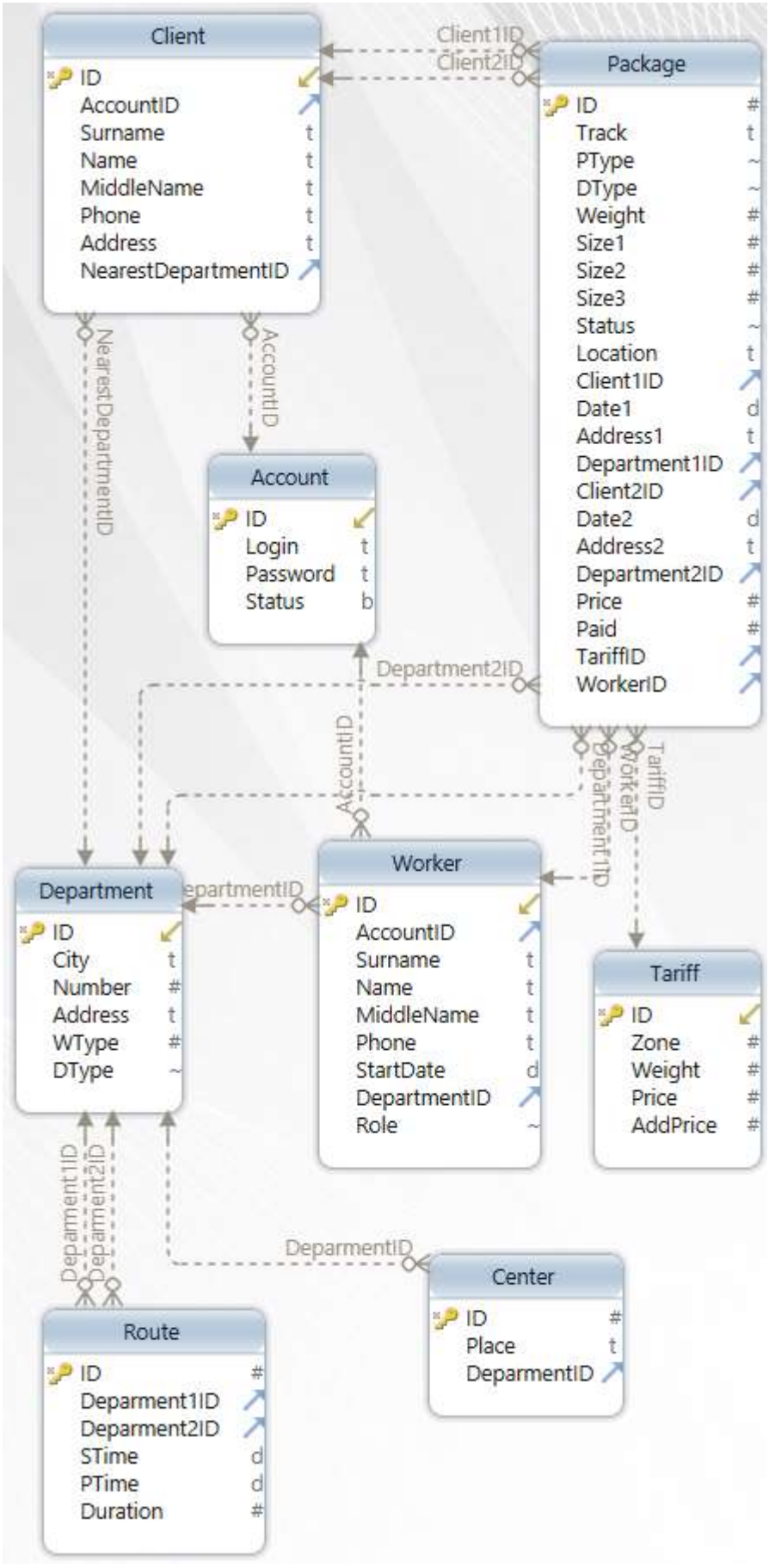

Рисунок 2.2 – Схема БД

Сутність Route визначена характеристиками:

– ID – номер;

– Department1ID – місце відправлення (зовнішній ключ до таблиці Department);

– Department2ID – місце прибуття (зовнішній ключ до таблиці Department);

– STime – час відправлення;

– PTime – час в шляху;

– Duration – відстань.

Сутність Center визначена характеристиками:

– ID – номер;

– Place – місце, для якого дане відділення є центром;

– DeparmentID – номер відділення, що є центром (зовнішній ключ до таблиці Department).

Сутність Tariff визначена характеристиками:

– ID – номер;

– Zone – номер тарифної зони;

– Weight – допустима максимальна вага;

– Price – вартість доставки;

– AddPrice – вартість кур’єрських послуг.

2.4 Визначення архітектури програмного забезпечення

Запроєктовано програмне забезпечення на основі патерну Movel-View-ViewModel (MVVM), що порівняно з проєктуванням і розробкою на основі предметної області спрощує відповідні процеси проєктування і розробки.

Патерн MVVM (рис. 2.3) передбачає розбиття системи на моделі, що містять дані, отримані з БД, і бізнес-логіку, моделі представлень, що містять

презентаційну логіку, та представлення, що визначає користувацький інтерфейс.

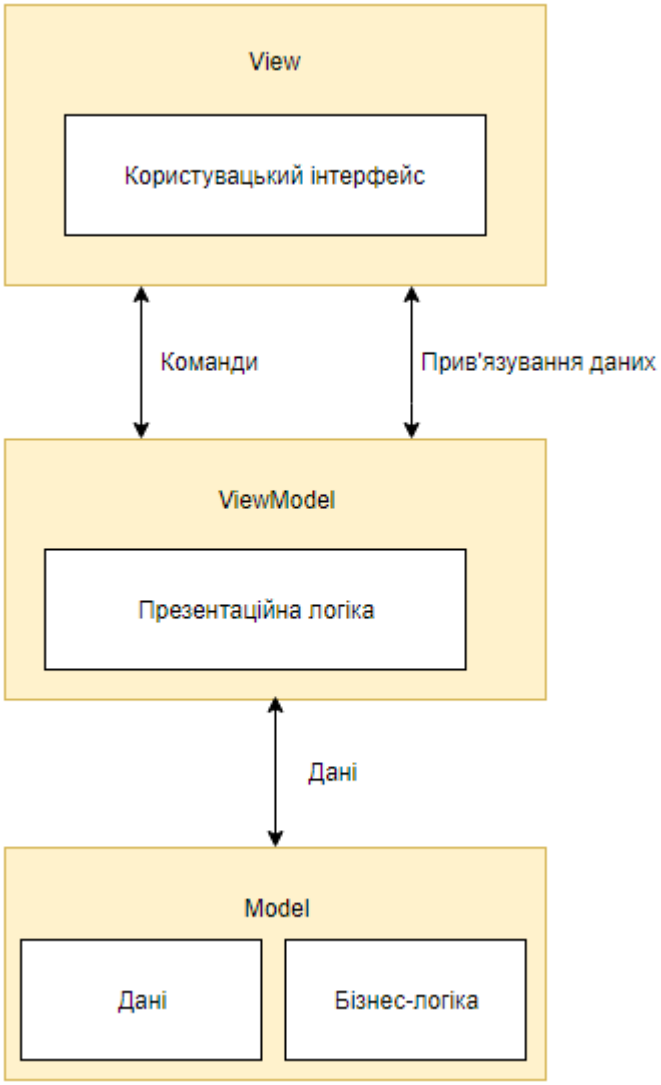

Рисунок 2.3 – Патерн MVVM

2.5 Висновки за розділом 2

За результатами порівняння мов програмування C# та C++ та визначеними для цього критеріями було обрано мову програмування C# для реалізації програмного забезпечення.

Визначено функціональні вимоги до розроблюваного програмного забезпечення, на основі цього побудовано діаграму прецедентів для роботи клієнтів, операторів, менеджерів та кур’єрів з програмою.

Запроєктовано БД з визначенням таблиць і їх полів та побудовою схеми БД.

Архітектуру програмного забезпечення побудовано на основі патерну MVVM, що і стало основою подальшого визначення структури програмного забезпечення.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Структура розробленого програмного забезпечення

Для створення моделей було реалізовано такі класи:

– DBTool – базовий клас, який реалізує можливості роботи з БД шляхом виконання відповідних запитів;

– Package – модель пакунку для доставки;

– Account – модель облікового запису;

– Client – модель клієнта;

– Manager – модель менеджера;

– Operator – модель оператора;

– Courier – модель кур’єра;

– Department – модель відділення сервісу;

– Route – модель шляху транспортування;

– Center – модель найближчого логістичного центру;

– Tariff – модель тарифного плану.

Для створення моделей представлень було реалізовано такі класи:

– NewApplicationViewModel – модель представлення внесення попередніх даних про пакунок клієнтом;

– OwnPackagesViewModel – модель представлення перегляду всіх пакунків, призначених для отримання клієнтом;

– SenderPackagesViewModel – модель представлення перегляду всіх пакунків, відправлених клієнтом;

– ClientPackageDetalizationViewModel – модель представлення роботи з окремим пакунком клієнтом;

– OperatorPackageDetalizationViewModel – модель представлення роботи з окремим пакунком оператором;

– CourierPackageDetalizationViewModel – модель представлення роботи з окремим пакунком кур’єром;

– ManagerPackageDetalizationViewModel – модель представлення роботи з окремим пакунком менеджером;

– RegistrationViewModel – модель представлення реєстрації облікового запису клієнта;

– AuthorizationViewModel – модель представлення авторизації облікового запису;

– DepartmentPackagesViewModel – модель представлення перегляду пакунків за відділенням;

– ReadyPackagesViewModel – модель представлення прибуття пакунків на відділення (тобто задається маршрут слідування і за ним надається перелік всіх пакунків, що спрощує роботу з ними, надаючи засоби загальної роботи з пакунками зокрема);

– CourierPackagesViewModel – модель представлення для роботи кур’єра з усіма призначеними для нього поточними пакунками;

– ManagerPackagesViewModel – модель представлення керування менеджером розподілом пакунків між кур’єрами;

– ClientMainViewModel – модель головного представлення роботи клієнта;

– OperatorMainViewModel – модель головного представлення роботи оператора;

– CourierMainViewModel – модель головного представлення роботи кур’єра;

– ManagerMainViewModel – модель головного представлення роботи менеджера.

Для кожної моделі представлення створено відповідне представлення для виведення даних.

У результаті було побудовано структурну схему програмного забезпечення (рис. 3.1).

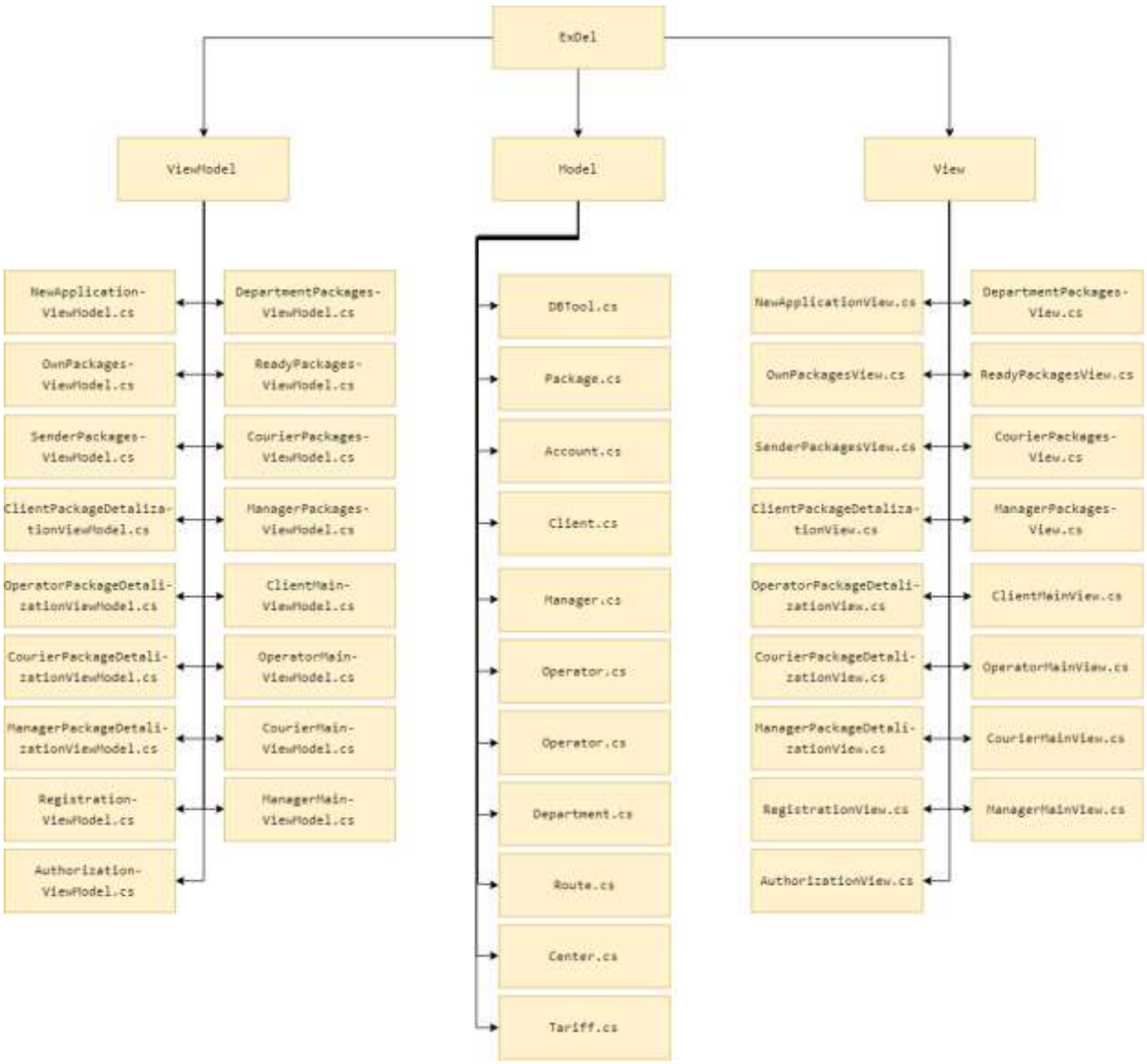

Рисунок 3.1 – Структура розробленого програмного забезпечення

3.2 Опис розроблених класів

Основним класом моделей є клас Package, який визначає основну сутність даної предметної області – пакунок (посилку), доставка якого відбувається в межах сервісу.

Даний клас має зокрема наступні атрибути:

– private int id – номер пакунку;

– private string track – трекінговий номер пакунку, який повідомляється клієнту;

– private int ptype – тип пакунку;

– private int dtype – тип доставки пакунку;

– private float weight – вага пакунку;

– private float size1 – довжина пакунку;

– private float size2 – ширина пакунку;

– private float size3 – висота пакунку;

– private int status – стан доставки пакунку;

– private string location – поточне місце розташування пакунку;

– private Client client1 – відправник;

– private DateTime date1 – дата і час відправлення;

– private string address1 – адреса відправлення (для кур'єрської доставки);

– private Department department1 – відділення відправлення;

– private Client client2 – отримувач;

– private DateTime date2 – дата і час отримання пакунку (прогнозована або фактична);

– private string address2 – адреса отримання (для кур'єрської доставки);

– private Department department2 – відділення отримання;

– private float price – вартість доставки;

– private float paid – сплачена сума за доставку;

– private float calctariff – тариф доставки;

– public string ptypetext – тип пакунку в текстовому представленні;

– public string dtypetext – тип доставки в текстовому представленні;

– public string statustext – стан доставки в текстовому представленні.

Визначення значень останніх трьох текстових атрибутів виконується шляхом вибору значень з відповідних масивів за індексами, що задають відповідне значення в звичайному представленні цих атрибутів.

Атрибут ptypetext має одне з наступних значень:

– не задано;

– вантаж;

– документи;

– посилка.

Атрибут dtypetext має одне з наступних значень:

– не задано;

– відділення-відділення;

– кур’єр-відділення;

– відділення-кур’єр;

– кур’єр-кур’єр.

Атрибут statustext має одне з наступних значень:

– не задано;

– створено;

– забрано кур’єром;

– у відділенні відправлення;

– на шляху до сортувального центру відправлення;

– у сортувальному центрі відправлення;

– у дорозі;

– у сортувальному центрі отримання;

– на шляху до відділення отримання;

– у кур’єра для отримання;

– у відділенні отримання;

– отримано;

– відмова;

– скасовано.

Для кожного атрибуту визначено пару методів get і set, які відповідно зчитують та встановлюють значення атрибутів, наприклад, для ваги побудовано відповідну конструкцію, в якій зокрема визначено прив’язку до властивості Weight, а також властивостей Tariff і Price, оскільки зміна ваги призводить до зміни обчисленої вартості доставки та обраного тарифу, відповідне переобчислення виконується шляхом виклику методу ChangePrice:

```
public float Weight
{
    get { return weight; }
    set
     {
       if (weight != value)
       {
          weight = value;
          ChangePrice();
          OnPropertyChanged("Price");
          OnPropertyChanged("Tariff");
          OnPropertyChanged("Weight");
       }
    }
}
```

Даний клас побудовано на основі класу DBTool, який встановлює підключення до БД, виконує відповідні запити для витягання даних, внесення даних в БД та оновлення цих даних.

Окрім методів даного типу клас також має наступні методи, зокрема конструктори:

– Package (string track, int ptype, int dtype, float weight, float size1, float size2, float size3, int status, string location, Client client1, DateTime date1, string address1, Department department1, Client client2, DateTime date2, string address2, Department department2, float price, float paid, float calctariff) – конструктор, який дозволяє встановити значення всіх атрибутів за переданими параметрами окрім номера;

– Package () – конструктор за замовчуванням без параметрів, який не містить дій, але дозволяє при цьому іменовано передати значення потрібних параметрів;

– Package (int id) – конструктор, який дозволяє автоматично встановити значення всіх атрибутів за заданим ідентифікаційним номером id.

Метод bool Save () дозволяє зберегти в БД новий запис про поточний пакунок.

Метод bool Update () дозволяє оновити дані про існуючий запис про пакунок в БД.

Метод bool GetByClient () дозволяє внести зміни в атрибут стану пакунку як поточного об’єкта та внести відповідні зміни одразу в БД.

Метод Client NewReceiver (string surname, string name, string mname, string phone, string address) дозволяє створити новий об’єкт клієнта-отримувача під час роботи над пакунком, якщо такого запису ще немає, або повернути існуючий об’єкт. Для цього відбувається перевірка за номером телефону:

```
int cl = getClient(phone);
if (cl == -1)
{
    Client fc = new Client();
    fc.Surname = surname;
    fc.Name = name;
    fc.MiddleName = mname;
    fc.Phone = phone;
    fc.Address = address;
    fc.Save();
    return fc;
}
else
{
    Client fc = new Client(cl);
    return fc;
```

}

Метод List<Department> GetPackageDepartments () повертає перелік всіх відділень, які здатні працювати з пакунком даного типу. Для цього здійснюється перевірка розмірів та ваги пакунку і в залежності від того, до якої групи належить даний пакунок, відбувається відсікання саме цих відділень.

Метод float CalcPrice() виконує обчислення вартості доставки в залежності від заданого варіанту адрес відправлення.

Метод bool ChangePrice() виконує не тільки обчислення вартості доставки, але і при цьому змінює значення відповідного атрибуту об’єкта, а також встановлення відповідного тарифу за необхідності.

Метод DateTime CalcDate() виконує обчислення очікуваної дати і часу доставки пакунку.

Метод bool ChangeDate() виконує обчислення очікуваної дати і часу доставки пакунку і змінює відповідні атрибути об’єкта на обчислені значення.

У моделях представлень реалізовано визначення у відповідних класів об’єктів та їх наборів, необхідних для роботи відповідного представлення. Для кожного класу визначено набір команд, які обробляють дії користувача (натиснення відповідної кнопки).

Зв’язування з представленням і прив’язування відповідних даних виконується наступним чином:

```
ClientPackageDetalizationView cpdView = new ClientPackageDetalizationView ((Package)obj);
cpdView.ShowDialog();
OnPropertyChanged("OwnPackages");
```

У даному випадку визначено роботу з пакунками, отримувачем яких є авторизований користувач.

3.3 Оброблення некоректних ситуацій

Для оброблення некоректних ситуацій передбачено в кожному класі набір відповідних методів.

Зокрема в базовій моделі, організованій за допомогою класу Package, передбачено такі методи.

Метод bool OkCheck організовує консолідовану перевірку всіх атрибутів об’єкту пакунку і якщо всі дані задано коректно, то повертає істинне значення, якщо хоча б якесь ні, то повертає хибне значення.

Метод string GetValidationError(string pos) виконує перевірку коректності введеного значення, встановленого за властивістю pos і повертає за наявності помилки відповідний текст повідомлення або порожнє значення, якщо помилки не виявлено.

Відповідні методи для перевірки атрибутів працюють наступним чином.

Наприклад, перевірка коректності введення значення адреси кур’єрської доставки реалізується наступним чином:

```
if (Address2.Count() > 255)
{
    return "Занадто довге значення";
}
else if (Address2 == Address1 && ! String.IsNullOrEmpty(Address2))
{
    return "Адреси не можуть співпадати";
}
return null;
```

При цьому спочатку перевіряється, що задане значення вміщається в обмеження БД, тобто довжина не перевищує 255 символів, а далі перевіряється, чи не співпадає адреса відправлення з адресою доставки.

3.4 Висновки за розділом 3

Представлено структуру розробленого програмного забезпечення з приведенням відповідних класів моделей, моделей представлень та представлень у вигляді структурної схеми.

Описано основні принципи розробки та засоби розроблених класів. Описано процес обробки некоректних ситуацій, що забезпечується в цих класах.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення програми

Програмне забезпечення призначене для керування процесом екпрес-доставки посилок, документів, вантажів як в режимі роботи між відділеннями, так і з використанням засобів кур’єрської доставки.

Програмне забезпечення може використовуватися клієнтами сервісу, менеджерами, операторами та кур’єрами.

4.2 Умови виконання програми

Вимоги до технічних засобів клієнтської сторони:

– стаціонарний або мобільний (ноутбук) комп'ютер під керуванням операційної системи Windows, починаючи з версії 7;

– процесор не нижче Intel Celeron з тактовою частотою не менше 300 МГц;

– не менше 1 Гб оперативної пам’яті;

– 100 Мб вільного простору на жорсткому диску;

– монітор і відеокарта, які мають забезпечувати роботу з роздільною здатністю не менше 800x600.

4.3 Тестування сценаріїв роботи з програмою

Після запуску програми відкривається стандартна форма авторизації. Після успішного виконання авторизації (або реєстрації облікового запису клієнта) відкривається основне вікно роботи клієнта (рис. 4.1).

Клієнт для створення власного відправлення має першочергово заповнити дані про себе в групі «Відправник», які будуть використовуватися для формування даних про відправлення пакунку. Для визначення

найближчого відділення сервісу потрібно спочатку задати місто у відповідному полі, а далі обрати з переліку потрібне відділення. Виконаний вибір буде використовуватися при створенні замовлення на доставку в якості даних відправника. Після внесення або коригування відповідних даних потрібно натиснути на кнопку «Зберегти».

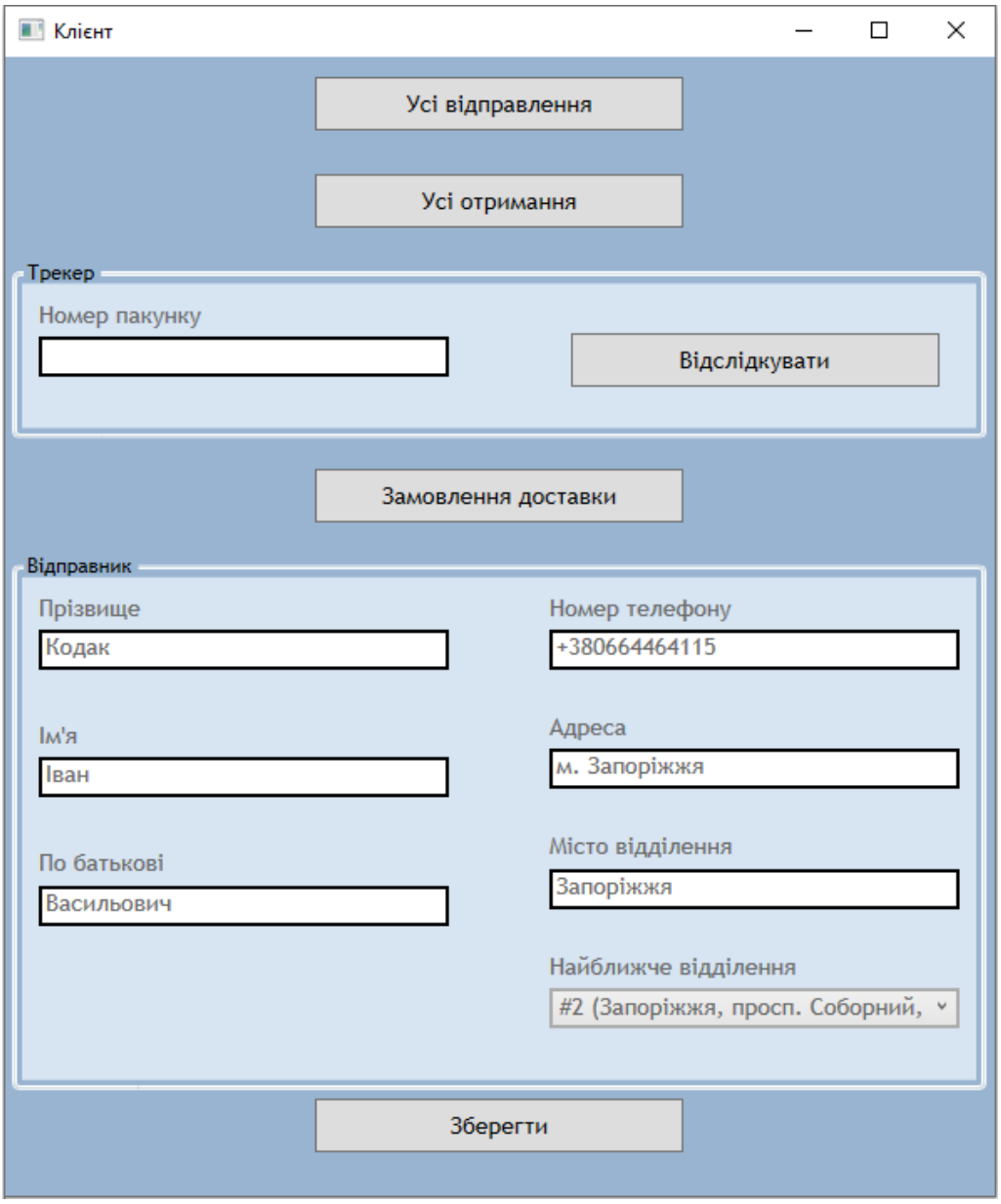

Рисунок 4.1 – Основна форма клієнта

Для відслідковування будь-якого відправлення (у такому випадку буде відображатися не повна інформація про відправника та отримувача) потрібно ввести в групі «Трекер» трекінговий номер пакунку, після чого

натиснути на кнопку «Відслідкувати», тоді відкриється форма перегляду детальних даних про пакунок. При перегляді такої інформації з форм усіх відправлень або отримань відкривається повний варіант форми, у даному випадку – обмежений.

Для створення нового замовлення доставки потрібно натиснути на кнопку «Замовлення доставки», тоді клієнт може самостійно створювати пакунок (рис. 4.2). Ця форма взаємодії названа замовленням на доставку, оскільки всі внесені дані потребують наступного уточнення оператором або кур’єром, лише після цього замовлення стане повноцінним пакунком, тобто перейде в стан прийнятого пакунку і відповідно даним про пакунок можна буде довіряти.

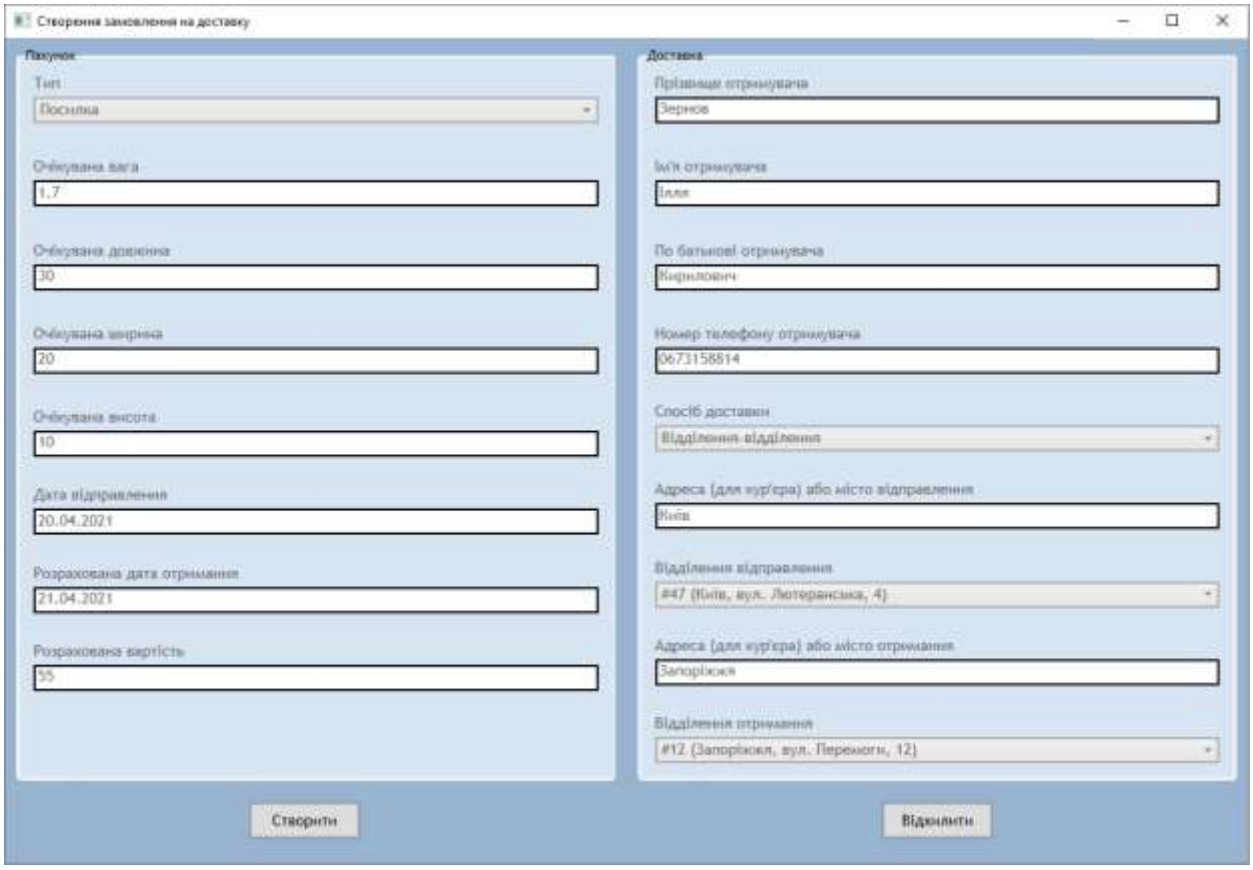

Рисунок 4.2 – Створення замовлення на доставку клієнтом

Всі дані замовлення поділені на 2 групи: пакунок та доставка.

Перша група визначає загальні параметри пакунку:

– тип, що може включати посилку, вантаж або документи;

– очікувані значення ваги і розмірів;

– дату відправлення: дане поле виключено з доставки, адже за замовчуванням встановлюється поточна дата і час, відповідно дане поле може не оперувати в процесі визначення параметрів доставки, але за необхідності може бути змінено;

– розраховану дату отримання: дата визначається автоматично на основі вказаних параметрів доставки, користувач не може вводити в це поле ніяких даних (режим читання);

– розраховану вартість, що обчислюється на основі параметрів пакунку та заданих параметрів доставки (режим читання).

Друга група параметрів стосується безпосередньо доставки і включає:

– особисті дані отримувача;

– спосіб доставки (з участю кур'єра або без такої);

– пару адреса і відділення відправлення: для вибору способу відправлення з участю кур'єра адреса вказує повну адресу, за якою буде забрано пакунок, а для вибору способу відправлення з відділення в полі адреси потрібно вказати назву міста, а з переліку відділень обрати потрібне відділення в цьому місті;

– пару адреса і відділення отримання: для вибору способу отримання з участю кур'єра адреса вказує повну адресу, за якою буде забрано пакунок, а для вибору способу отримання з відділення в полі адреси потрібно вказати назву міста, а з переліку відділень обрати потрібне відділення в цьому місті.

Для збереження замовлення потрібно натиснути на кнопку «Створити» внизу, а для скасування, щоб повернутися до основної форми без збереження, потрібно натиснути на кнопку «Відхилити».

Клієнт вводить тільки орієнтовні дані про розміри пакунку, його вагу, ці дані потрібні, що сумістити в даному вікні зі створенням замовлення на доставку розрахунок вартості такої доставки та строків доставки.

Під час збереження замовлення виконується перевірка контактних

даних клієнта. Якщо вже збережено в БД номер телефону такого клієнта, то створюється посилання на цей запис, а якщо такого запису не було знайдено, то спочатку він створюється, а далі використовується. Аналогічні дії виконуються під час реєстрації. Якщо користувач вкаже номер телефону, який вже був збережений, то буде створено посилання на існуючий клієнтський запис.

Відбувається перевірка всіх введених полів. Якщо хоча би в одному з полів введено дані некоректно, то буде виведено відповідне повідомлення під час підтвердження збереження замовлення.

При натисканні на кнопку «Всі відправлення» основної форми клієнта (рис. 4.1) відкривається форма перегляду всіх власних відправлень авторизованого користувача, в яких він є відправником.

При натисканні на кнопку «Всі отримання» основної форми клієнта (рис. 4.1) відкривається форма перегляду всіх власних відправлень авторизованого користувача, в яких він є отримувачем (рис. 4.3).

Рисунок 4.3 – Робота клієнта з власними отриманнями

Якщо обрати один з пакунків, натиснувши на кнопку «Перейти», то відкриється вікно перегляду повних даних пакунку (рис. 4.4).

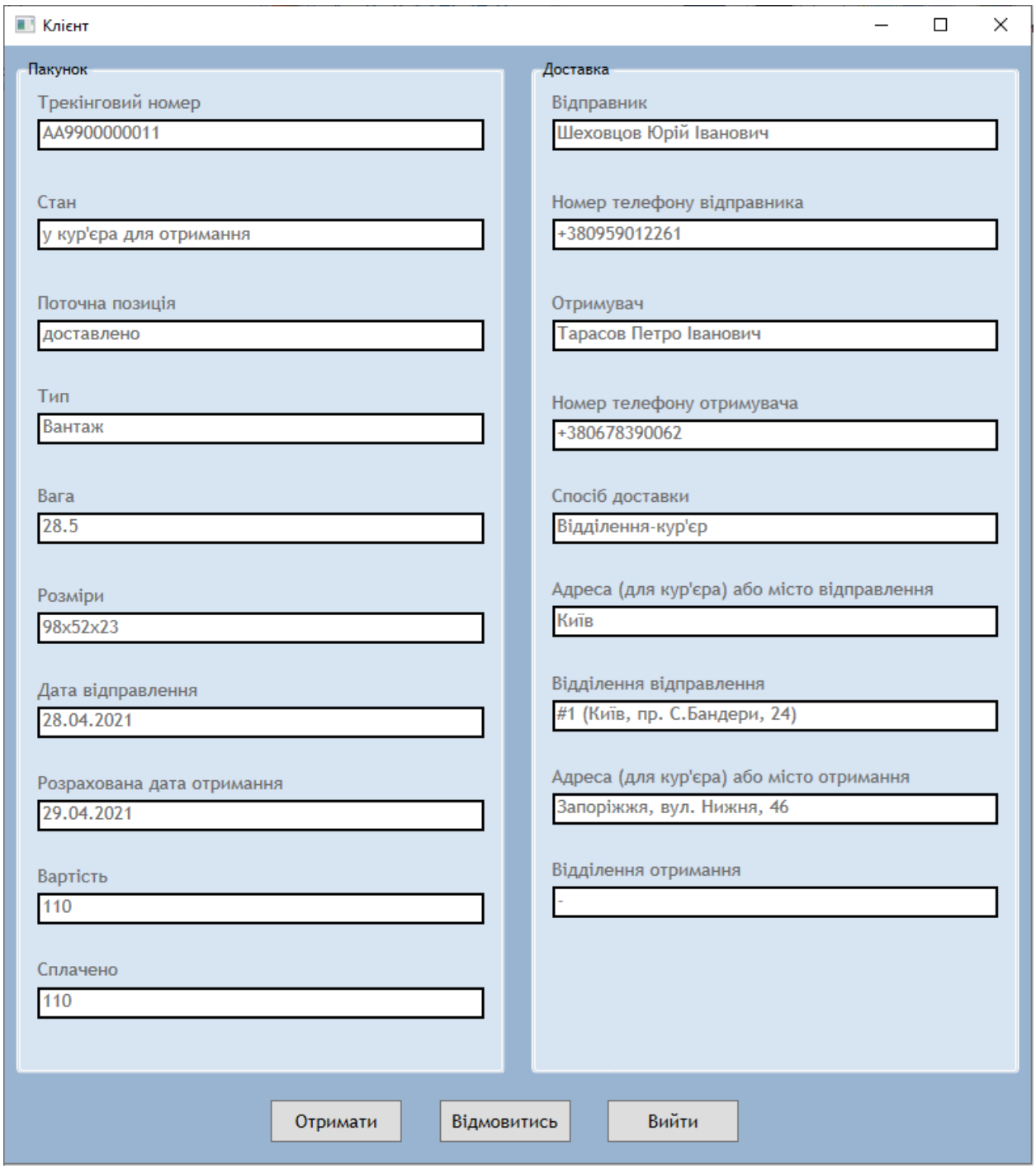

Рисунок 4.4 – Робота клієнта з власним пакунком

У даному випадку стан вікна відображено для пакунку, який доставляється кур’єрською доставкою додому, при цьому пакунок вже знаходиться у кур’єра, який прибув до клієнта, а клієнт сплатив йому за доставку. Відповідно тепер клієнт має можливість підтвердити отримання замовлення. Таким чином забезпечується аналогія підпису клієнта в документах. Натискання на кнопку «Отримати» буде відповідати

підтвердженню отримання пакунку від кур’єра. Дана кнопка активна тільки тоді, коли пакунок знаходиться в даному поточному стані і внесена сплата за пакунок дорівнює вартості доставки. Весь інший час кнопка буде неактивна.

Якщо клієнт вирішив відмовитись від отримання пакунку, то він має натиснути на кнопку «Відмовитись». Дана кнопка активна тільки тоді, коли пакунок досягає даного поточного стану.

Для виходу без збереження змін потрібно натиснути на кнопку «Вийти».

Усі поля даної форми відображаються в режимі для читання, тобто вносити зміни клієнт не може у пакунок.

Якщо в програмі було авторизовано оператора, то відкривається основне вікно оператора (рис. 4.5).

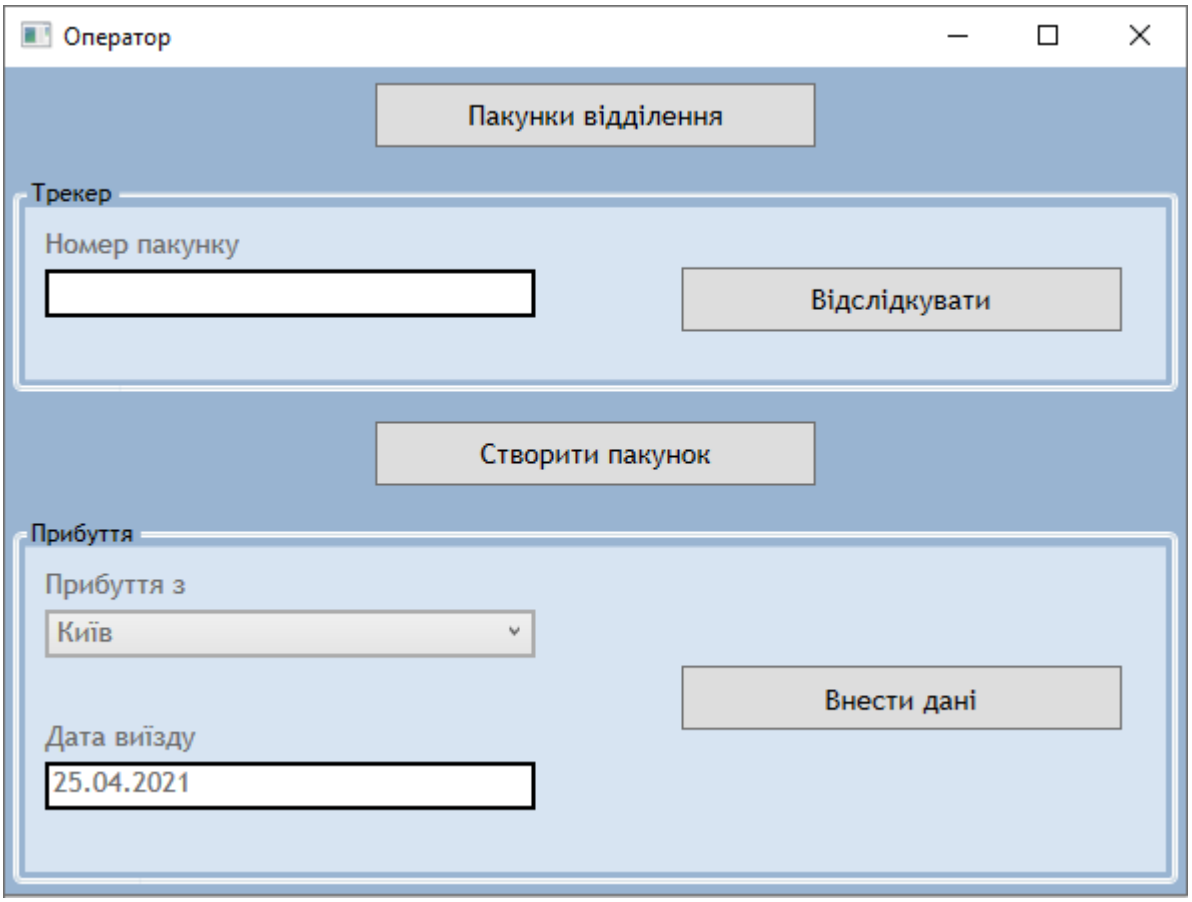

Рисунок 4.5 – Основна форма оператора

Оператор може переглянути набір пакунків, які в даний момент знаходяться в роботі в відділенні, тобто доставленні у відділення, але ще не забрані клієнтами або такі, від яких клієнти відмовились. Для цього потрібно натиснути на кнопку «Пакунки відділення».

Для того щоб створити нове відправлення, коли клієнт прийшов у відділення і не має внесеного замовлення в систему, потрібно натиснути на кнопку «Створити пакунок».

Для внесення даних при прибуття посилок з прибуттям транспорту потрібно спочатку обрати місто, з якого прибув автомобіль, далі вказати дату її виїзду і натиснути на кнопку «Внести дані», тоді відкриється вікно роботи з пакунками, що мали бути доставленими цим автомобілем, на якій можна відзначати ті пакунки, що прибули, встановлюючи галочки, тобто не потрібно переходити на окреме вікно з кожним окремим пакунком.

Для відслідковування будь-якого відправлення потрібно ввести в групі «Трекер» трекінговий номер пакунку, після чого натиснути на кнопку «Відслідкувати», тоді відкриється форма перегляду повних і детальних даних про пакунок (рис. 4.6).

Цей же варіант потрібно використовувати, коли клієнт приніс пакунок у відділення для відправлення, попередньо створивши замовлення самостійно.

У даному випадку під час роботи з замовленням оператору не на читання доступне тільки поле «Сплачено», адже пакунок знаходиться в стані «у відділенні отримання». Оператор може у даному випадку натиснути на кнопку «Видача» для фіксування факту видачі пакунку одержувачу. При цьому перед цим оператор має внести в сплату суму, що дорівнює вартості доставки. Якщо відповідна сума не задана, а натиснути на кнопку «Видача», то буде виведено повідомлення про необхідність внесення сплати, а стан пакунку змінений не буде.

Якщо клієнт відмовився від отримання пакунку, то оператор має натиснути на кнопку «Відмова».

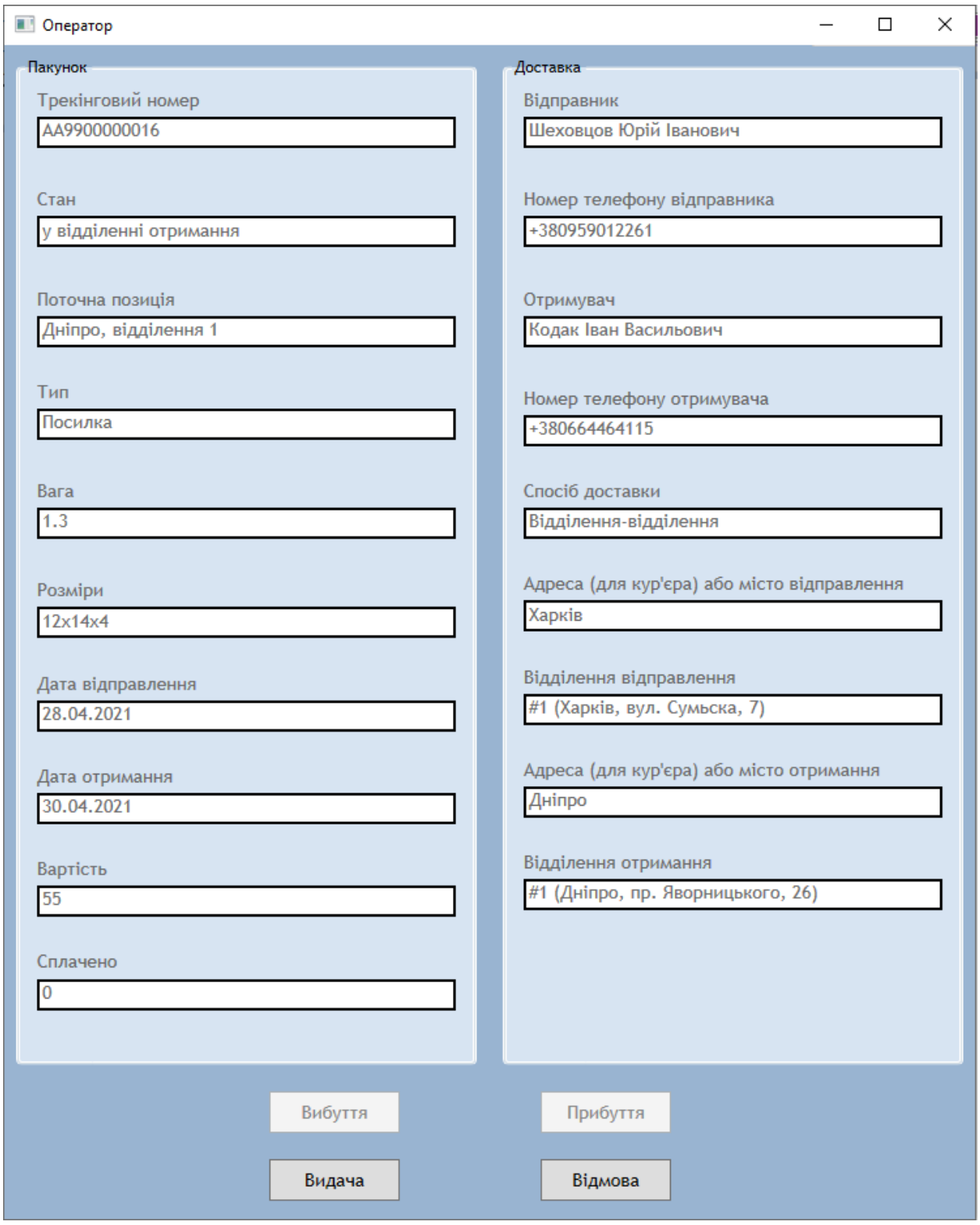

Рисунок 4.6 – Керування пакунком оператором у відділенні

До моменту доставки пакунку у відділення для нього кнопки «Відмова» і «Видача» недоступні. Аналогічно робота відбувається для кур’єра.

Кнопка «Вибуття» доступна для пакунку, що міститься у відділенні, відповідно при його відправленні оператор таким чином може фіксувати відправлення пакунку. В інший момент часу кнопка неактивна.

Кнопка «Прибуття» дозволяє зафіксувати прибуття пакунку у відділення призначення. Призначена для автоматичної зміни стану пакунку. Доступна тільки для пакунку в дорозі, в іншому випадку неактивна.

У результаті для прибуття пакунків відкривається окрема форма для обраного пункту відправлення та зазначеної дати і відділення, для якого належить авторизований оператор (рис. 4.7).

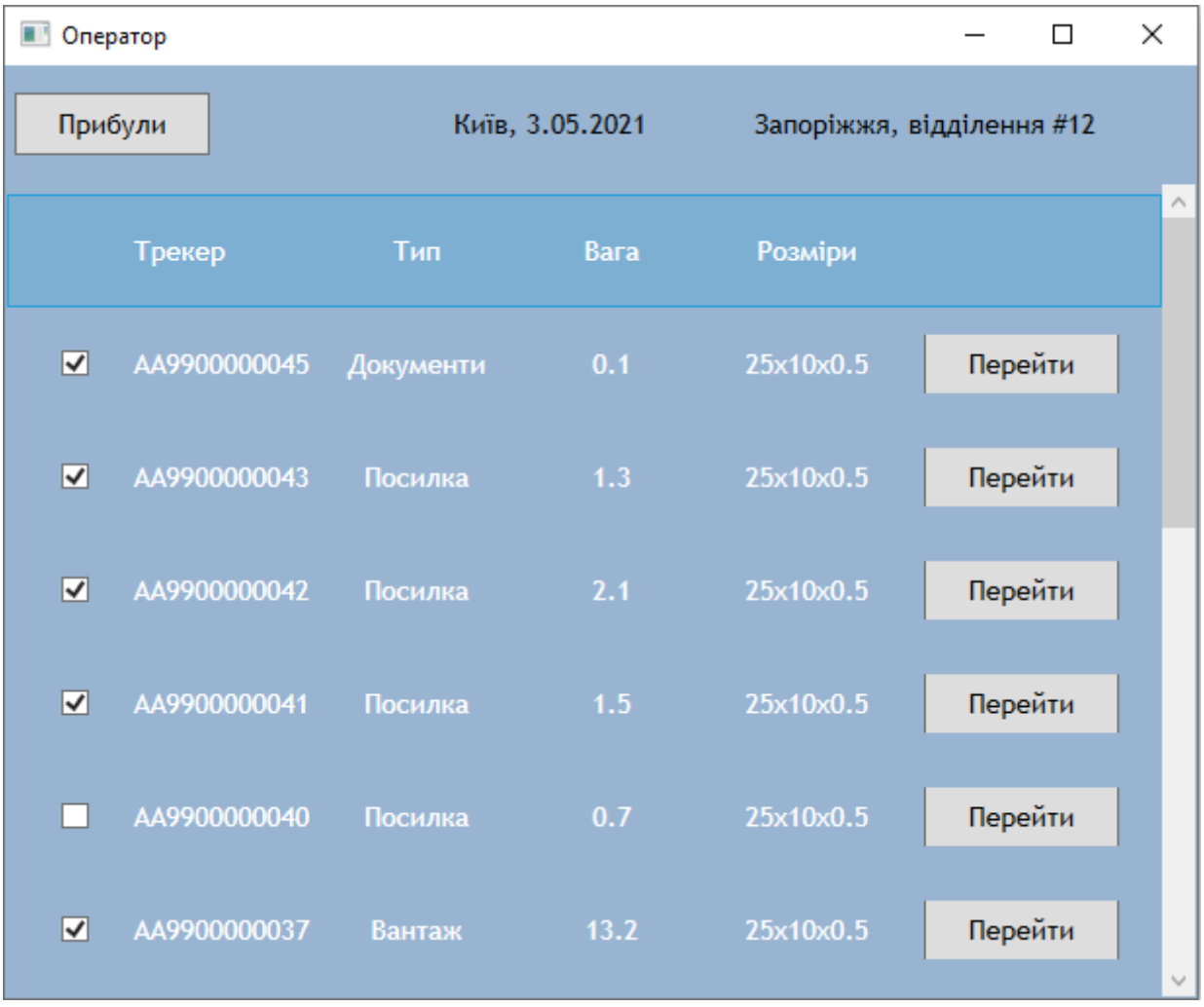

Рисунок 4.7 – Прибуття пакунків

Аналогічно відбувається робота кур’єра та менеджера, робота яких розпочинається з аналогічних основних форм. При цьому кур’єр може

переглядати всі пакунки, виділені йому для доставки на день, а менеджер може працювати зі всіма пакунками, що є у відділенні, визначаючи (подібно оператору під час прибуття), які пакунки якому кур’єру призначити.

4.4 Висновки за розділом 4

Описано основні режими роботи користувачів різного типу з програмою. Описано процес функціонування програми щодо оброблення дій користувача, а також представлено можливі варіанти такої роботи.

ВИСНОВКИ

Дипломну кваліфікаційну роботу бакалавра присвячено проблемі підтримки сервісу експрес-доставки. Для цього розглянуто основні можливості сервісів експрес-доставки, особливості експрес-доставки порівняно зі звичайною доставкою пакунків, що включають швидкість, безпеку та точність виконання операцій, представлено моделі організації експрес-доставки та програмного забезпечення з даної сфери, обрано модель програм поштової та кур'єрської служби в якості основи для розробки програмного забезпечення в роботі.

Було обрано мову програмування C# для реалізації програмного забезпечення, визначено функціональні вимоги до розроблюваного програмного забезпечення, на основі цього побудовано діаграму прецедентів для роботи клієнтів, операторів, менеджерів та кур’єрів з програмою. Запроєктовано БД з визначенням таблиць і їх полів та побудовою схеми БД. Архітектуру програмного забезпечення побудовано на основі патерну MVVM, що і стало основою визначення структури програмного забезпечення. Виконано програмну реалізацію сервісу і описано розроблені класи.

Створене програмне забезпечення надає можливість керування процесом екпрес-доставки шляхом відслідковування стану доставки пакунку, перегляду всіх власних відправлених пакунків і пакунків для отримання, визначення очікуваної дати прибуття пакунку, підтвердження доставки кур’єром, видачі пакунку, визначення прибуття пакунків, відправлення пакунку, внесення попередніх даних відправлення, перегляду пакунків за відділенням, розподілу пакунків за кур’єрами.

Всі завдання, які було визначено у дипломній кваліфікаційній роботі бакалавра, виконано у повному обсязі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Gbenga-Labiyi, F. How can parcel (package) delivery logistics company re-engineering their process to minimize the high send-again (returns) [Electronic resource] : Master Thesis within Business Administration / F. Gbenga-Labiyi. – Jonkoping : Jonkoping University, 2018. – Access mode : https://www.diva-portal.org/smash/get/diva2:1293109/FULLTEXT01.pdf.

2. Karcz, J. Improvements in the quality of courier delivery [Text] / J. Karcz, B. Slusarczyk // International Journal for Quality Research. – 2016. – № 10 (2). – Pp. 355-372.

3. Zhu, H. Logistics System and Process in Express Delivery Service Companies [Electronic resource] : Bachelor's Thesis / H. Zhu. – Jyväskylä : JAMK University of Applied Sciences, 2010. – Access mode : https://www.theseus.fi/bitstream/handle/10024/16763/Zhu_Hanzheng.pdf.

4. Four Key Elements of a Service Delivery System [Electronic resource]. – Access mode : https://www.servicefutures.com/four-key-elements-service-delivery-system.

5. Ramstad, K. F. 4 business models that are disrupting the parcel delivery indusrty [Electronic resource]. – Access mode : https://www.mixmove.io/blog/4-business-models-that-are-disrupting-the-parcel-delivery-industry.

6. The Guide to Building an On-demand Parcel Delivery App [Electronic resource]. – Access mode : https://easternpeak.com/blog/the-most-complete-guide-to-building-an-on-demand-parcel-delivery-app/.

7. FedEx | Експрес-доставка, кур'єрські та транспортні послуги [Електронний ресурс]. – Режим доступу : https://www.fedex.com/uk-ua/home.html.

8. FedEx Mobile – Додатки в Google Play [Електронний ресурс]. – Режим доступу : https://play.google.com/store/apps/details?id=com.fedex.ida.android&hl=uk&gl=US.

9. Глобальна логістика – міжнародні перевезення | DHL Home | Україна [Електронний ресурс]. – Режим доступу : https://www.dhl.com/ua-uk/home.html.

10. DHL | Press Release | English [Electronic resource]. – Access mode : https://www.dhl.com/en/press/releases/releases_2013/logistics/dhl_app_simplifies_worldwide_logistics_management.html#.YF8B-68zaUl.

11. Обзор языка C# – руководство по C# | Microsoft Docs [Электрон. ресурс]. – Режим доступа : https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/.

12. Фленов, М. Библия C# [Текст] / М. Фленов. – СПб. : БХВ-Петербург, 2020. – 512 с.

13. Страуструп, Б. Язык программирования C++. Краткий курс [Текст] / Б. Страуструп. – М. : Диалектика, 2019. – 320 с.

14. Рао, С. Освой самостоятельно C++ за 21 день, 7-е издание (C++11) [Текст] / С. Рао. – М. : Вильямс, 2013. – 688 с.

ДОДАТОК А

Технічне завдання

Вступ

Програмне забезпечення, що розробляється, повинно забезпечувати повний контроль як за відслідковуванням відправлених пакунків різними групами користувачів, так і за керування даним процесом.

А.1 Підстави для розробки

Розробка має бути виконана в межах завдання на дипломну кваліфікаційну роботу, яке було поставлено за темою «Програмне забезпечення сервісу експрес-доставки» у відповідності з наказом № 103 від 30 березня 2021 р. за Національним університетом «Запорізька політехніка».

А.2 Призначення розробки

Розроблюване програмне забезпечення призначене для керування процесом екпрес-доставки посилок, документів, вантажів як в режимі роботи між відділеннями, так і з використанням засобів кур’єрської доставки. Програмне забезпечення може використовуватися клієнтами сервісу, менеджерами, операторами та кур’єрами.

А.3 Основні вимоги до програми

А.3.1 Вимоги до функціональних характеристик

Для реалізації визначеного призначення програма має надавати такі функціональні особливості:

– відслідковування стану доставки пакунку за номером;

– перегляд всіх власних відправлених пакунків;

– перегляд всіх пакунків для отримання;

– визначення очікуваної дати прибуття пакунку;

– підтвердження доставки кур’єром;

– видача пакунку;

– прибуття пакунків;

– відправлення пакунку;

– внесення попередніх даних відправлення;

– пошук пакунку;

– перегляд пакунків за відділенням;

– визначення відділень доставки за параметрами пакунку;

– реєстрація облікового запису клієнта (відправника або отримувача);

– авторизація клієнта, менеджера, оператора, кур’єра;

– перегляд кур’єром всіх пакунків для доставки;

– розподіл пакунків за кур’єрами.

А.3.2 Вимоги до надійності

За результатами введених користувачем даних має здійснюватися перевірка того, чи було введено коректні дані для кожного поля (зокрема для кожного рядка має перевірятися, чи введене значення не є більшим за довжину вказану для відповідного поля в таблиці БД). Окрім того має перевірятися, чи вірно задано співвідношення між даними (наприклад, дата відправлення не може бути пізнішою за дату отримання, а пакунок не можна відправити самому собі).

А.3.3 Умови експлуатації

Для експлуатації програми особливих вмінь від користувачів окрім вміння користуватися комп’ютером не висувається.

Для забезпечення роботи БД на сервері необхідні вміння системного адміністратора для підтримки роботи сервера системи керування БД Microsoft SQL Server.

А.3.4 Вимоги до складу та параметрів технічних засобів

Вимоги до технічних засобів клієнтської сторони:

– стаціонарний або мобільний (ноутбук) комп'ютер під керуванням операційної системи Windows, починаючи з версії 7;

– процесор не нижче Intel Celeron з тактовою частотою не менше 300 МГц;

– не менше 1 Гб оперативної пам’яті;

– 100 Мб вільного простору на жорсткому диску;

– монітор і відеокарта, які мають забезпечувати роботу з роздільною здатністю не менше 800х600.

А.4 Порядок контролю та приймання

Принципи контролю виконання роботи визначаються календарним планом, розташованим у завданні на дипломну кваліфікаційну роботу, де вказано назви етапів, строки їх виконання та результат.

ДОДАТОК Б

Текст програми

```
        using System;
        using System.Collections.Generic;
        using System.Collections.ObjectModel;
        using System.ComponentModel;
        using System.Data.SqlClient;
        using System.Linq;
        using System.Runtime.CompilerServices;
        using System.Text.RegularExpressions;

        namespace ExDel.Model
        {
            public class Package : DBTool, INotifyPropertyChanged,
IDataErrorInfo
            {
                private int id = -1;
                private string track = "";
                private int ptype = 0;
                private int dtype = 0;
                private float weight = 0;
                private float size1 = 0;
                private float size2 = 0;
                private float size3 = 0;
                private int status = 0;
                private string location = "";
                private Client client1 = new Client();
                private DateTime date1 =
DateTime.Now.ToString("dd.MM.yyyy");
                private string address1 = "";
                private Department department1 = new Department();
                private Client client2 = new Client();
                private DateTime date2 =
DateTime.Now.ToString("dd.MM.yyyy");
                private string address2 = "";
                private Department department2 = new Department();
                private float price = 0;
                private float paid = 0;
                private float calctariff = new Tariff();
                public string ptypetext { get { return PTypes[ptype]; }
}
                public string dtypetext { get { return DTypes[dtype]; }
}
                public string statustext { get { return
Statuses[status]; } }

                public int ID
                {
                    get { return id; }
```

```
        set
        {
            if (id != value)
            {
                id = value;
                Track = setNextTrack();
                OnPropertyChanged("ID");
            }
        }
    }

    public string Track
    {
        get { return track; }
        set
        {
            if (track!= value)
            {
                track = value;
                OnPropertyChanged("Track");
            }
        }
    }

    public int PType
    {
        get { return ptype; }
        set
        {
            if (ptype != value)
            {
                ptype = value;
                OnPropertyChanged("PackageType");
            }
        }
    }

    public int DType
    {
        get { return dtype; }
        set
        {
            if (dtype != value)
            {
                dtype = value;
                ChangePrice();
                OnPropertyChanged("Price");
```

```
            OnPropertyChanged("Tariff");
            OnPropertyChanged("DeliveryType");
        }
    }
}

public float Weight
{
    get { return weight; }
    set
    {
        if (weight != value)
        {
            weight = value;
            ChangePrice();
            OnPropertyChanged("Price");
            OnPropertyChanged("Tariff");
            OnPropertyChanged("Weight");
        }
    }
}

public float Size1
{
    get { return size1; }
    set
    {
        if (size1 != value)
        {
            size1 = value;
            OnPropertyChanged("SizeX");
        }
    }
}

public float Size2
{
    get { return size2; }
    set
    {
        if (size2!= value)
        {
            size2 = value;
            OnPropertyChanged("SizeY");
        }
    }
}
```

```
public float Size3
{
    get { return size3; }
    set
    {
        if (size3!= value)
        {
            size3 = value;
            OnPropertyChanged("SizeZ");
        }
    }
}

public int Status
{
    get { return status; }
    set
    {
        if (status != value)
        {
            status  = value;
            OnPropertyChanged("DeliveryStatus");
        }
    }
}

public string Location
{
    get { return location; }
    set
    {
        if (location != value)
        {
            location = value;
            OnPropertyChanged("Location");
        }
    }
}

public CLient Client1
{
    get { return client1; }
    set
    {
        if (client1 != value)
        {
            client1 = value;
```

```
            OnPropertyChanged("Sender");
        }
    }
}

public DateTime Date1
{
    get { return date1; }
    set
    {
        if (date1 != value)
        {
            date1 = value;
            ChangeDate();
            OnPropertyChanged("RecDate");
            OnPropertyChanged("SendDate");
        }
    }
}

public string Address1
{
    get { return address1; }
    set
    {
        if (address1! = value)
        {
            address1 = value;
            ChangePrice();
            OnPropertyChanged("Price");
            OnPropertyChanged("Tariff");
            OnPropertyChanged("SenderAddress");
        }
    }
}

public Department Department1
{
    get { return department1; }
    set
    {
        if (department1 != value)
        {
            department1 = value;
            ChangePrice();
            OnPropertyChanged("Price");
            OnPropertyChanged("Tariff");
```

```
            OnPropertyChanged("SenderDepartment");
        }
    }
}

public CLient Client2
{
    get { return client2; }
    set
    {
        if (client2 != value)
        {
            client2 = value;
            OnPropertyChanged("Receiver");
        }
    }
}

public DateTime Date2
{
    get { return date2; }
    set
    {
        if (date2 != value)
        {
            date2 = value;
            OnPropertyChanged("RecDate");
        }
    }
}

public string Address2
{
    get { return address2; }
    set
    {
        if (address2 ! = value)
        {
            address2 = value;
            ChangePrice();
            OnPropertyChanged("Price");
            OnPropertyChanged("Tariff");
            OnPropertyChanged("RecAddress");
        }
    }
}
```

```
public Department Department2
{
    get { return department2; }
    set
    {
        if (department2 != value)
        {
            department2 = value;
            ChangePrice();
            OnPropertyChanged("Price");
            OnPropertyChanged("Tariff");
            OnPropertyChanged("RecDepartment");
        }
    }
}

public float Price
{
    get { return price; }
    set
    {
        if (price != value)
        {
            price = value;
            OnPropertyChanged("Price");
        }
    }
}

public float Paid
{
    get { return paid; }
    set
    {
        if (paid != value)
        {
            paid = value;
            OnPropertyChanged("Paid");
        }
    }
}

public Tariff CalcTariff
{
    get { return calctariff; }
    set
    {
```

```
                    if (calctariff != value)
                    {
                        calctariff = value;
                        OnPropertyChanged("Tariff");
                    }
                }
            }

            public List<String> PTypes
            {
                get
                {
                    return new List<string> { "Не задано", "Вантаж",
"Документи", "Посилка"};
                }
            }

            public List<String> DTypes
            {
                get
                {
                    return new List<string> { "Не задано",
"Відділення-відділення", "Кур’єр-відділення", "Відділення-кур’єр",
"Кур’єр-кур’єр"};
                }
            }

            public List<String> Statuses
            {
                get
                {
                    return new List<string> { "Не задано",
"Створено", "Забрано кур’єром", "У відділенні відправлення", "На шляху
до сортувального центру відправлення", "У сортувальному центрі
відправлення", "У дорозі", "У сортувальному центрі отримання", "На
шляху до відділення отримання", "У кур’єра для отримання", "У
відділенні отримання", "Отримано", "Відмова", "Скасовано"};
                }
            }

            public Package (string track, int ptype, int dtype,
float weight, float size1, float size2, float size3, int status,
string location, Client client1, DateTime date1, string address1,
Department department1, Client client2, DateTime date2, string
address2, Department department2, float price, float paid, float
calctariff)
            {
```

```
        Track = track;
        PType = ptype;
        DType = dtype;
        Weight = weight;
        Size1 = size1;
        Size2 = size2;
        Size3 = size3;
        Status = status;
        Location = location;
        Client1 = client1;
        Date1 = date1;
        Address1 = address1;
        Department1 = department1;
        Client2 = client2;
        Date2 = date2;
        Address2 = address2;
        Department2 = department2;
        Price = price;
        Paid = paid;
        CalcTariff = calctariff;
    }

    public Package ()
    {
    }

    public Package (int id)
    {
        Package p = getPackageFromId(id);
        ID = p.ID;
        Track = p.Track;
        PType = p.PType;
        DType = p.DType;
        Weight = p.Weight;
        Size1 = p.Size1;
        Size2 = p.Size2;
        Size3 = p.Size3;
        Status = p.Status;
        Location = p.Location;
        Client1 = p.Client1;
        Date1 = p.Date1;
        Address1 = p.Address1;
        Department1 = p.Department1;
        Client2 = p.Client2;
        Date2 = p.Date2;
        Address2 = p.Address2;
        Department2 = p.Department2;
```

```
                Price = p.Price;
                Paid = p.Paid;
                CalcTariff = p.Calctariff;
                return p;
            }

            public bool Save ()
            {
                int id = addPackage (Track, PType, DType, Weight,
Size1, Size2, Size3, Status, Location, Client1.ID, Date, Address1,
Department1.ID, Client2.ID, Date2, Address2, Department2.ID, Price,
Paid, CalcTariff.ID);
                if (id == -1) return false;
                else
                {
                    ID = id;
                    return true;
                }
            }

            public bool Update ()
            {
                return updatePackage (ID, Track, PType, DType,
Weight, Size1, Size2, Size3, Status, Location, Client1.ID, Date,
Address1, Department1.ID, Client2.ID, Date2, Address2, Department2.ID,
Price, Paid, CalcTariff.ID);
            }

            public bool GetByClient ()
            {
                if (Status == 12 || Status == 9)
                {
                  Status = 11;
                  if (setPackageStatus(ID, Status))
                    return true;
                  else return false;
                }
                else return false;
            }

            public Client NewReceiver (string surname, string name,
string mname, string phone, string address)
            {
                int cl = getClient(phone);
                if (cl == -1)
                {
                   Client fc = new Client();
```

```
                    fc.Surname = surname;
                    fc.Name = name;
                    fc.MiddleName = mname;
                    fc.Phone = phone;
                    fc.Address = address;
                    fc.Save();
                    return fc;
                }
                else
                {
                    Client fc = new Client(cl);
                    return fc;
                }
            }

            public List<Department> GetPackageDepartments ()
            {
                List <Department> ld = getAllDepartments();
                List<Department> ldres = new List<Department>();
                int pwt = -1;
                if (Weight < 30 && Size1 < 120  && Size2 < 120  &&
Size2 < 120)
                    pwt = 1;
                else if (Weight < 30 && Size1 < 58  && Size2 < 24
&& Size2 < 42)
                    pwt = 0;
                else if (Weight < 1000 && Size1 < 300  && Size2 <
170  && Size2 < 170)
                    pwt = 2;
                foreach (Department d in ld)
                    if (d.WType <= pwt)
                        ldres.Add(d);
                return ldres;
            }

            public float CalcPrice()
            {
                string city1 = "";
                string city2 = "";
                float p = 0;
                if (Department1.ID == 0)
                {
                    city1 = Address1.Split(' ')[0];
                }
                Center cl1 = getCenterDepartmentFromPlace(city1);
                if (Department2.ID == 0)
                {
```

```
                    city2 = Address2.Split(' ')[0];
                }
                Center cl2 = getCenterDepartmentFromPlace(city2);
                float rst = getRouteDuration(cl1.DepartmentID,
cl2.DepartmentID);
                int zone;
                if (rst < 30)
                    zone = 0;
                else if (rst < 200)
                    zone = 1;
                else zone = 2;
                Tariff t = getTariff(zone, Weight);
                p = t.Price;
                float ap = t.AddPrice;
                if (Department1.ID == 0)
                    p = p + ap;
                if (Department2.ID == 0)
                    p = p + ap;
                return p;
            }

            public bool ChangePrice()
            {
                string city1 = "";
                string city2 = "";
                float p = 0;
                if (Department1.ID == 0)
                {
                    city1 = Address1.Split(' ')[0];
                }
                Center cl1 = getCenterDepartmentFromPlace(city1);
                if (Department2.ID == 0)
                {
                    city2 = Address2.Split(' ')[0];
                }
                Center cl2 = getCenterDepartmentFromPlace(city2);
                float rst = getRouteDuration(cl1.DepartmentID,
cl2.DepartmentID);
                int zone;
                if (rst < 30)
                    zone = 0;
                else if (rst < 200)
                    zone = 1;
                else zone = 2;
                Tariff t = getTariff(zone, Weight);
                p = t.Price;
                float ap = t.AddPrice;
```

```
                if (Department1.ID == 0)
                    p = p + ap;
                if (Department2.ID == 0)
                    p = p + ap;
                Price = p;
                CalcTariff = t;
            }

            public DateTime CalcDate()
            {
                string city1 = "";
                string city2 = "";
                int res = 0;
                DateTime dt = Date1;
                if (Department1.ID == 0)
                {
                    city1 = Address1.Split(' ')[0];
                    res = res + 120;
                }
                Center cl1 = getCenterDepartmentFromPlace(city1);
                if (Department2.ID == 0)
                {
                    city2 = Address2.Split(' ')[0];
                }
                Center cl2 = getCenterDepartmentFromPlace(city2);
                List<float> rst = getRouteTime(cl1.DepartmentID,
cl2.DepartmentID);
                int dy = Date1.DayOfYear;
                int dyy = rst[0].DayOfYear;
                res = Date1.Hour*60+Date1.Minute;
                if ((dy % dyy == 0) && (res >=
(rst[0].Hour*60+rst[0].Minute)))
                {
                    dt = new DateTime (Date1.Year, Date1.Month,
Date1.Day, rst[0].Hour, rst[0].Minute,0);
                    dt.AddDays(dyy);
                }
                else if ((dy % dyy == 0) && (res <
(rst[0].Hour*60+rst[0].Minute)))
                {
                    dt = new DateTime (Date1.Year, Date1.Month,
Date1.Day, rst[0].Hour, rst[0].Minute,0);
                }
                else
                {
                    dt = new DateTime (Date1.Year, Date1.Month,
Date1.Day, rst[0].Hour, rst[0].Minute,0);
```

```
                    dt.AddDays(dy – (dy % dyy));
                }
                if (Department2.ID == 0)
                {
                    dt.AddHours(2);
                }
                return dt;
            }

            public bool ChangeDate()
            {
                Date2 = CalcDate();
                return true;
            }

            public event PropertyChangedEventHandler pceh;
            public void OnPropertyChanged([CallerMemberName]string
cmn = "")
            {
                if (pceh != null)
                    PropertyChanged(this, new
PropertyChangedEventArgs(cmn));
            }

            public bool OkCheck
            {
                get
                {
                    if (PTypeOk) != null)
                            return false;
                    else if (DTypeOk) != null)
                            return false;
                    else if (StatusOk) != null)
                            return false;
                    else if (TrackOk) != null)
                            return false;
                    else if (WeightOk) != null)
                            return false;
                    else if (Size1Ok) != null)
                            return false;
                    else if (Size2Ok) != null)
                            return false;
                    else if (Size3Ok) != null)
                            return false;
                    else if (LocationOk) != null)
                            return false;
                    else if (Address1Ok) != null)
```

```
                return false;
        else if (Address2Ok) != null)
                return false;
        else if (PriceOk) != null)
                return false;
        else if (Date1Ok) != null)
                return false;
        else if (Date2Ok) != null)
                return false;
        else if (CalcTariffOk) != null)
                return false;
        else if (Client1Ok) != null)
                return false;
        else if (Client2Ok) != null)
                return false;
        else if (Department1Ok) != null)
                return false;
        else if (Department2Ok) != null)
                return false;
        else if (PaidOk) != null)
                return false;
        return true;
    }
}

string GetValidationError(string pos)
{
        if (pos == "PackageType"))
                return PTypeOk();
        else if (pos == "DeliveryType")
                return DTypeOk();
        else if (pos == "DeliveryStatus")
                return StatusOk();
        else if (pos == "Track")
                return TrackOk();
        else if (pos == "Weight")
                return WeightOk();
        else if (pos == "SizeX")
                return Size1Ok();
        else if (pos == "SizeY")
                return Size2Ok();
        else if (pos == "SizeZ")
                return Size3Ok();
        else if (pos == "Location")
                return LocationOk();
        else if (pos == "SenderAddress")
                return Address1Ok();
```

```
        else if (pos == "RecAddress")
                return Address2Ok();
        else if (pos == "Price")
                return PriceOk();
        else if (pos == "SendDate")
                return Date1Ok();
        else if (pos == "RecDate")
                return Date2Ok();
        else if (pos == "Tariff")
                return CalcTariffOk();
        else if (pos == "Sender")
                return Client1Ok();
        else if (pos == "Receiver")
                return Client2Ok();
        else if (pos == "SenderDepartment")
                return Department1Ok();
        else if (pos == "RecDepartment")
                return Department2Ok();
        else if (pos == "Paid")
                return PaidOk();
}

public string PTypeOk()
{
    if (PType < 0 || PType > 3)
    {
        return "Невірно обрано тип пакунку";
    }
    return null;
}

public string DTypeOk()
{
    if (DType < 0 || DType > 4)
    {
        return "Невірно обрано тип пакунку ";
    }
    return null;
}

public string StatusOk()
{
    if (Status < 0 || Status > 13)
    {
        return "Даного статусу пакунку не існує";
    }
    return null;
```

```
}

public string TrackOk()
{
    if (Track.Count() == 0)
    {
        return "Трек-номер не встановлено";
    }
    else if (Track.Count() > 15)
    {
        return "Трек-номер не може бути даної довжини";
    }
    return null;
}

public string WeightOk()
{
    if(Weight < 0)
    {
        return "Вага не може бути від’ємною";
    }
    else if (Weight > 1000)
    {
        return "Вага більше 1000 кг недопустима";
    }
    return null;
}

public string Size1Ok()
{
    if(Size1 < 0)
    {
        return "Довжина не може бути від’ємною";
    }
    else if (Size1 > 300)
    {
        return "Довжина більше 300 см недопустима";
    }
    return null;
}

public string Size2Ok()
{
    if(Size2 < 0)
    {
        return "Ширина не може бути від’ємною";
    }
```

```
    else if (Size2 > 170)
    {
        return "Ширина більше 170 см недопустима";
    }
    return null;
}

public string Size3Ok()
{
    if(Size3 < 0)
    {
        return "Висота не може бути від’ємною";
    }
    else if (Size3 > 170)
    {
        return "Висота більше 170 см недопустима";
    }
    return null;
}

public string LocationOk()
{
    if (Location.Count() == 0)
    {
        return "Поточна локація має бути вказана";
    }
    else if (Location.Count() > 255)
    {
        return "Занадто довге значення";
    }
    return null;

}

public string Address1Ok()
{
    if (Address1.Count() > 255)
    {
        return "Занадто довге значення";
    }
    return null;

}

public string Address2Ok()
{
    if (Address2.Count() > 255)
```

```
                {
                    return "Занадто довге значення";
                }
                else if (Address2 == Address1 && !
String.IsNullOrEmpty(Address2))
                {
                    return "Адреси не можуть співпадати";
                }
                return null;

            }

            public string PriceOk()
            {
                if(Price <= 0)
                {
                    return "Вартість не може бути від’ємною";
                }
                return null;
            }

            public string Date1Ok()
            {
                if(Date1 > Date2)
                {
                    return "Дата і час прибуття пакунку вказані
раніше за відправлення";
                }
                return null;
            }

            public string Date2Ok()
            {
                if(Date2 < Date1)
                {
                    return "Дата і час прибуття пакунку вказані
раніше за відправлення";
                }
                return null;
            }

            public string CalcTariffOk()
            {
                if (CalcTariff < 0 || CalcTariff > getMaxTariffId())
                {
                    return "Обрано невірний тарифний план";
                }
```

```
                return null;
            }

            public string Client1Ok()
            {
                if (Client1 < 0 || Client1 > getMaxClientId())
                {
                    return "Такого клієнта не існує";
                }
                else if (Client2 == Client1 && Address1 == Address2
&& ! String.IsNullOrEmpty(Address1))
                {
                    return "Не можна відправити пакунок самому
собі";
                }
                else if (Client2 == Client1 && Department1 ==
Department2 &&  String.IsNullOrEmpty(Address1))
                {
                    return "Не можна відправити пакунок самому
собі";
                }
                return null;
            }

            public string Client2Ok()
            {
                if (Client2 < 0 || Client2 > getMaxClientId())
                {
                    return "Такого клієнта не існує";
                }
                else if (Client1 == Client2 && Address1 == Address2
&& ! String.IsNullOrEmpty(Address1))
                {
                    return "Не можна відправити пакунок самому
собі";
                }
                return null;
            }

            public string Department1Ok()
            {
                if (Department1 < 0 || Department1 >
getMaxDepartmentId())
                {
                    return "Такого відділення не існує";
                }
```

```
                else if (Department2 == Department1 &&
String.IsNullOrEmpty(Address1))
                {
                    return "Не можна відправити пакунок самому
собі";
                }
                return null;
            }

            public string Department2Ok()
            {
                if (Department2 < 0 || Department2 >
getMaxDepartmentId())
                {
                    return "Такого відділення не існує";
                }
                else if (Department1 == Department2 &&
String.IsNullOrEmpty(Address2))
                {
                    return "Не можна відправити пакунок самому
собі";
                }
                return null;
            }

            public string PaidOk()
            {
                if(Paid <= 0)
                {
                    return "Сума не може бути від’ємною";
                }
                else if (Paid > Price)
                {
                    return "Внесена сума не може бути більшою за
розраховану вартість";
                }

                return null;
            }
        }}
```

ДОДАТОК В

Слайди презентації

Рисунок В.1 – Слайд № 1

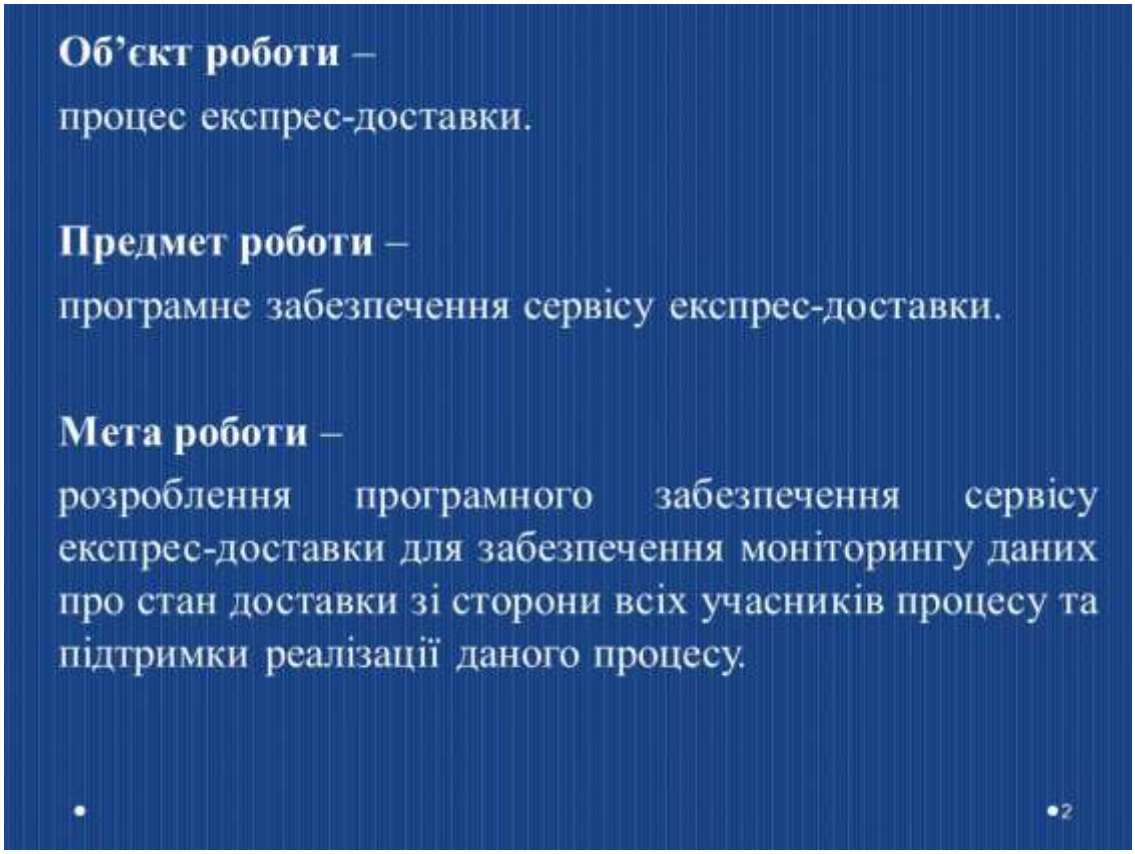

Рисунок В.2 – Слайд № 2

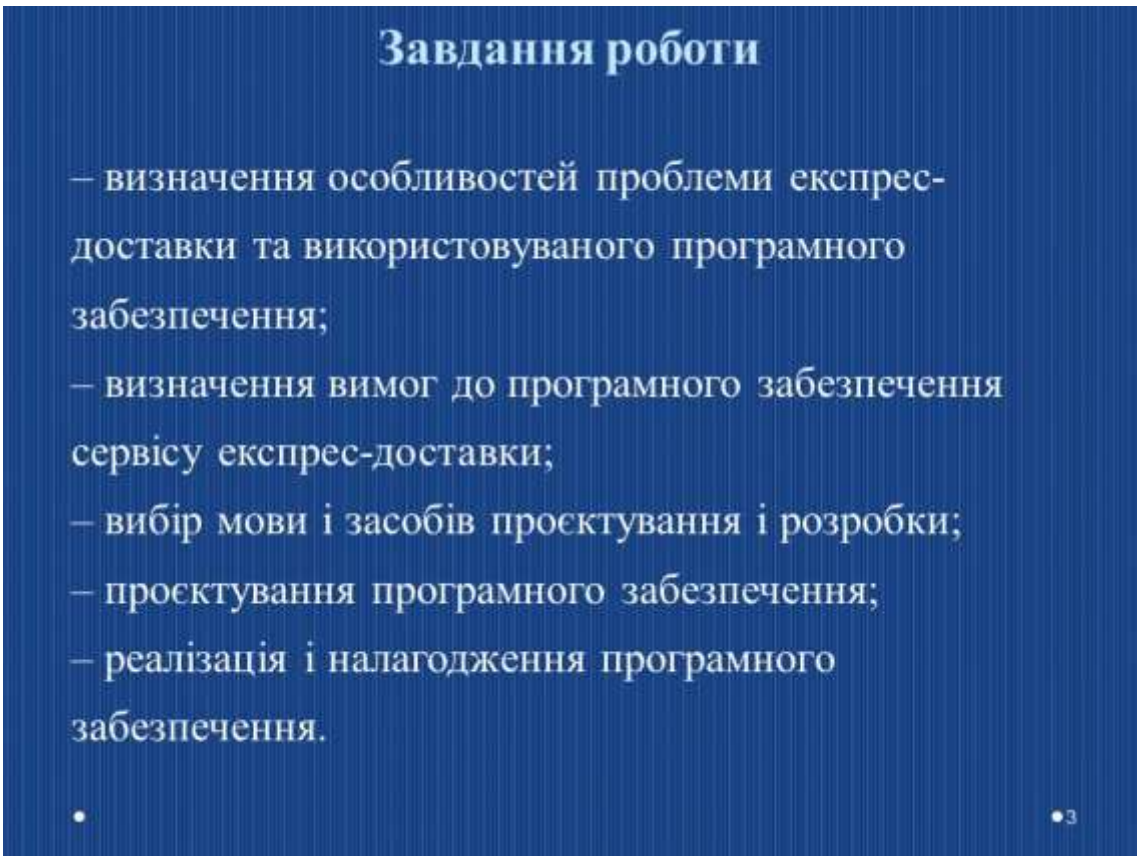

Рисунок В.3 – Слайд № 3

Рисунок В.4 – Слайд № 4

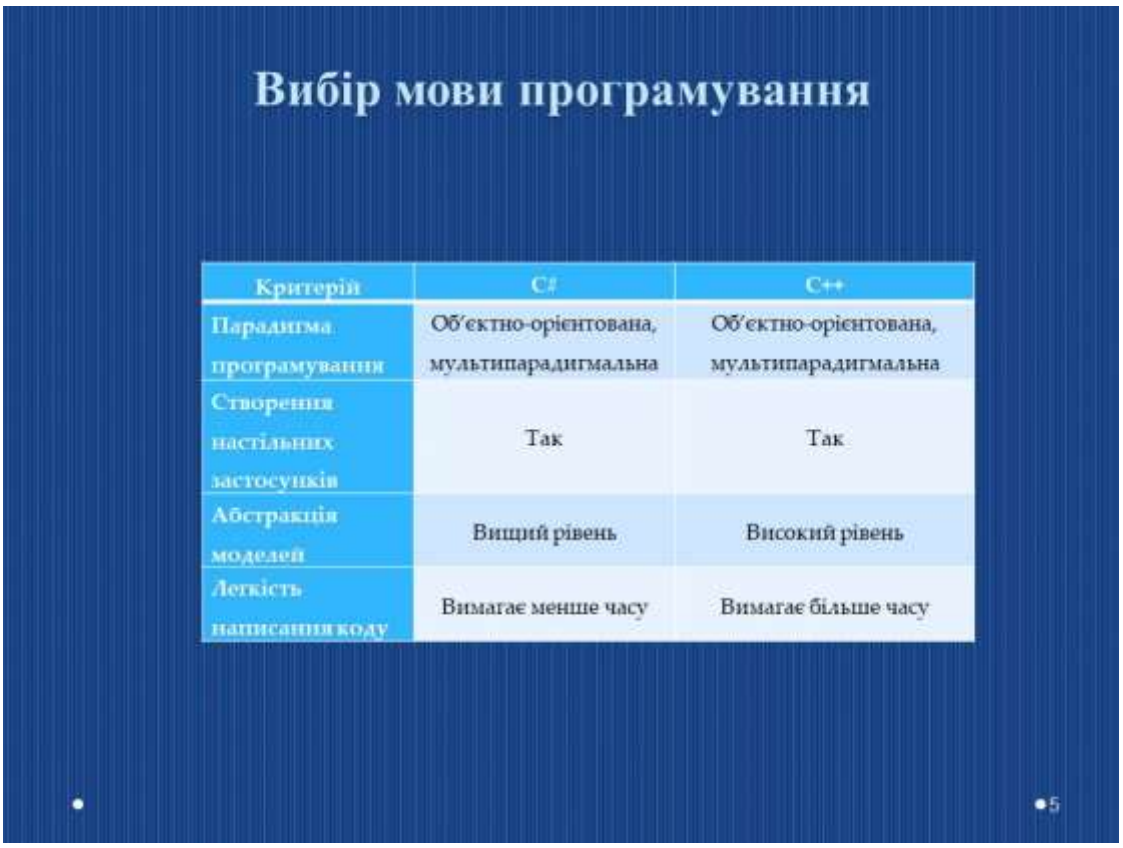

Рисунок В.5 – Слайд № 5

Рисунок В.6 – Слайд № 6

Рисунок В.7 – Слайд № 7

Рисунок В.8 – Слайд № 8

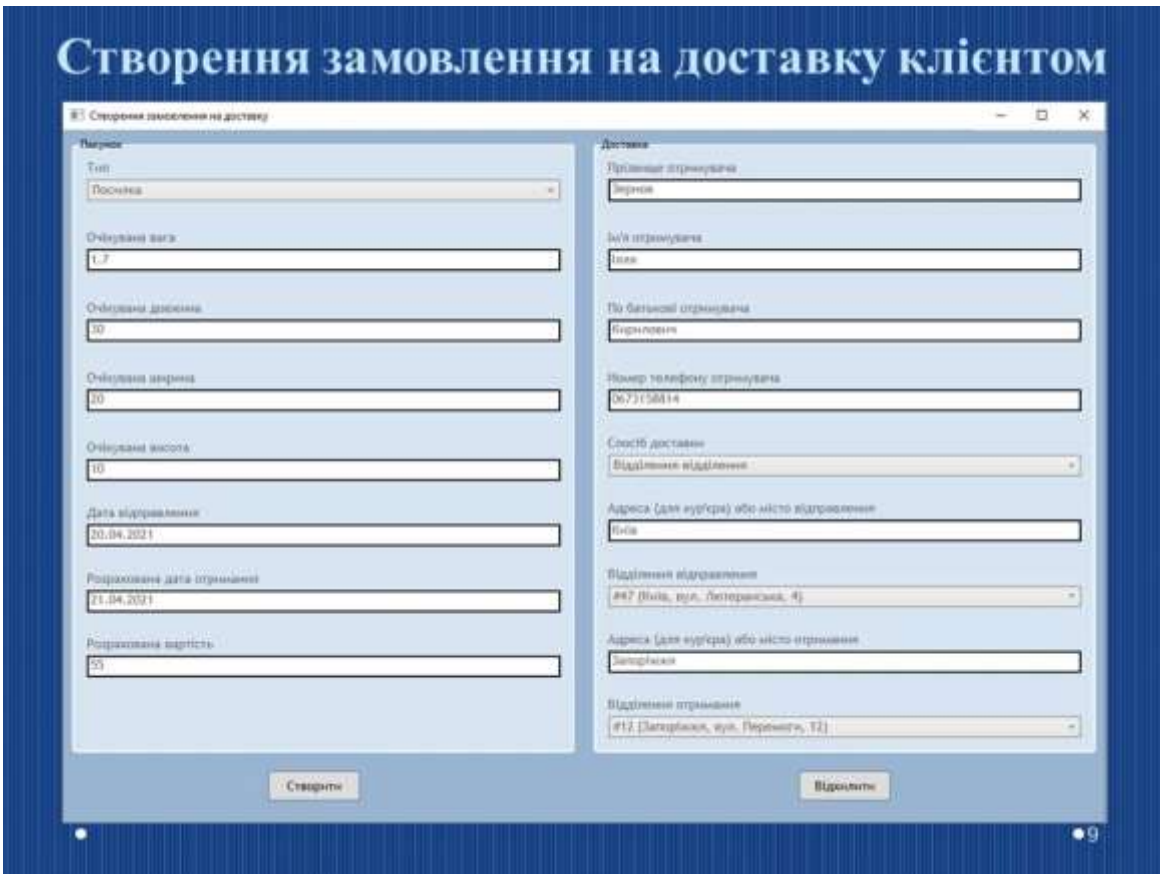

Рисунок В.9 – Слайд № 9

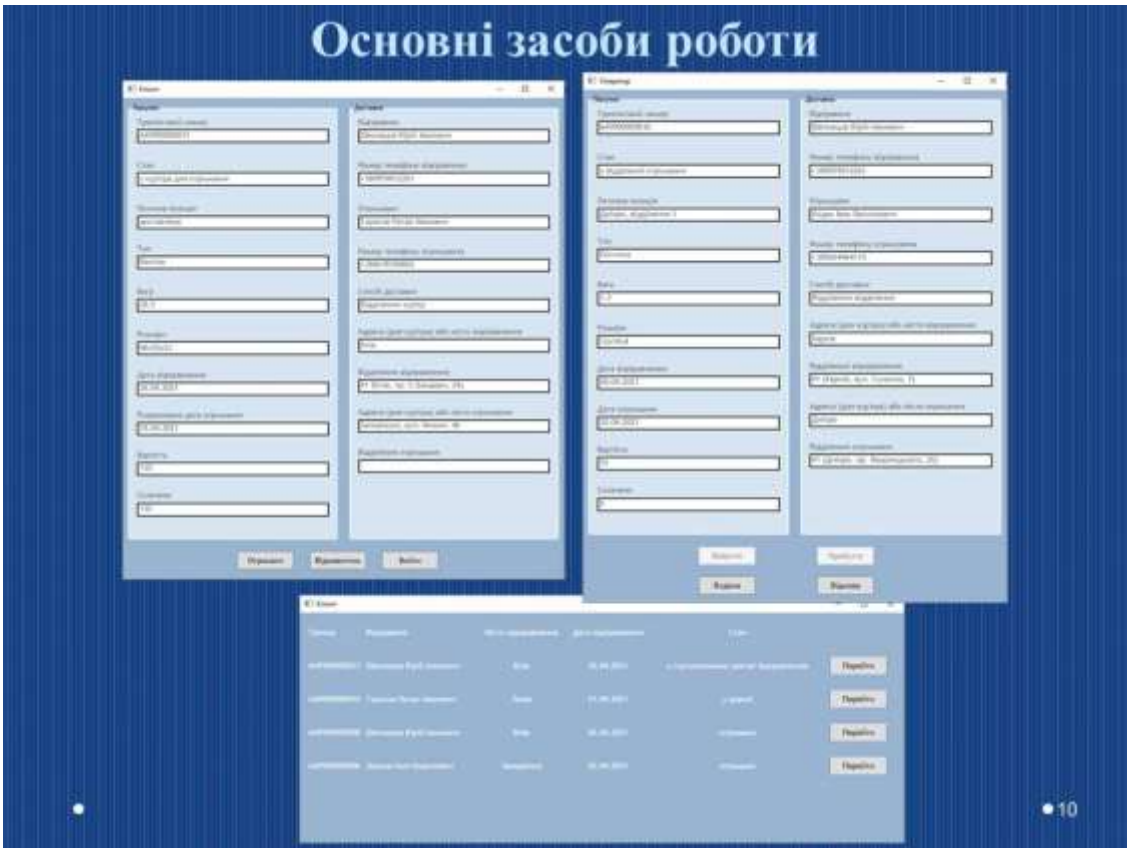

Рисунок В.10 – Слайд № 10

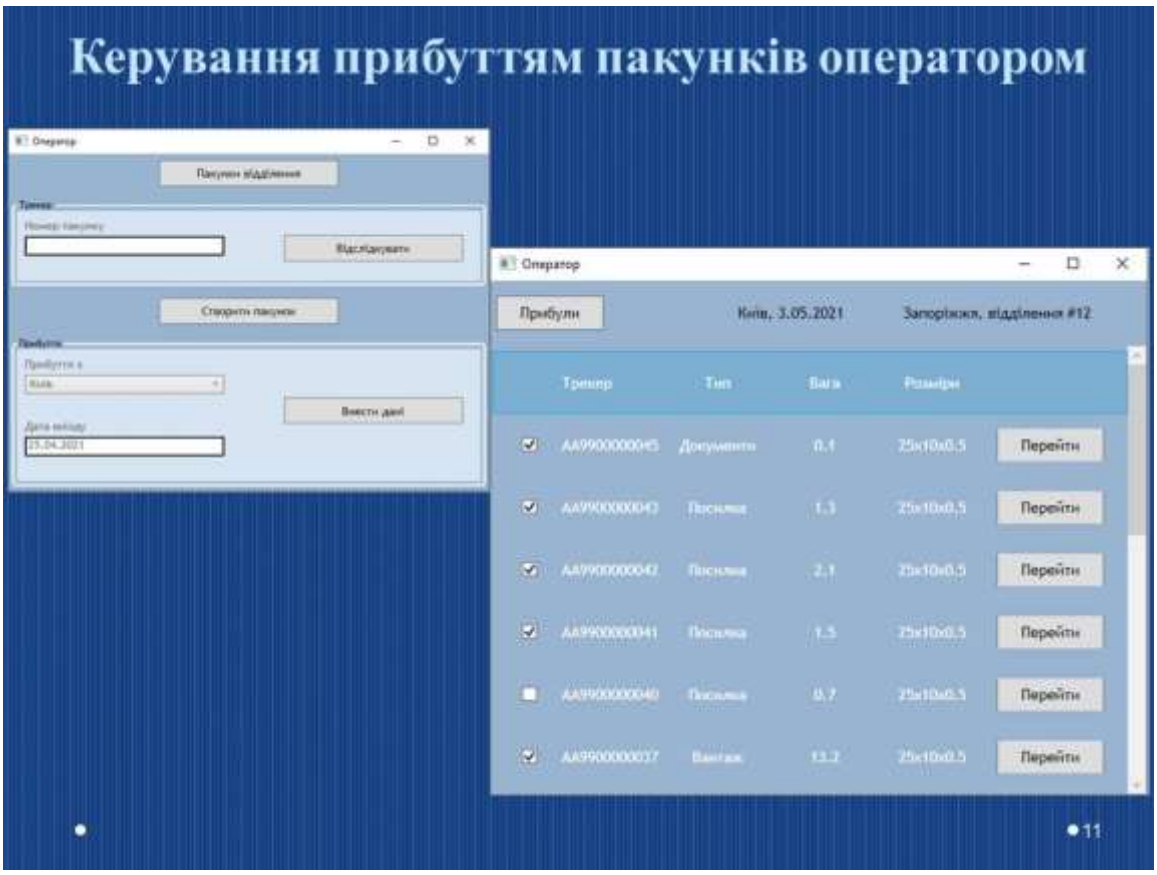

Рисунок В.11 – Слайд № 11

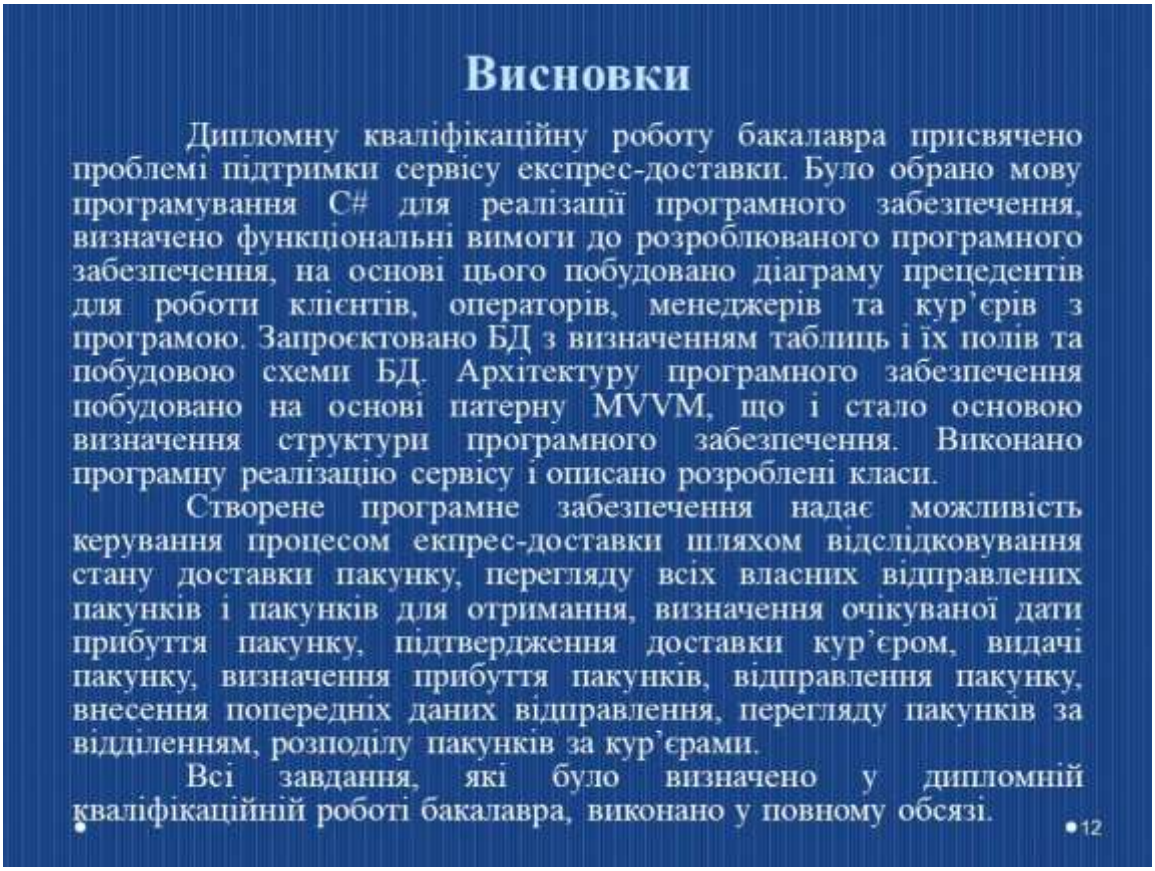

Рисунок В.12 – Слайд № 12