НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕЇНІКИ

ЛАБОРАТОРНА РОБОТА 1

з дисципліни:

«Сучасні методології і технології розробки програмного забезпечення»

на тему:

«Багатошарова архітектура програмних додатків. Використання системи контролю версій. Основи UML.»

Студента 3 курсу групи ІТ-81
Венделовського Івана Сергійовича

Кількість балів:_____ Оцінка_____

Викладач:_____к.т.н. Штифурак Юрій Михайлович
(оцінка)

Київ – 2021

ЗМІСТ

# ВСТУП

В даній роботі ми розглянемо розробку додатку універсальної системи доставок. Розроблювана нами система може застосовуватись для організації перевезень вантажів для будь-яких цілей.

Розгляд додатку почато з опису вимог та особливостей котрим повинен відповідати проект. У розділі постановка задачі було описано загальне призначення додатку та конкретні функціональні вимоги до нього.

В наступному розділі ми конкретизували дії котрі будуть доступні користувачу при взаємодії з додатком.

Далі в розділі розробка діаграми класів ми описали ідею та принцип роботи всього проекту, а також його архітектуру.

Наступним кроком з допомогою діаграми послідовностей ми конкретизували порядок взаємодії між класами для відпрацювання запитів користувача з веб сторінок.

В розділі діаграма станів було представлено та описано всі стани в котрих може перебувати додаток.

В наступному розділі представлена діаграма компонентів та з її допомогою описана взаємодія розроблених нами компонентів між собою їх взаємодія з зовнішніми інфраструктурними компонентами.

Завершує дану роботу розділ висновки де було підсумовано результат розробки, проаналізовано ступінь досягнення мети, звернуто увагу на важливі моменти описані на протязі всієї роботи та вказано можливі подальші напрямки розвитку розробки.

# 1 ПОСТАНОКА ЗАДАЧІ

Необхідно розробити базу даних та веб додаток для системи контролю доставок у репозиторії «https://github.com/VINIPOOH/delivery_dot_net».

1) Система повинна надавати можливість зареєстрованим користувачам формувати замовлення доставки вантажу та підтверджувати факт отримання адресованих йому доставок.

2) Також за регістрований користувач повинен мати можливість у особистому кабінеті поповнити свій грошовий рахунок та оплатити рахунки чеки що були сформовані для оплати створених ним заявок. Підтвердити отримання доставки можна тільки після факту оплати відправником доставки.

3) Користувач може відправити доставку сам собі, оскільки існує можливість використання служби доставок за для перевезення власних речей.

4) Доставка повинна містити інформацію про вагу, місто призначення, місто відправки та ідентифікатор користувача-адресата.

5) Після створення заявки на доставку система автоматично повинна сформувати рахунок для її оплати і почати відображати його у кабінеті користувача.

6) Не авторизованому користувачу повинна бути доступна сторінка розрахунку вартості та часу необхідних на доставку з заданими параметрами. Для розрахунку цієї інформації повинен проводитись при кожному зверненні клієнта по діючій на момент запиту тарифікації.

8) Вартість доставок повинна визначатись та залежати від відстані та вагового діапазону у який потрапляє доставка. Для кожного маршруту можуть бути встановлені різні вагові діапазони та додаткова плата за кілометр залежно від вагового діапазону.

# 1 РОЗРОБКА ДІАГРАМИ ВАРІАНТІВ ВИКОРИСТАННЯ

## 1.1 Опис та діаграми використання

Як показано на рисунку 1.1 є два види користувачів, авторизовані та не авторизовані. Не авторизований користувач може користуватись тільки функцією логіну, регістрації та калькуляції умов доставки. Аторизованому користувачу доступний повний функціонал додатку, що включає в себе перегляд та поповнення грошового балансу, перегляд історії платежів, замовлення, оплату та підтвердження доставок.

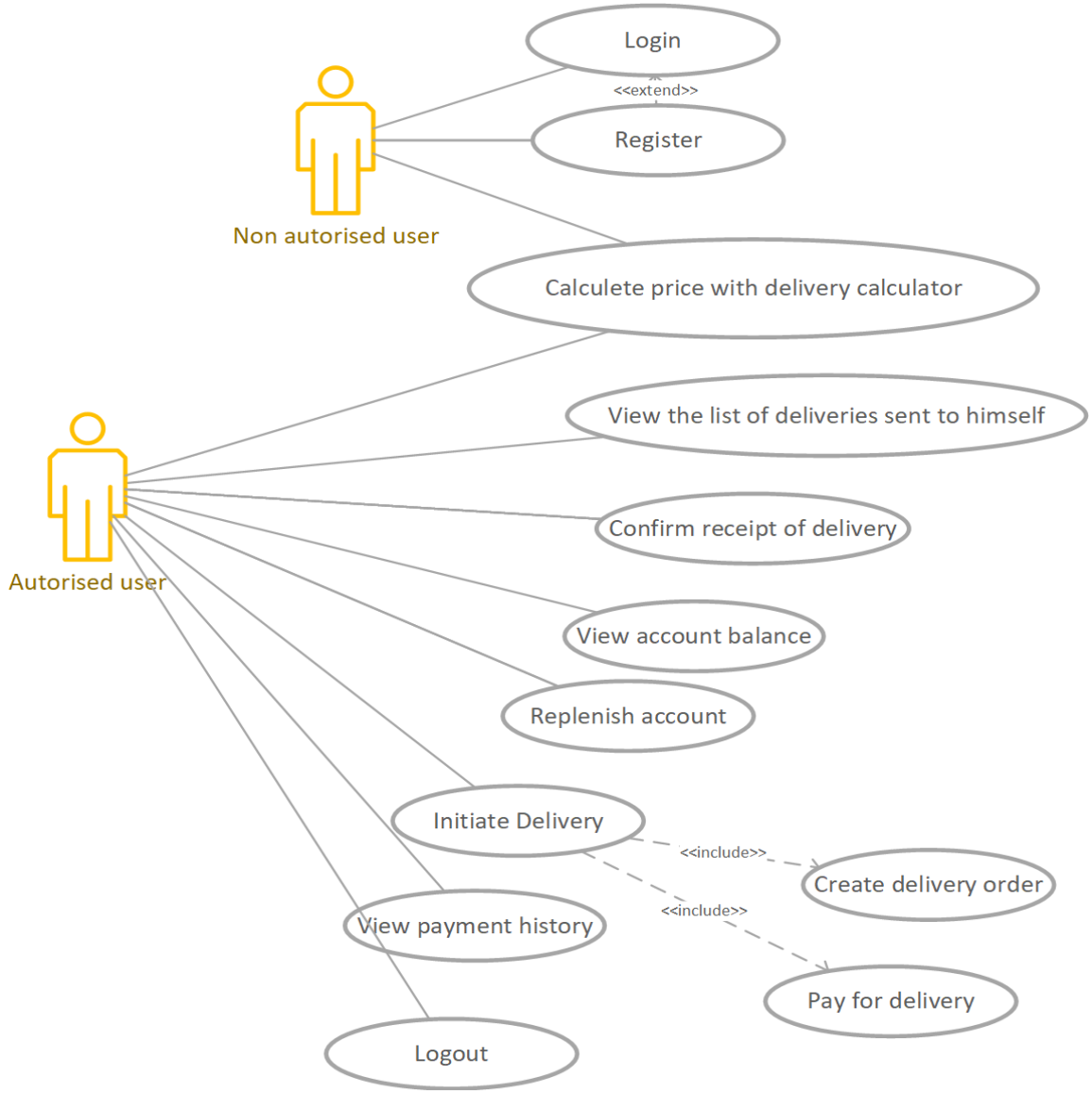


Рисунок 1.1 – Діаграма використання додатку.

# 2 РОЗРОБКА ДІАГРАМИ КЛАСІВ

## 2.1 Опис та структуризація процесу розробки діаграми класів.

Для забезпечення масштабованості веб додаток повинен бути поділений щонайменше на три шари: Шар роботи з базою данних, шар бізнес логіки, та шар контролерів.[9] Проектування класів слід проводити у напрямку від бази даних і закінчуючи рівнем веб преставлення.[9] Таким чином спершу ми спроектуємо діаграму класів для рівня доступу до даних. Наступним кроком ми спроектуємо шар бізнес логіки. Завершимо проектування шаром контролерів.

В звіті представлено частини діаграми класів повна діаграму класів можна знайти в папці додатків в GitHub репозиторії.

## 2.2 Опис рівня доступу до даних.

Проектування рівня доступу до даних почнемо з створення так званих класів сутностей. Вони будуть в подальшому застосовуватися для об’єктно орієнтованого відображення сутностей бази даних для подальшої роботи з ними. Для цього відображення створимо по класу у відповідність кошній таблиці, з виключенням таблиць що організовують зв'язок багато до багатьох. Та створимо в них властивості котрі будуть відповідати атрибутам відповідних таблиць у базі даних. Результат даних операцій представлено на рисунку 2.1.

Далі слід визначити класи котрі будуть керувати (давати доступ до сутностей). Ці класи шару доступу до даних прийнято називати репозиторіями. Вони відповідають за відправку запитів до бази даних та перетворення отриманих даних з реляційного відображення у об’єктно орієнтоване. Репозиторії верхнього рівня формують програмний інтерфейс шару доступу до бази даних. Задля забезпечення принципу єждиної відповідальності для роботи з кожною сутністю було зтворено окремий репозиторій. На рисунку 2.2 показано взаємодію репозиторіїв та сутностей з якими вони пов’язані.

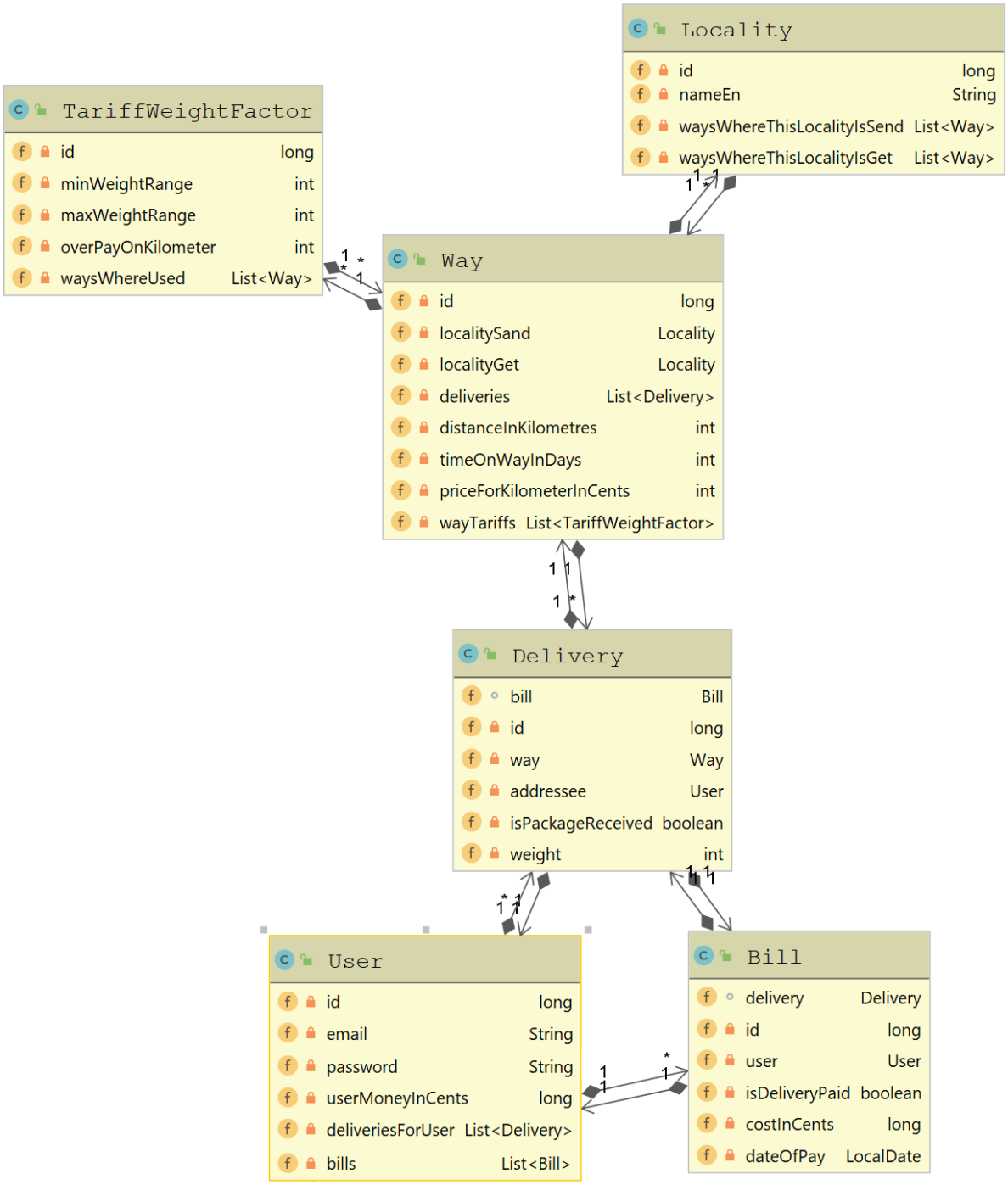


Рисунок 2.1 – Діаграма класів, що представляє сутності.

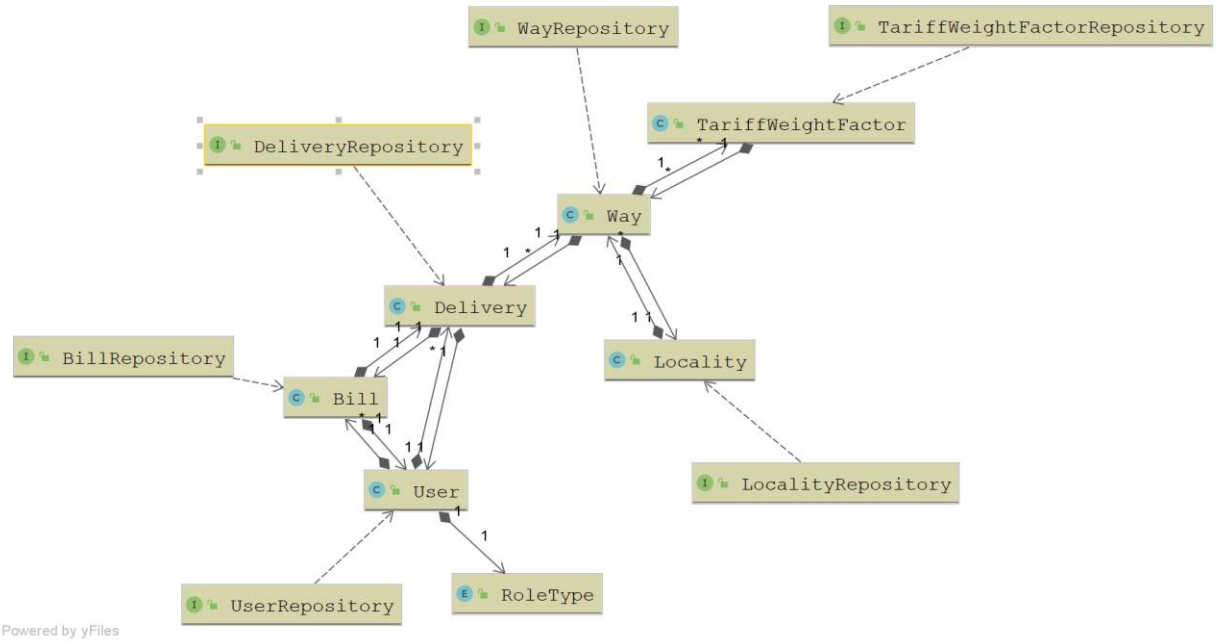


Рисунок 2.2 – Діаграма класів що демонструє зв’язки між репозиторіями та сутностями.

2.3 Опис рівня бізнес логіки та його взаємодії з рівнем доступу до даних.

Саме у шарі бізнес логіки реалізовується вся поведінка веб додатку. Класи, що її реалізовують називаються сервіси. Інтерфейси сервісів верхнього рівня утворюють програмний інтерфейс шару бізнес логіки. Для забезпечення себе даними бізнес шар звертається до інтерфейсів репозиторіїв верхнього рівня. Класи сутності виступають транспортними засобами для передачі інформації від рівня доступу до даних до рівня сервісу. На рисунку 9.3 представлено програмний інтерфейс сервісного рівня, класи сервісного рівня та взаємодію класів сервісного рівня з інтерфейсом рівня доступу до даних.

2.4 Опис рівня контролерів та їх зв’язку з програмним інтерфейсом рівня бізнес логіки.

Завданням шару контролерів є отримання даних від клієнта, перевірка їх на правильність та передача запиту у випадку його коректності бізнес логіці. Далі

отримавши результат від бізнес логіки рівень контролерів формує відображення для клієнта результату обробки запиту.

Для отримання та передачі цієї інформації застосовуються спеціальні класи без поведінки.

На рисунку 9.4 Представлено рівень контролерів та його взаємодію з програмним інтерфейсом рівня бізнес логіки. На рисунках 9.3 та 9.4 видно що шари взаємодіють через інтерфейси. Це необхідно для забезпечення принципу інверсії залежностей «SOLID»[1].

Цей принцип вимагає від об’єктно орієнтованих програм будувати залежності всередині системи на основі абстракцій, що не залежать від деталей, а деталі залежати від абстракцій. Це дозволяє в результаті отримати архітектуру в якій модулі вищих рівнів не залежать від модулів нижчих рівнів.

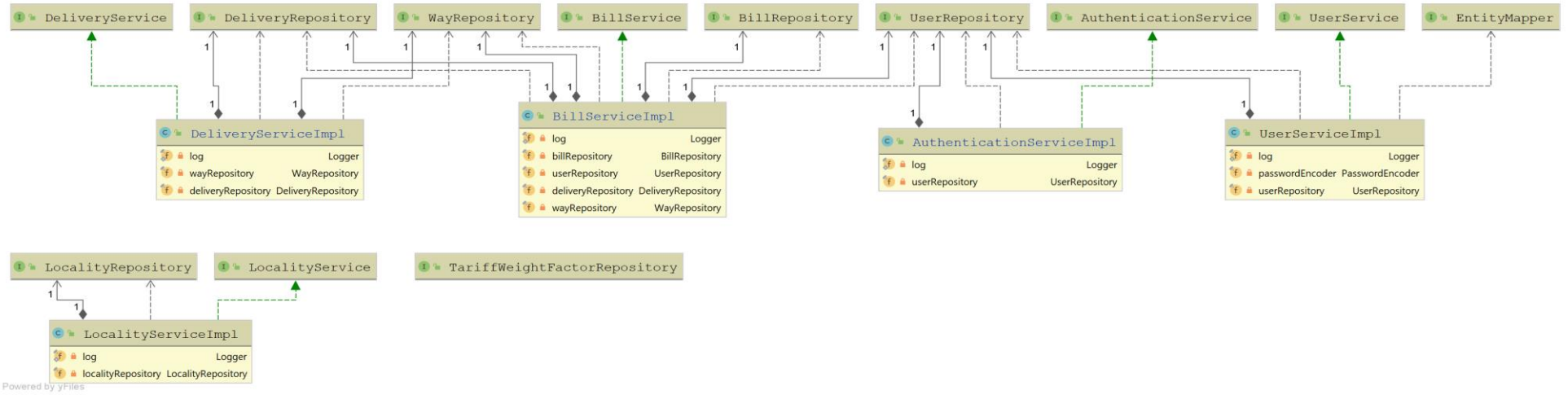


Рисунок 2.3 – Діаграма зв’язків між сервісами та репозиторіями.

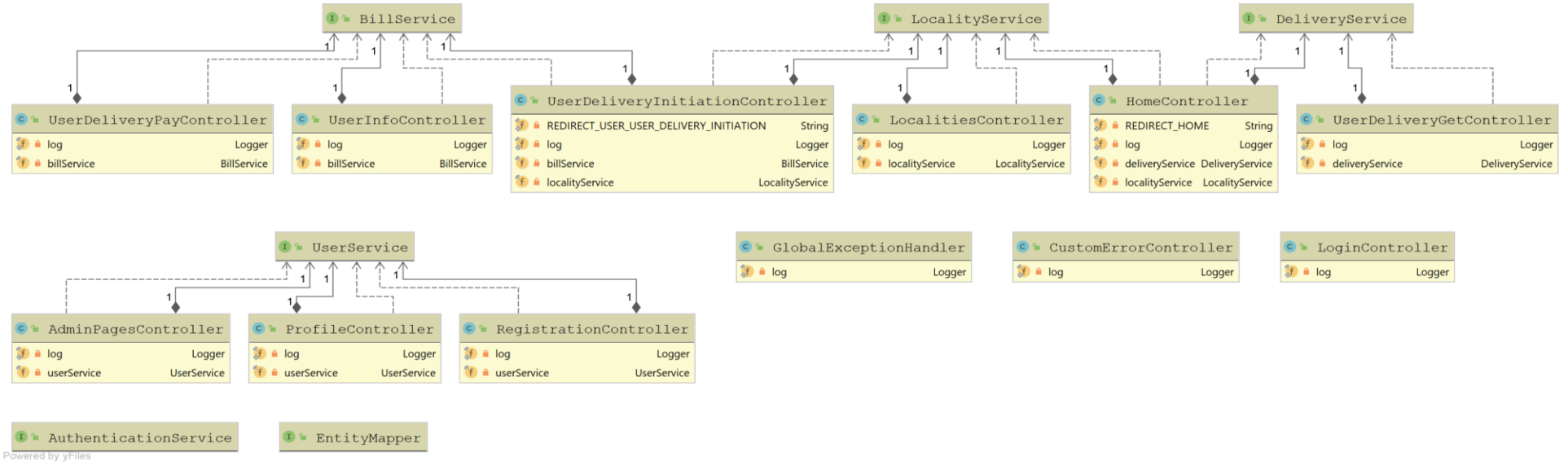


Рисунок 2.4 – Діаграма зв’язків між контролерами та сервісами.

# 3 РОЗРОБКА ДІАГРАМИ ПОСЛІДОВНОСТЕЙ

## 3.1 Опис діаграм

Оскільки додаток побудовано з допомогою тришарової безстанової архітектури діаграму використання ми розіб’ємо на логічні частини за сторінками. Оскільки кожній сторінці відповідає один контролер, котрий в свою чергу і є входом в наскрізний безстановий логічний блок обробки даних.

На рисунку 3.1 представлено діаграму послідовностей для сторінки створення замовлення. Далі на рисунках 3.2 та 3.3 представлено діаграми послідовностей для оплати та підтвердження отримання доставки відповідно.

На рисунку 3.4 представлено діаграму для сторінки регістрації в систему.

На рисунку 3.5 зображена діаграма для сторінки калькуляції.

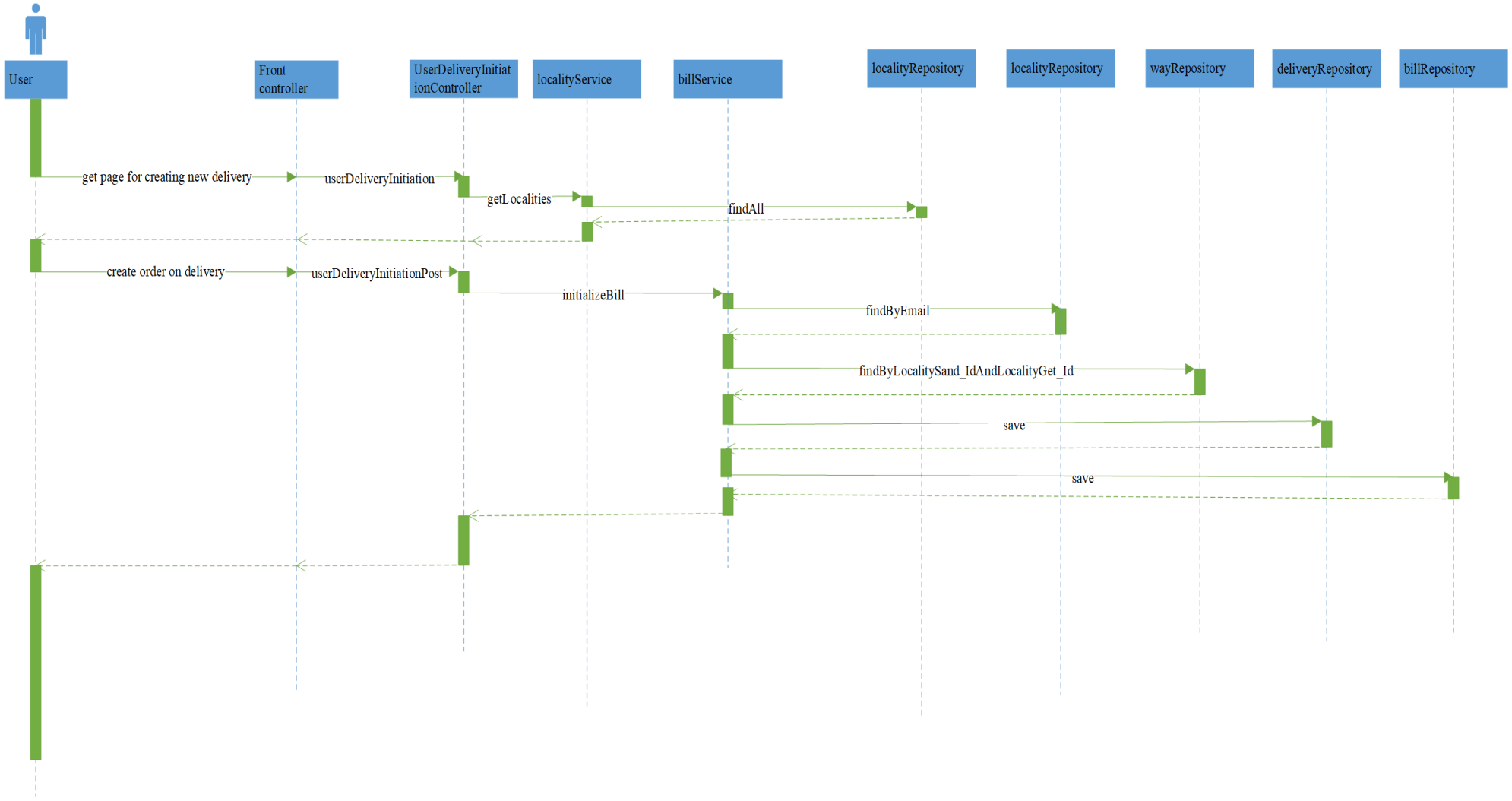


Рисунок 3.1 - діаграма послідовностей сторінки створення замовлення.

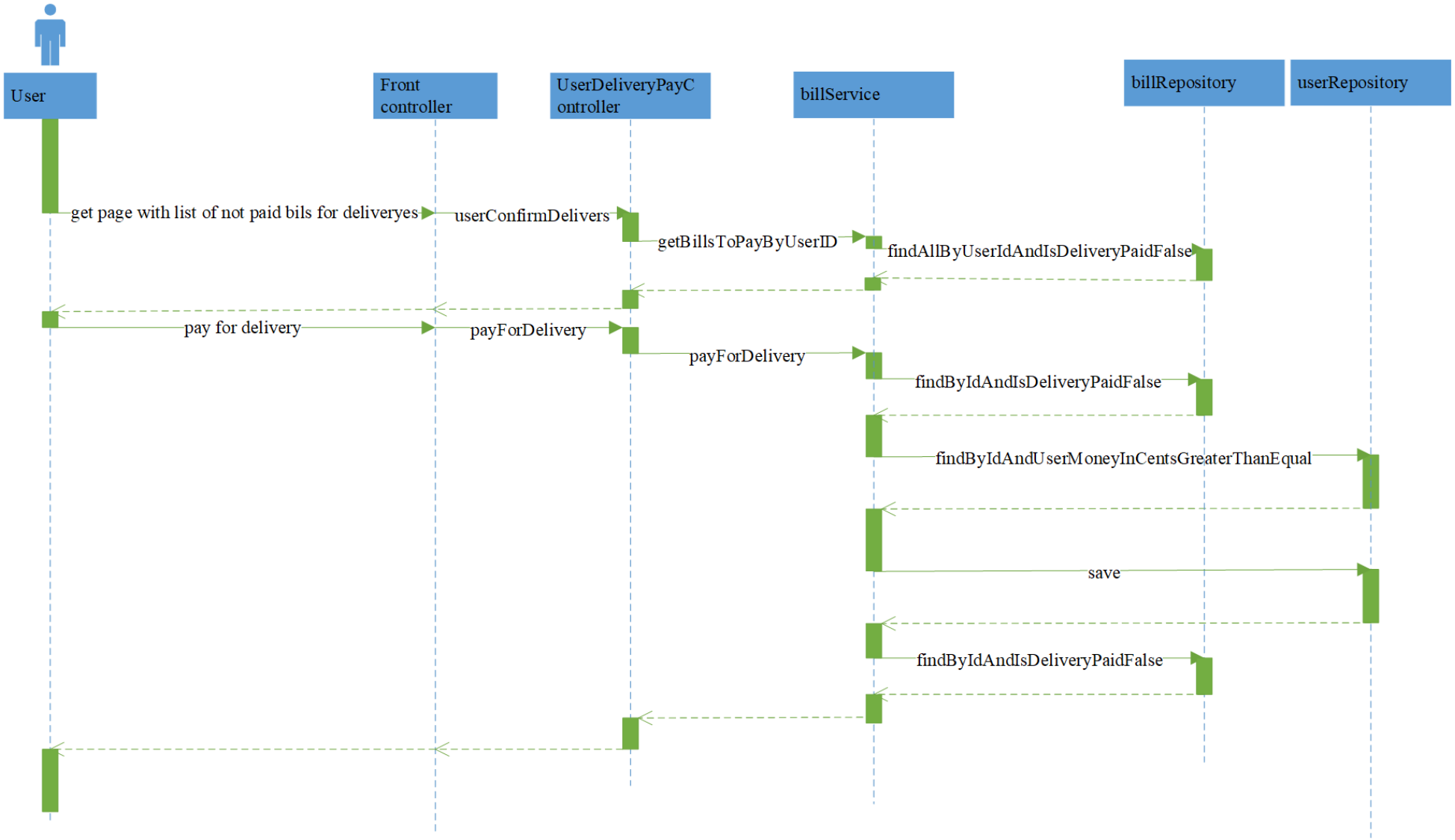


Рисунок 3.2 - діаграма послідовностей сторінки оплати замовлення.

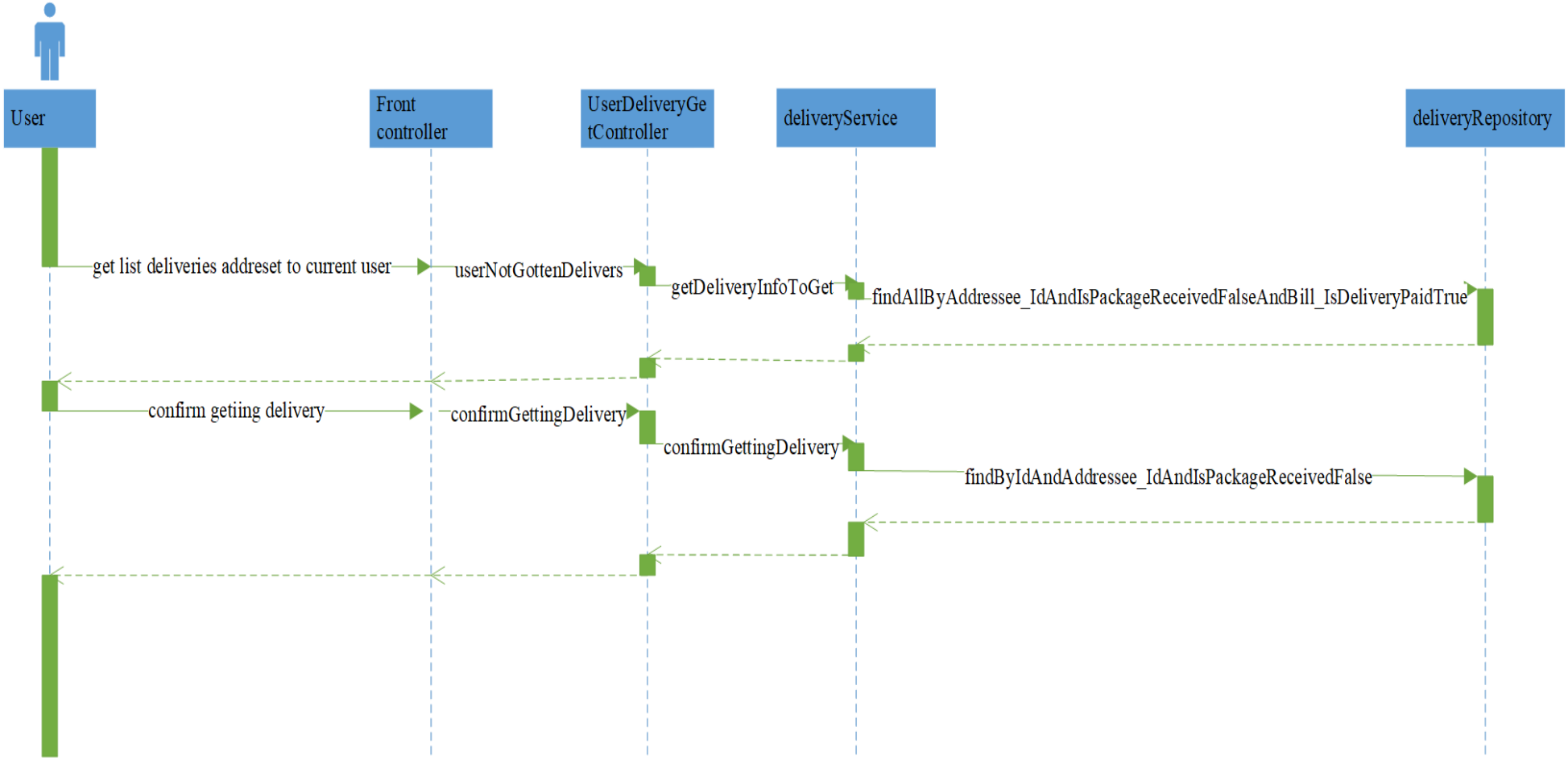


Рисунок 3.3 - діаграма послідовностей сторінки підтвердження отримання замовлення.

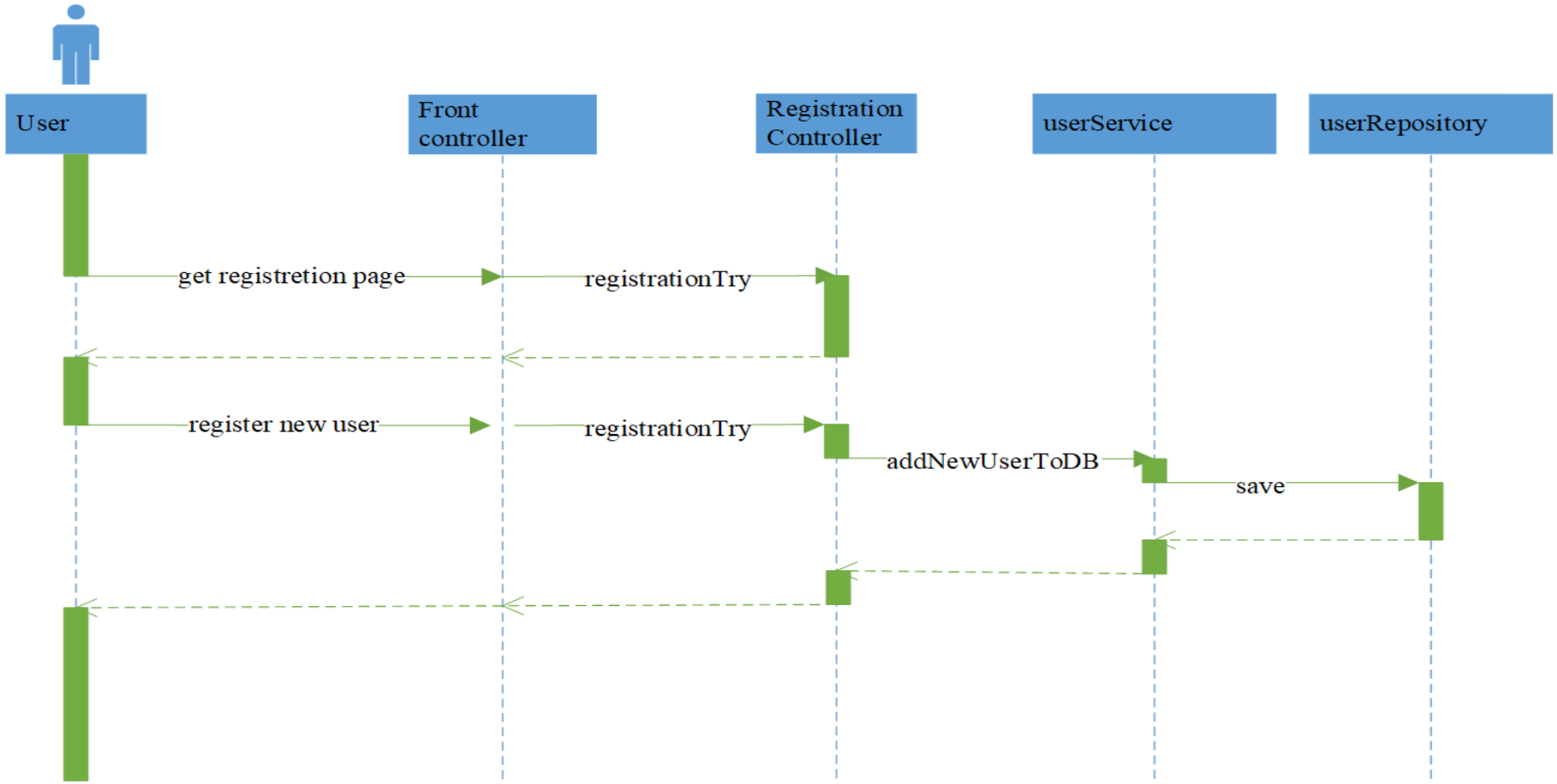


Рисунок 3.4 - діаграма послідовностей сторінки регістрації користувача.

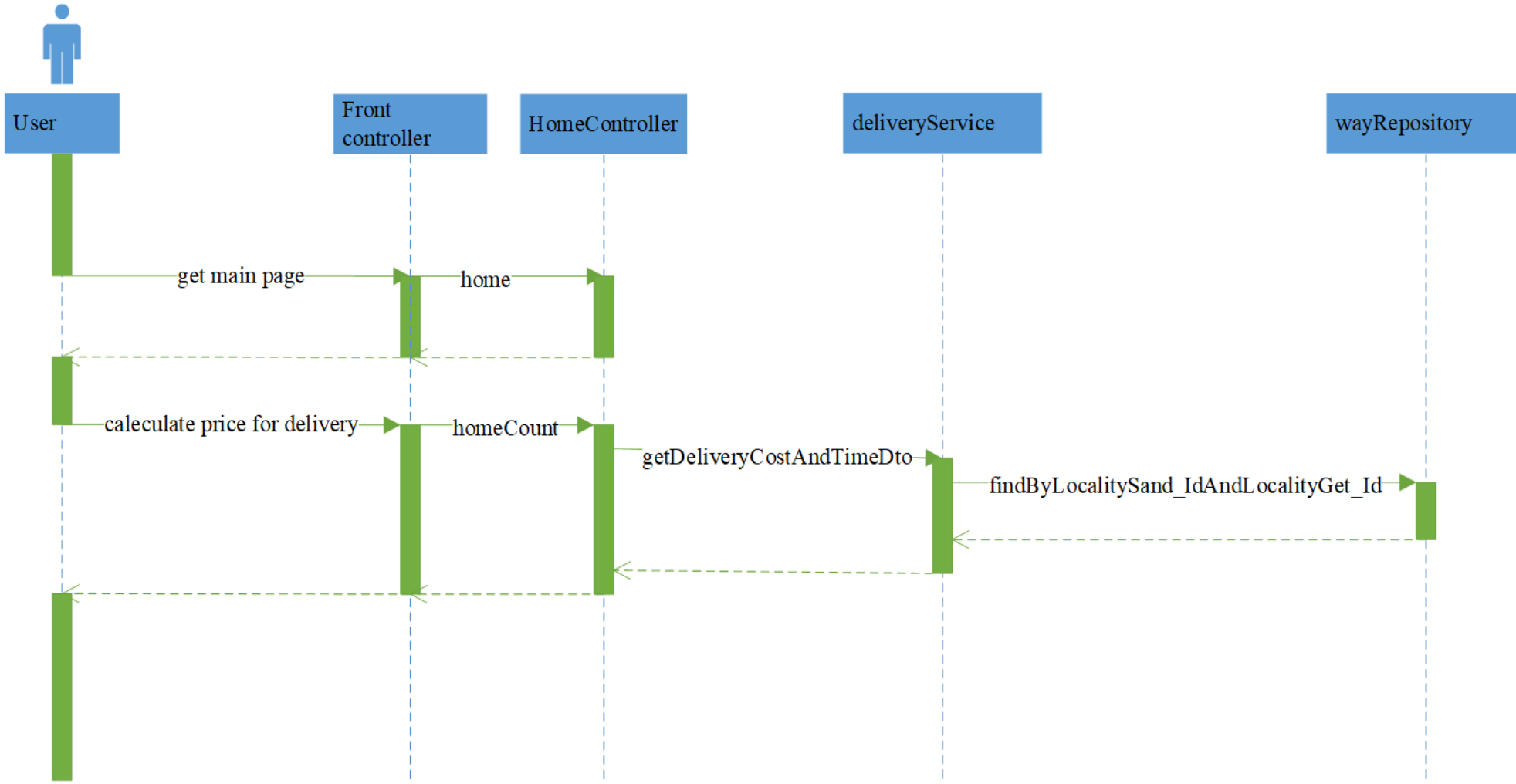


Рисунок 3.5 - діаграма послідовностей головної сторінки.

# 4 РОЗРОБКА ДІАГРАМИ СТАНІВ

## 4.1 Опис діаграм

Додаток було спроектовано з використанням тришарової архітектури та «statles» підходу. Таким чином додаток немає станів вся інформація зберігається виключно у базі даних. Це дозволяє уникнути проблем в багато поточному режимі оскільки безстанові елемент є потокобезпечними. І зробити додаток абсолютно толерантним до зупинок в роботі з будь-яких причин.

Додаток має виключно стани пов’язані з початком та завершенням роботи серверу. Ці стани представлено на рисунку 4.1.

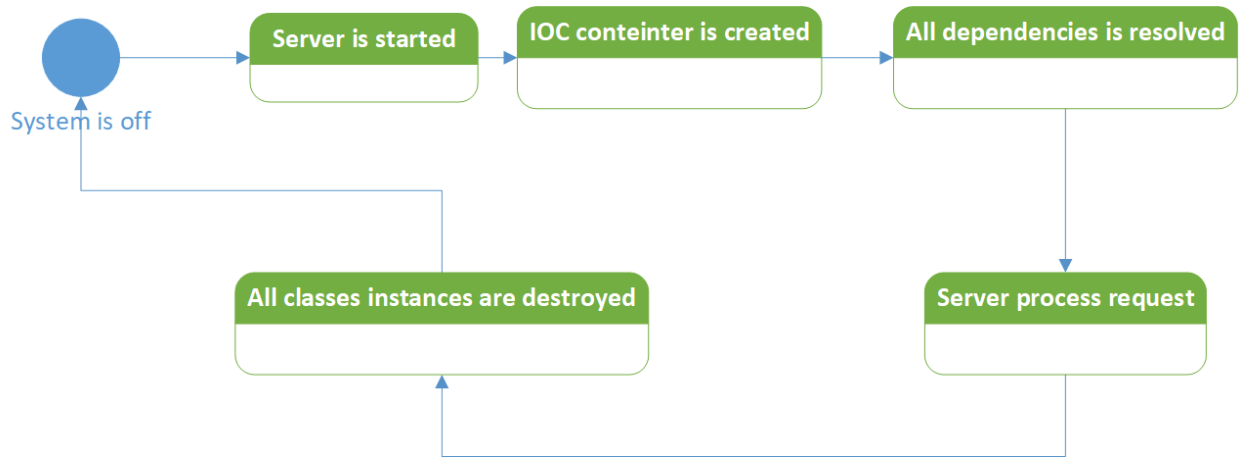


Рисунок 4.1 – Діаграма станів додатку.

# 5 РОЗРОБКА ДІАГРАМИ КОМПОНЕНТІВ

## 5.1 Опис діаграми компонентів

На рисунку 5.1 зображено компоненти основних шарів додатку, а саме шару контролерів котрий диктує API додатку, шару бізнес логіки та шару доступу до даних. Клієнтом виступає браузер. Він взаємодіє з сервером через API рівня контролерів.

Шар доступу до даних зв’язується з базою даних через інтерфейс посередника «.NET Core for Windows». Це універсальний драйвер для роботи з реляційними базами даних, він дозволяє абстрагуватись від конкретної СУБД і надає можливість у випадку необхідності замінити базу даних.

«AccountController» використовує «Microsoft.Aspnet.Identity» як провайдер автентифікації та авторизації користувачів. З допомогою нього акаунт контролер реалізує логіку контролю доступу до API користувачів, а також надає функціонал регістрації нових користувачів.

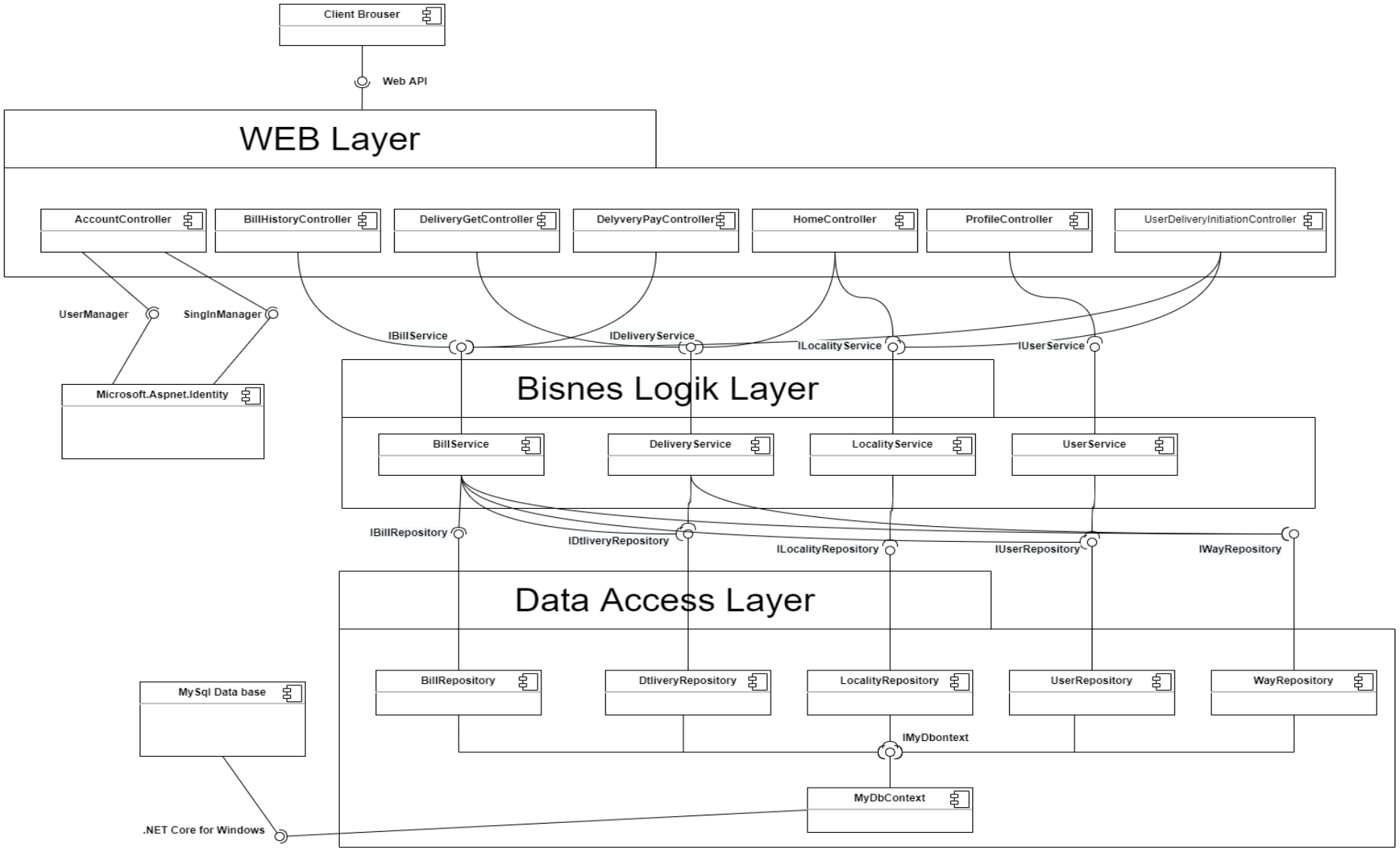


Рисунок 5.1 – Діаграма компонентів додатку.

# ВИСНОВКИ

В дані роботі успішно створено модель додатку згідно всіх поставлених вимог.

Завдяки застосування платформо незалежних інструментів нам вдалось досягти кросплатформеності розробленого додатку.

Створений додаток є гнучким для розширення, це було досягнуто за рахунок застосування шаблонів проектування, ООП та організацій контролю інверсії залежностей «ASP net».

В процесі виконання завдання було закріплено знання з проектування діаграм класів, прецедентів, станів послідовностей та компонентів.

Результат розроби даної роботи в повному вигляді було представлено в репозиторії за посиланням «https://github.com/VINIPOOH/delivery_dot_net».

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Ноубл, Дж., Андерсон, Т., Брэйтуэйт, Г., Казарио, М., Третола, Р. Flex 4. Рецепты программирования. — БХВ-Петербург, 2011. — С. 548. — 720 с

2) Самоучитель UML 2. – СПб.: БХВ-Петербург, 2007. – 567 с.: ил. ISBN 978-5-94157-878-8

3) Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. П75 Приемы объектно-ориентированного проектирования. Паттерны проектирования. — СПб: Питер, 2001. — 368 с.: ил. (Серия «Библиотека программиста») ISBN 5-272-00355-1

4) Мартин Фаулер., Чистий код: создание, анализ и рефакторинг. – СПб.: Питер, 2019. – 464 с.: ил.