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

ЛАБОРАТОРНА РОБОТА 3

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

1 ПОСТАНОКА ЗАДАЧІ

1) Реалізувати розроблені на попередньому етапі роботи інтерфейси.
2) Реалізувати усі необхідні для стабільної роботи ПЗ класи та налагодити їх взаємодію відповідно до SOLID принципів.

1 РЕАЛІЗАЦІЯ РОЗРОБЛЕНИХ НА ПОПЕРЕДНЬОМУ ЕТАПІ ІНТЕРФЕЙСІВ

1.1Реалізація інтерфейсів шару бізнес логіки

1.1.1 Реалізація інтерфейсу «IBillSevice»

```
public class BillService : IBillService
{
    private readonly IBillRepository _billRepository;
    private readonly IUserRepository _userRepository;
    private IDeliveryRepository _deliveryRepository;
    private IWayRepository _wayRepository;
```

Рисунок 1.1 – Залежності реалізації інтерфейсу.

```
public bool PayForDelivery(string userName, long billId)
{
    Bill bill = _billRepository.FindByIdAndIsDeliveryPaidFalse(billId);
    if (bill == null)
    {
        throw new DeliveryAlreadyPaidException();
    }
    User user = _userRepository.FindByIdAndUserMoneyInCentsGreaterThanEqual(userName, bill.CostInCents);
    if (user == null)
    {
        throw new NotEnoughMoneyException();
    }

    user.UserMoneyInCents = (user.UserMoneyInCents - bill.CostInCents);
    bill.IsDeliveryPaid = true;
    bill.DateOfPay = DateTime.Now;
    _userRepository.Save();
    _billRepository.Save();
    return true;
}
```

IUserRepository BLL.impl.BillService._userRepository

Рисунок 1.2 – Реалізація оплати доставки.

```
public Bill InitializeBill(DeliveryOrderCreateModel deliveryOrderCreateDto, string initiatorName)
{
    User addressee = _userRepository.FindByEmail(deliveryOrderCreateDto.AddresseeEmail);
    if (addressee == null){...}
    Way way = _wayRepository.FindByLocalitySand_IdAndLocalityGet_Id(deliveryOrderCreateDto.LocalitySandId
        , deliveryOrderCreateDto.LocalityGetId);
    if (way == null)
    {
        throw new NoSuchWayException();
    }
    Delivery newDelivery = getBuildDelivery(deliveryOrderCreateDto, addressee, way);
    _deliveryRepository.Create(newDelivery);
    User user = _userRepository.FindByName(initiatorName);
    if (user == null)
    {
        throw new NoSuchUserException();
    }
    Bill buildBill = getBuildBill(newDelivery
        , calculateDeliveryCost(deliveryOrderCreateDto.DeliveryWeight, way)
        , user);
    _billRepository.Create(
        buildBill);
    _billRepository.Save();
    return buildBill;
}
```

Рисунок 1.3 – Реалізація створення рахунку.

```
public List<BillModel> GetBillHistoryByUserName(string userName)
{
    return BillToBillDtoMapper.mapToList(
        entities: _billRepository.FindAllByUserNameAndIsDeliveryPaidTrue(userName));
}
```

Рисунок 1.4 – Реалізація отримання історії рахунків.

1.1.2 Реалізація інтерфейсу «IDeliverySevice»

```
public class DeliveryService : IDeliveryService
{
    private IWayRepository _wayRepository;
    private IDeliveryRepository _deliveryRepository;

    1 usage
    public DeliveryService(IWayRepository wayRepository, IDeliveryRepository deliveryRepository)
    {
        _wayRepository = wayRepository;
        _deliveryRepository = deliveryRepository;
    }
```

Рисунок 1.1 – Залежності реалізації інтерфейсу.

```
public List<DeliveryInfoToGetDto> GetDeliveryInfoToGet(string userName)
{
    return DeliveryToDeliveryInfoToGetDtoMapper.mapToList( entities: _deliveryRepository
        .FindAllByAddressee_IdAndIsPackageReceivedFalseAndBill_IsDeliveryPaidTrue(userName));
}
```

Рисунок 1.2 – Реалізація отримання списку доставок що очікують отримання

```
public bool ConfirmGettingDelivery(string userName, long deliveryId)
{
    Delivery delivery =
        _deliveryRepository.FindByIdAndAddressee_IdAndIsPackageReceivedFalse(userName, deliveryId);
    if (delivery == null)
    {
        throw new AskedDataIsNotExist();
    }

    delivery.IsPackageReceived = true;
    _deliveryRepository.Save();
    return true;
}
```

Рисунок 1.3 – Реалізація підтвердження отримання доставки.

```
public PriceAndTimeOnDeliveryModel GetDeliveryCostAndTimeDto(DeliveryInfoRequestModel deliveryInfoRequestDto)
{
    Way way = getWay(deliveryInfoRequestDto.LocalitySandId, deliveryInfoRequestDto.LocalityGetId);
    return new PriceAndTimeOnDeliveryModel(calculateDeliveryCost(deliveryInfoRequestDto.DeliveryWeight, way),
        way.TimeOnWayInDays);
}
```

Рисунок 1.4 – Реалізація отримання часту та вартості для конкретної доставки.

```
public interface IBillService
{
    List<BillInfoToPayModel> GetBillsToPayByUserName(string userName);

    bool PayForDelivery(string userName, long billId);

    Bill InitializeBill(DeliveryOrderCreateModel deliveryOrderCreateDto, string initiatorName);

    List<BillModel> GetBillHistoryByUserName(string userId);

}
```

Рисунок 1.5 – Інтерфейс сервісу рахунків.

```
public interface IDeliveryService
{
    List<DeliveryInfoToGetDto> GetDeliveryInfoToGet(string userName);

    bool ConfirmGettingDelivery(string userName, long deliveryId);

    PriceAndTimeOnDeliveryModel GetDeliveryCostAndTimeDto(DeliveryInfoRequestModel deliveryInfoRequestDto);
}
```

Рисунок 1.6 – Інтерфейс сервісу доставок.

```
public interface ILocalityService
{
    List<LocalityModel> GetLocalities();

    List<LocalityModel> FindGetLocalitiesByLocalitySetId(long id);
}
```

Рисунок 1.7 – Інтерфейс сервісу локацій.

```
public interface IUserService
{
    User FindByName(String email);

    User ReplenishAccountBalance(string userName, long amountMoney);
}
```

Рисунок 1.8 – Інтерфейс сервісу користувачів.

1.1.3 Реалізація інтерфейсу «ILocalitySevice»

```
public class LocalityService : ILocalityService
{
    private ILocalityRepository _localityRepository;
```

Рисунок 1.9 – Залежності реалізації інтерфейсу.

```
public List<LocalityModel> GetLocalities()
{
    return LocalityToLocalityModelMapper.mapToList( entities: _localityRepository.Get());
}
```

Рисунок 1.10 – Реалізація отримання списку локацій що є в системі.

1.1.4 Реалізація інтерфейсу «IUserService»

```
2 usages
public class UserService:IUserService
{
    private readonly IUserRepository _userRepository;

    1 usage  ··· More
    public UserService(IUserRepository userRepository)
    {
        _userRepository = userRepository;
    }
```

Рисунок 1.11 – Залежності реалізації інтерфейсу.

```
public User FindByName(string email)
{
    User byEmail = _userRepository.FindByName(email);
    if (byEmail==null)
    {
        throw new UsernameNotFoundException();
    }

    return byEmail;
}
```

Рисунок 1.12 – Реалізація отримання користувача за його ім’ям.

```
public User ReplenishAccountBalance(string userName, long amountMoney)
{
    User user = _userRepository.FindByName(userName);
    if (user == null)
    {
        throw new NoSuchUserException();
    }
    if (user.UserMoneyInCents + amountMoney <= 0) {
        throw new ToMuchMoneyException();
    }
    user.UserMoneyInCents= (user.UserMoneyInCents + amountMoney);
    _userRepository.Save();
    return user;
}
```

Рисунок 1.13 – Реалізація поповнення рахунку користувача.

1.2 Реалізація інтерфейсів шару доступу до даних

1.2.1 Реалізація інтерфейсу «IBillRepository»

```
public IEnumerable<Bill> FindAllByUserIdAndIsDeliveryPaidFalse(string userId)
{
    return Context.Bills.Include( navigationPropertyPath: u => u.User)
        .Include( navigationPropertyPath: d => d.Delivery).ThenInclude(w => w.Way)
        .ThenInclude(l => l.LocalitySand).Include( navigationPropertyPath: d => d.Delivery)
        .ThenInclude(w => w.Way)
        .ThenInclude(l => l.LocalityGet)
        .Where(bill => bill.User.UserName.Equals(userId) && !bill.IsDeliveryPaid);
}
```

Рисунок 1.2 – Реалізація пошуку неоплачених рахунків користувача.

```
public IEnumerable<Bill> FindAllByUserNameAndIsDeliveryPaidTrue(string userName)
{
    return base.Get( filter: bill => bill.User.UserName.Equals(userName) && bill.IsDeliveryPaid);
}
```

Рисунок 1.3 – Реалізація пошуку оплачених рахунків користувача.

```
public Bill FindByIdAndIsDeliveryPaidFalse(long billId)
{
    return base.Get( filter: bill => bill.BillId.Equals(billId) && !bill.IsDeliveryPaid).FirstOrDefault();
}
```

Рисунок 1.4 – Реалізація пошуку рахунку неоплаченої доставки.

1.2.2 Реалізація інтерфейсу «IDeliveryRepository»

```
public IEnumerable<Delivery> FindAllByAddressee_IdAndIsPackageReceivedFalseAndBill_IsDeliveryPaidTrue(
    string billUserId)
{
    return Context.Deliveries.Include( navigationPropertyPath: u => u.Addressee)
        .Include( navigationPropertyPath: b => b.Bill).ThenInclude(u => u.User)
        .Include( navigationPropertyPath: l => l.Way).ThenInclude(g => g.LocalityGet)
        .Include( navigationPropertyPath: l => l.Way)
        .ThenInclude(g => g.LocalitySand).Where(delivery =>
            delivery.Addressee.UserName.Equals(billUserId) && !delivery.IsPackageReceived &&
            delivery.Bill.IsDeliveryPaid);
}
```

Рисунок 1.2 – Реалізація пошуку доставок користувача що не були отримані та не були оплачені.

```
public Delivery FindByIdAndAddressee_IdAndIsPackageReceivedFalse(string userName, long deliveryId)
{
    return base.Get( filter: delivery =>
        delivery.DeliveryId.Equals(deliveryId) && delivery.Addressee.UserName.Equals(userName) &&
        !delivery.IsPackageReceived).FirstOrDefault();
}
```

Рисунок 1.3 – Реалізація пошуку неоплачених доставок користувача.

1.2.3 Реалізація інтерфейсу «ILocalityRepository»

```
public IEnumerable<Locality> FindGetLocalitiesByLocalitySandId(long idToSearch)
{
    return base.Get( filter: locality => locality.WaysWhereThisLocalityIsGet
        .Exists( match: way => way.LocalitySandLocalityId.Equals(idToSearch)));
}
```

Рисунок 1.2 – Реалізація пошуку варіантів кінцевих локацій за початковою.

1.2.4 Реалізація інтерфейсу «IUserRepository»

```
public User FindByEmail(string email)
{
    return base.Get( filter: user => user.Email.Equals(email)).FirstOrDefault();
}
```

Рисунок 1.2 – Реалізація пошуку користувача за електронною адресою.

```
public User FindByName(string userName)
{
    return base.Get( filter: user => user.UserName.Equals(userName)).FirstOrDefault();
}
```

Рисунок 1.3 – Реалізація пошуку користувача за іменем.

```
public User FindByIdAndUserMoneyInCentsGreaterThanEqual(string userName, long userMoneyInCents)
{
    return base.Get( filter: user => user.UserName.Equals(userName) && user.UserMoneyInCents>=userMoneyInCents)
        .FirstOrDefault();
}
public Bill FindByIdAndIsDeliveryPaidFalse(long billId)
{
    return base.Get( filter: bill => bill.BillId.Equals(billId) && !bill.IsDeliveryPaid).FirstOrDefault();
}
```

Рисунок 1.4 – Реалізація пошуку користувача за іменем та сумою коштів на рахунку.

1.2.1 Реалізація інтерфейсу «IWayRepository»

```
public Way FindByLocalitySand_IdAndLocalityGet_Id(long localitySandID, long localityGetID)
{
    return Context.Ways.Include( navigationPropertyPath: p => p.WayToTariffWeightFactors)
        .ThenInclude(factor =>factor.TariffWeightFactor).
        FirstOrDefault(way => way.LocalitySandLocalityId.Equals(localitySandID) &&
                              way.LocalityGetLocalityId.Equals(localityGetID));
}
```

Рисунок 1.2 – Пошук шляху за початковим і кінцевим пунктами.

2 РЕЗУЛЬТАТ РОЗРОБКИ ДОДАТКУ

В даному розділі запропоновано скріншоти роботи додатку. На рисунку 2.1 запропоновано сторінку калькулятора вартості доставки. На рисунках 2.2 та 2.3 запропоновано сторінки регістрації та в ходу в додаток відповідно.

На рисунках 2.4, 2.5, 2.6 та 2.7 зображено сторінки доступні ідентифікованому користувачу.

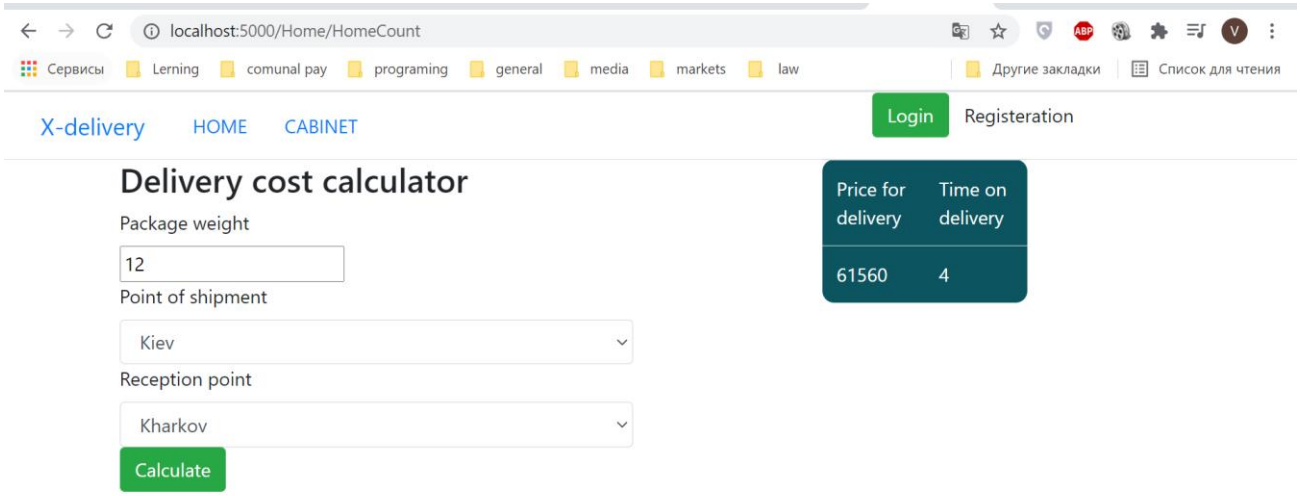

Рисунок 2.1 – Головна сторінка, калькулятор вартості та часу доставок.

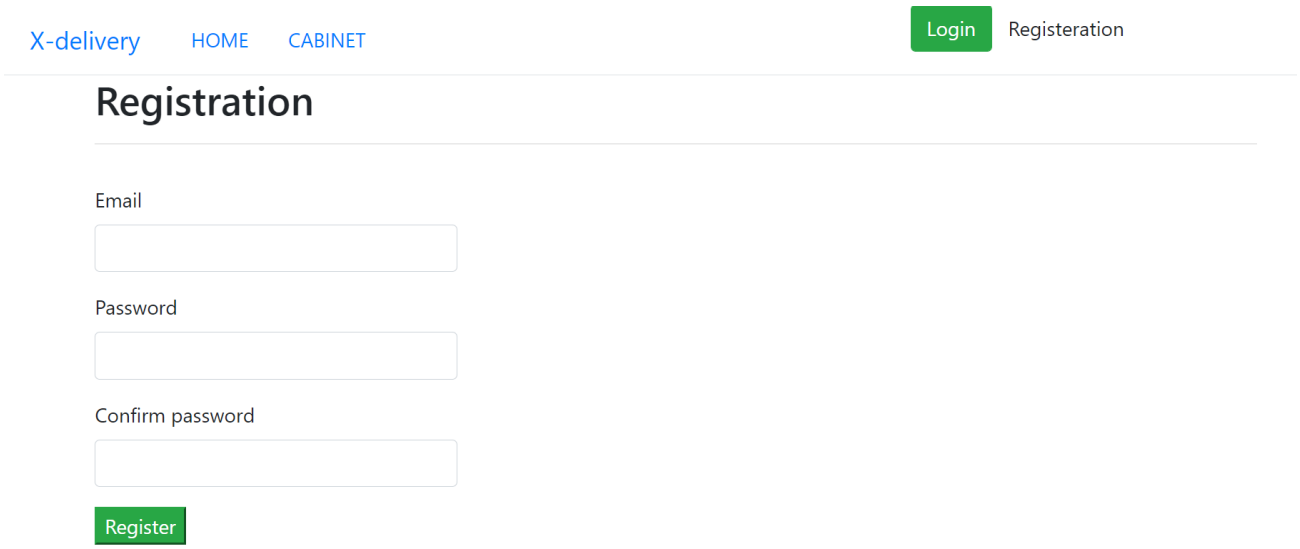

Рисунок 2.2 – Сторінка регістрації.

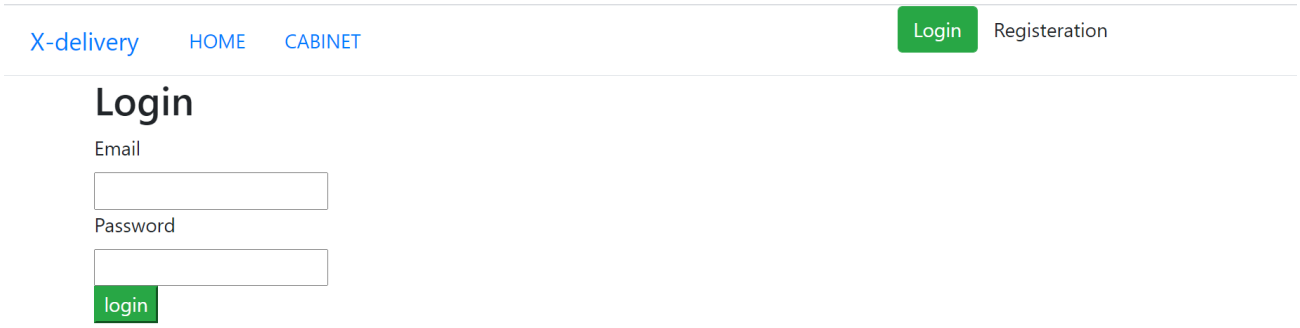

Рисунок 2.3 – Сторінка входу в додаток.

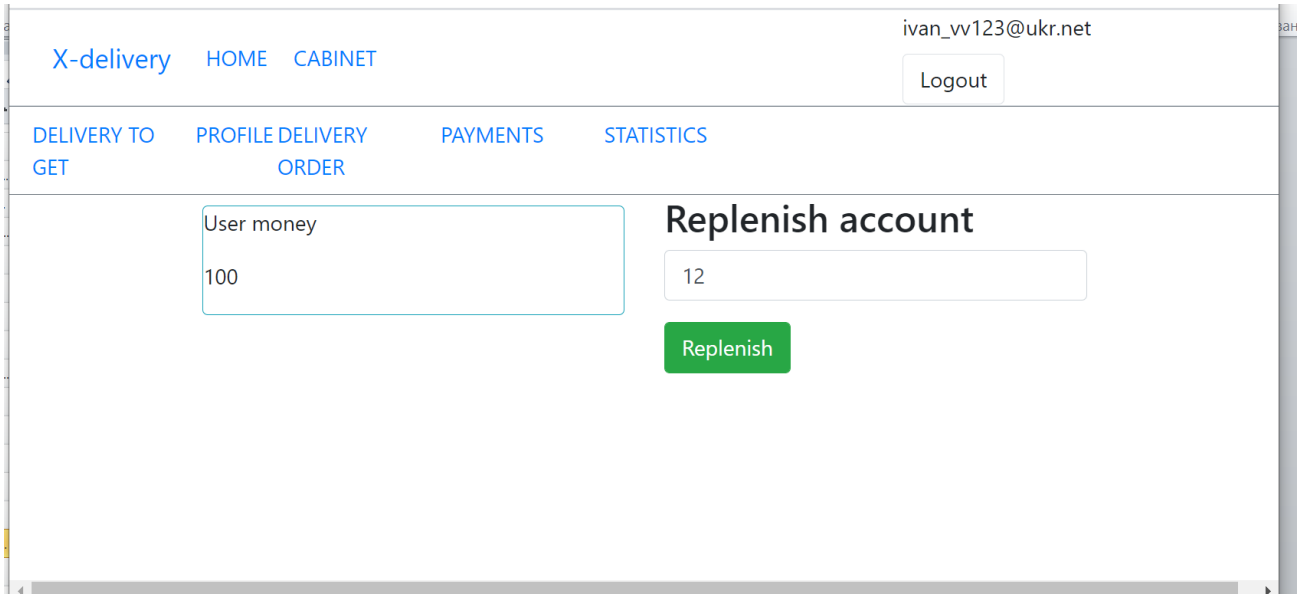

Рисунок 2.4 – Сторінка балансу користувача.

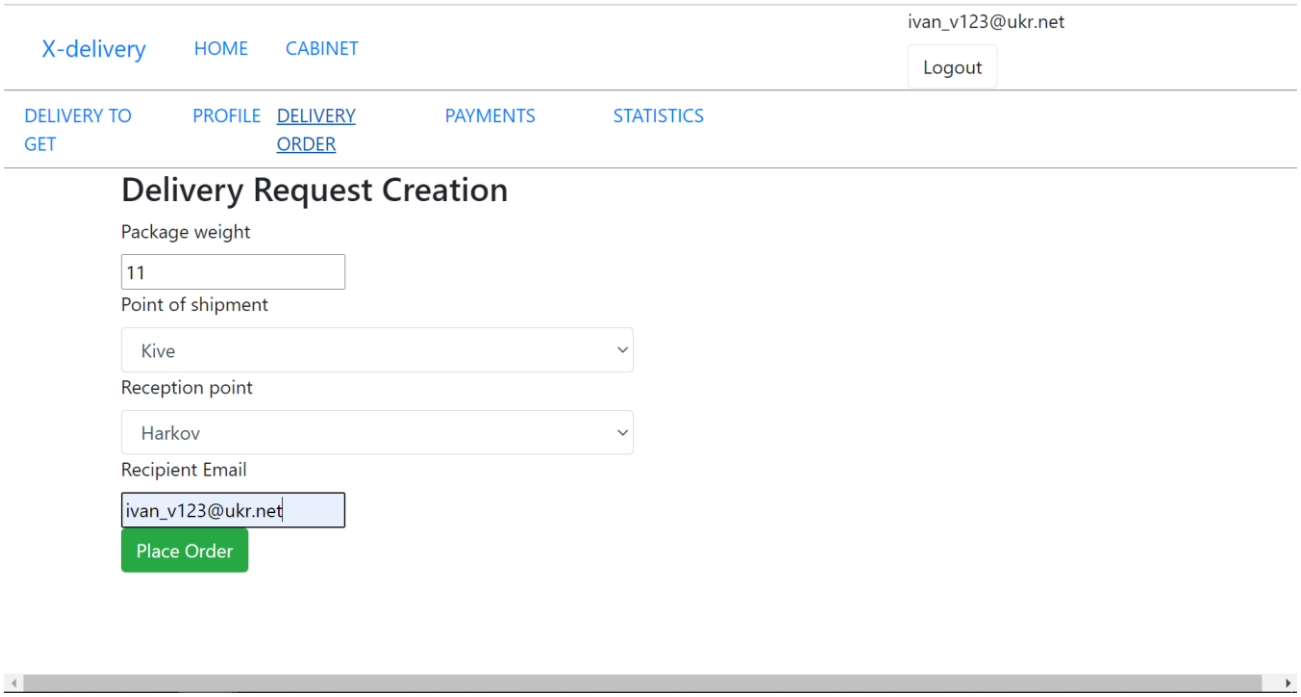

Рисунок 2.5 – Сторінка створення замовлення.

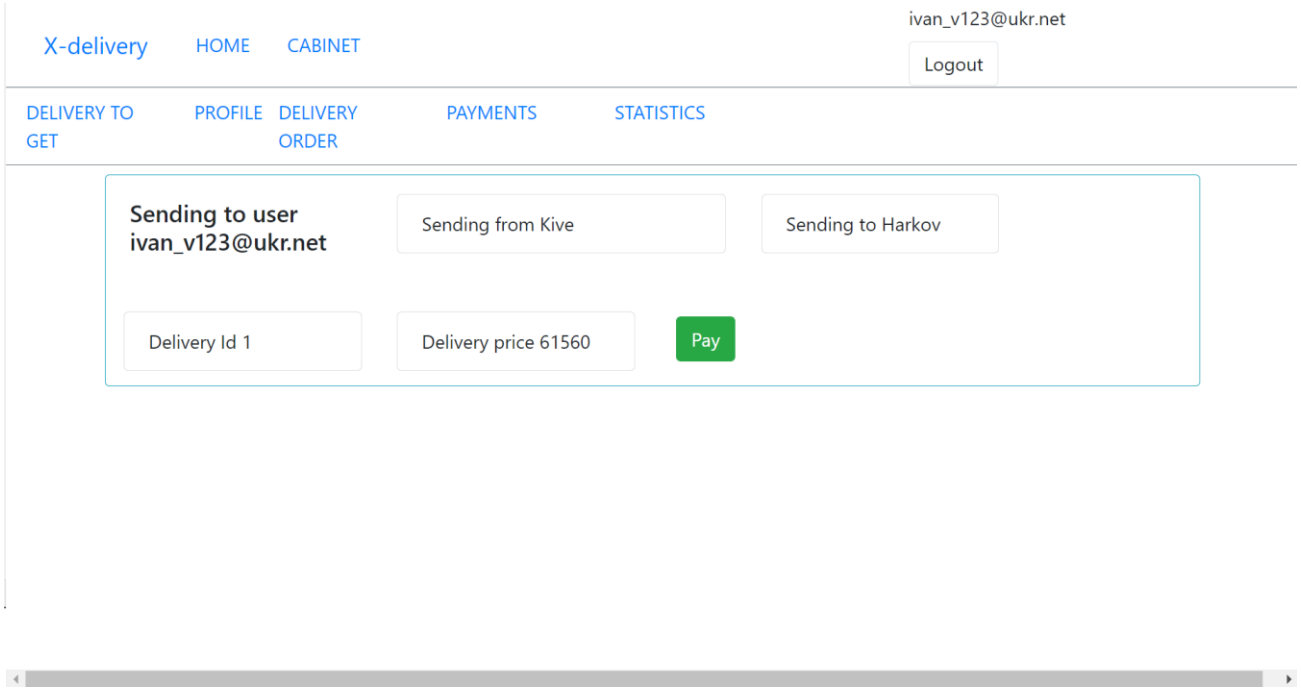

Рисунок 2.6 – Сторінка оплати замовлення.

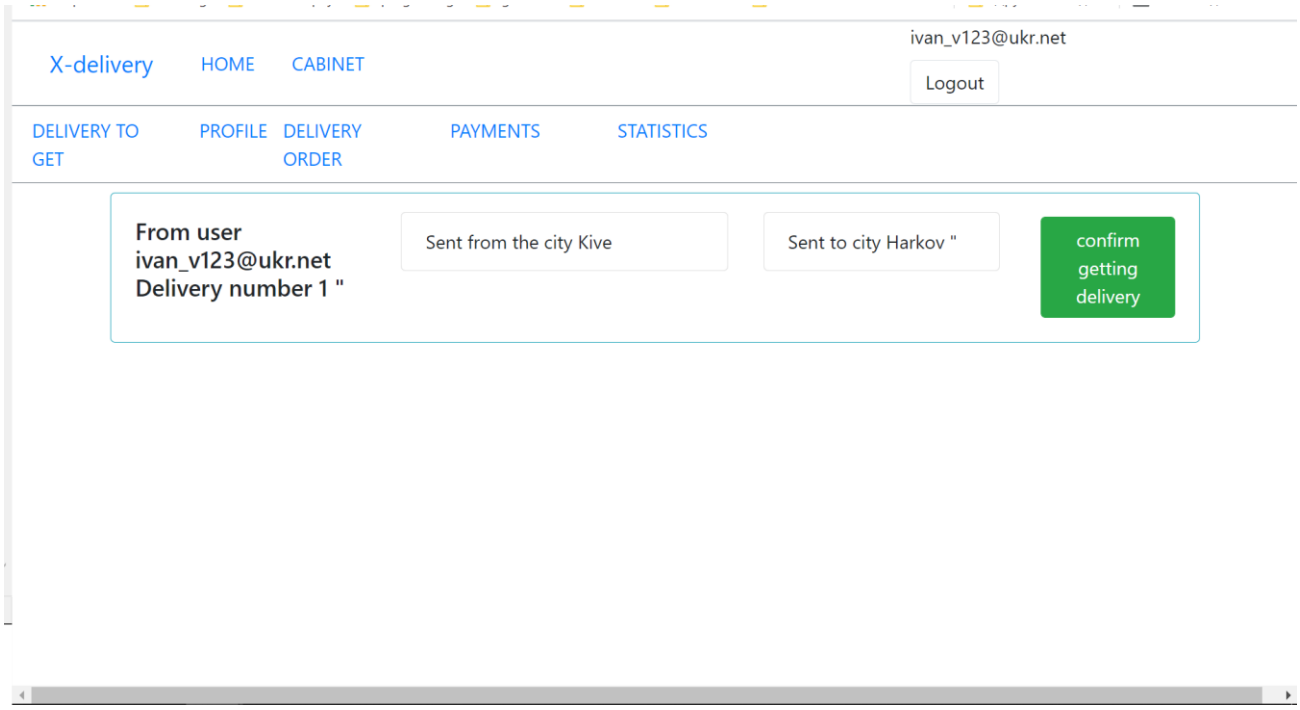

Рисунок 2.7 – Сторінка підтвердження отримання замовлення.

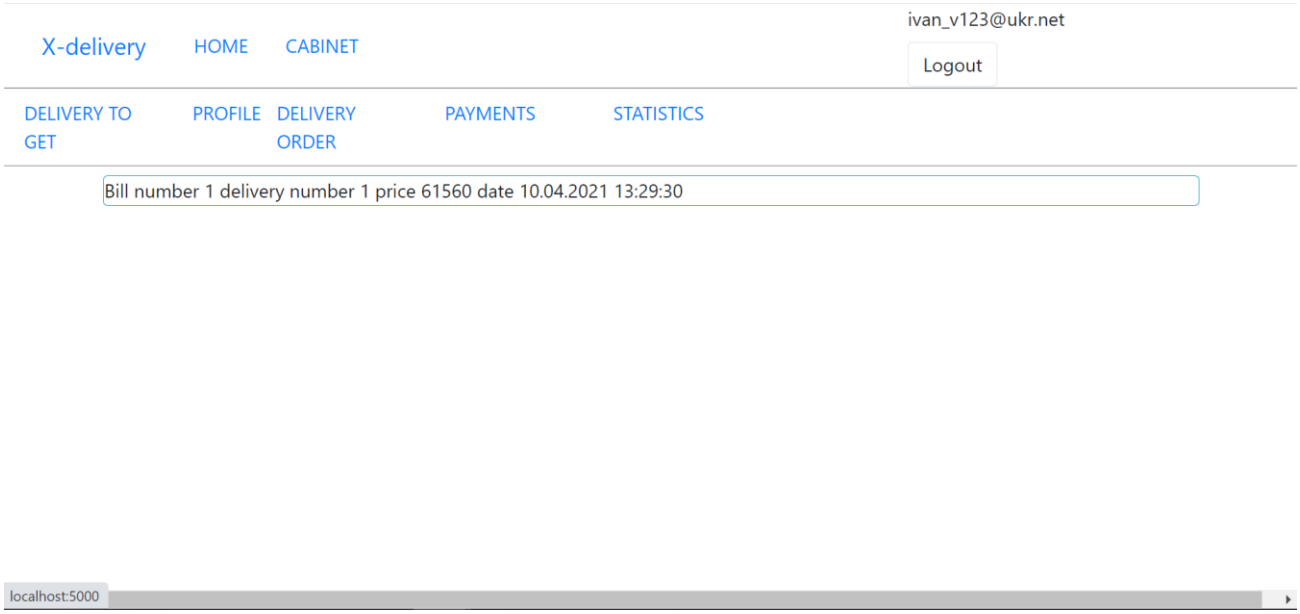

Рисунок 2.7 – Сторінка історії платежів.

ВИСНОВКИ

В дані роботі було повністю створено спроектований у попередніх лабораторних роботах додаток. В звіт було представлено реалзіації ключових інтерфейсів. Окрім цього було створено рівень контролерів, що забезпечує реалізацію API за HTTP протоколом. А також займається перевіркою на коректність даних що отримує сервер зі сторони клієнта.

Для забезпечення ізоляції шарів та забезпечення модульності гнучкості та тестованості коду, всі класи було реалізовано з дотриманням принципів СОЛІД. Створення екземплярів об’єктів було делеговано інфраструктурі. Передача повідомлень між шарами здійснюється через інтерфейси та з допомогою транспортних класів.

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.