

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

ЛАБОРАТОРНА РОБОТА 2

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 Постановка задачі .....	3
1 Реалізація основних інтерфейсів .....	4
1.1 Інтерфейси рівня бізнес логіки .....	4
1.2 Інтерфейси рівня доступу до даних .....	5
2 реалізація інверсії залежностей .....	7
2.1 Пояснення прийнятого рішення .....	7
2.2 Конфігурація та налаштування інверсії залежностей .....	7
Висновки .....	8
Список використаних джерел .....	9

## 1 ПОСТАНОВА ЗАДАЧІ

1) Реалізувати основні інтерфейси відповідно до розробленої у попередній лабораторній роботі UML моделі проекту.

2) Реалізувати п'ятий принцип солід «інверсії залежностей», з допомогою ІоС контейнеру чи іншої сутності що відповідає за впровадження залежностей між модулями, шарами, сервісами та дозволяє розбирати тестопридатне програмне забезпечення.

## 1 РЕАЛІЗАЦІЯ ОСНОВНИХ ІНТЕРФЕЙСІВ

### 1.1 Інтерфейси рівня бізнес логіки

```
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 1.1 – Інтерфейс сервісу рахунків.

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

    bool ConfirmGettingDelivery(string userName, long deliveryId);

    PriceAndTimeOnDeliveryModel GetDeliveryCostAndTimeDto(DeliveryInfoRequestModel deliveryInfoRequestDto);
}
```

Рисунок 1.2 – Інтерфейс сервісу доставок.

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

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

Рисунок 1.3 – Інтерфейс сервісу локацій.


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

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

Рисунок 1.4 – Інтерфейс сервісу користувачів.

## 1.2 Інтерфейси рівня доступу до даних

```
public interface IBillRepository:IGenericRepository<Bill>
{
     5 usages  1 implementation
    IEnumerable<Bill> FindAllByUserIdAndIsDeliveryPaidFalse(string userId);

     3 usages  1 implementation
    IEnumerable<Bill> FindAllByUsernameAndIsDeliveryPaidTrue(string userId);




     5 usages  1 implementation
    Bill FindByIdAndIsDeliveryPaidFalse(long billId);
}
```

Рисунок 1.5 – Інтерфейс репозиторію рахунків.

```
public interface IDeliveryRepository:IGenericRepository<Delivery>
{
     3 usages  1 implementation
    IEnumerable<Delivery> FindAllByAddressee_IdAndIsPackageReceivedFalseAndBill_IsDeliveryPaidTrue
    (string billUserId);



     4 usages  1 implementation
    Delivery FindByIdAndAddressee_IdAndIsPackageReceivedFalse(string userName, long deliveryId);
}
```

Рисунок 1.6 – Інтерфейс репозиторію доставок.







```
public interface ILocalityRepository:IGenericRepository<Locality>
{
     1 usage  1 implementation
    IEnumerable<Locality> FindGetLocalitiesByLocalitySandId(long idToSearch);
}
```

Рисунок 1.7 – Інтерфейс репозиторію локацій.

```
public interface IUserRepository: IGenericRepository<User>
{
     4 usages  1 implementation
    User FindByEmail(String email);

     9 usages  1 implementation
    User FindByName(string userName);



     4 usages  1 implementation
    User FindByIdAndUserMoneyInCentsGreaterThanOrEqualTo(string userName, long userMoneyInCents);
}
```

Рисунок 1.8 – Інтерфейс репозиторію користувачів.



```
public interface IWayRepository: IGenericRepository<Way>
{
     9 usages  1 implementation
    Way FindByLocalitySand_IdAndLocalityGet_Id(long localitySandID, long localityGetID);
}
```

Рисунок 1.9 – Інтерфейс репозиторію маршрутів.

## 2 РЕАЛІЗАЦІЯ ІНВЕРСІЇ ЗАЛЕЖНОСТЕЙ

### 2.1 Пояснення прийнятого рішення

В якості реалізації контролю інверсії залежностей було вирішено застосувати інструменти що пропонує .NET core фреймворк. Таке рішення було прийнято оскільки, за загальною практикою краще використовувати перевірені бібліотеки для будь-яких шаблонних задач. В джава реалізації даного проекту запропоновано ручну реалізацію ІОС контейнеру. Її можна знайти за посиланням «<https://github.com/VINIPOOH/ServletFinalProject>»

### 2.2 Конфігурація та налаштування інверсії залежностей

На рисунку 2.1 запропоновано частину конфігурації із «Startup.cs» файлу.

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllersWithViews();
    services.AddDbContext<MyDbContext>();
    services.AddIdentity<User, IdentityRole>( setupAction: opt =>
    {
        opt.Password.RequireUppercase = false;
        opt.Password.RequireNonAlphanumeric = false;
        opt.Password.RequiredLength = 3;
        opt.Password.RequireDigit = false;
        opt.User.RequireUniqueEmail = true;
    }).AddEntityFrameworkStores<MyDbContext>().AddDefaultTokenProviders();

    services.AddScoped<IDeliveryService, DeliveryService>();
    services.AddScoped<ILocalityService, LocalityService>();
    services.AddScoped<IWayRepository, WayRepository>();
    services.AddScoped<IDeliveryRepository, DeliveryRepository>();
    services.AddScoped<ILocalityRepository, LocalityRepository>();
    services.AddScoped<IUserService, UserService>();
    services.AddScoped<IUserRepository, UserRepository>();
    services.AddScoped<IBillService, BillService>();
    services.AddScoped<IBillRepository, BillRepository>();
}
```

Рисунок 2.1 – Конфігурація інверсії залежностей в стартап файлі.

## ВИСНОВКИ

В даній роботі було створено основні інтерфейси розроблені в процесі попередньої лабораторної роботи, а також проаналізовано різні варіанти досягнення інверсії залежностей та обрано ASP net в якості бібліотеки для забезпечення інверсії залежностей.



## 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., Чистый код: создание, анализ и рефакторинг. — СПб.: Питер, 2019. — 464 с.: ил.