КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра інтелектуальних технологій

ЗВІТ

з проектно-технологічної практики

студента 3 курсу групи КН-32
спеціальності **122** «Комп'ютерні науки»
Кутрика Олега Ігоровича

База практики: ТОВ "АДВАНСЕД СОФТВЕР СОЛЮШНС"

Керівник практики: Іскендерова Людмила Григорівна
Оцінка керівника практики: _____
(Керівник від підприємства)

М.П. 

Керівник практики: _____
Оцінка керівника практики: _____
(Керівник від Університету)

Оцінка, отримана при захисті: _____
_____ (_____)

КИЇВ 2023

Київський національний університет імені Тараса Шевченка
Факультет інформаційних технологій
Кафедра інтелектуальних технологій

ЗАВДАННЯ

на індивідуальну роботу на проектно-технологічну практику
студенту Кутрику Олегу Ігоровичу
(прізвище, ім'я, по батькові)

1. Frontend
   1.1. Розробка дизайну сайту. Дизайн повинен відповідати усім сучасним вимогам та трендам, передбачити, що сайт буде адаптивним.
   1.2. Адаптивне верстання сайту.
2. Backend
   2.1. Розробка реляційної БД, а саме створення концептуальної, логічної та фізичної моделей БД. БД повинна включати не менше трьох таблиць.
   2.2. Створення таблиць БД у конкретній СУБД, наприклад MySQL.
   2.3. Розробка серверної частини роботи з реляційною БД.

Дата видачі завдання «23» січня 2023 р.

Вступ

Виробнича практика студентів є невід’ємною частиною процесу підготовки фахівців зі спеціальності 122 «Комп’ютерні науки». Практика студентів проводиться на базах навчальних закладів, підприємств і організацій різних галузей господарства, які займаються розробкою і впровадженням інформаційних технологій. Розроблена програма покликана забезпечити системність, єдиний комплексний підхід до організації практичної підготовки, неперервність та наступність навчання студентів.

Основним завданням виробничої практики є поглиблення та вдосконалення навичок використання прикладних систем обробки даних та інформаційних мереж для аналізу даних та розробки і оформлення документації; формування системного підходу до розв'язування фахових задач та прийняття рішень з використанням комп’ютерних технологій. Під час проходження виробничої практики студенти набувають базових навичок, необхідних для подальшого опанування сучасними технологіями програмування з метою. Виробнича практика покликана сформувати у студентів професійні вміння, навички прийняття самостійних рішень на конкретній ділянці роботи в реальних виробничих умовах шляхом виконання обов'язків, властивих їх майбутній професійній та організаційно-управлінській діяльності.

Виробнича практика – це вид роботи, спрямований на розширення та закріплення теоретичних і практичних знань, отриманих студентами у процесі навчання, удосконалення навичок проектної і експертної діяльності, навичок застосування інструментів аналізу даних.

Під час першого тижню практики студент знайомиться з колективом, із темою та завданнями практики, вивчає структуру підприємства, підписує необхідні для роботи документи, проходить інструктаж зі своїм куратором практики. Наступні два тижні студент виконує конкретні завдання та доручення.

Я проходив практику у ТОВ ”АДВАНСЕД СОФТВЕР СОЛЮШНС”. Підприємство займається розробкою веб-застосунків для маркетингу, логістики, розробки навчальних платформ, блокчейну, тощо.

У компанії застосовуються передові технології для створення застосунків, а саме хмарні ресурси, штучний інтелект та блокчейн технології

Зміст та результати виконання робіт

За видами задач, що вирішуються під час проходження проектно-технологічної практики, передбачається такий орієнтовний план-графік:

№	ЗАХІД	Індивідуальна	Самостійна
		робота студента (годин)	
1.	Проведення інструктажу з охорони праці, ознайомлення з підприємством чи організацією (базою практики).	10	2
2.	Ознайомлення із задачами управління технологічними процесами на підприємстві (організації).	15	8
3.	Аналіз структури підприємства (організації), функціонального призначення окремих підрозділів.	10	10
4.	Ознайомлення зі складом, функціями та характеристиками програмного забезпечення. що експлуатується на підприємстві (організації) чи окремому підрозділі.	15	15
5.	Аналіз основних напрямів подальшого удосконалення процесу експлуатації програмного забезпечення.	10	15
6.	Узагальнення матеріалів з практики, оформлення звіту складання заліку.	10	20
7.	**Всього**	**70**	**70**

Моя практична частина проходила у відділі комп’ютерного програмування, у команді веб-розробників. За період практики я ознайомився з декількома додатками для створення сервера для імплементації REST інтерфейсу : Django та Rest Framework

Django це потужний інструмент для розробки веб серверів, який використовує міграції для створення таблиць в базі даних. Також, у Django є своя технологія Django ORM , яка дозволяє швидше та ефективніше писати запити до бази даних.

Rest Framework - це фреймворк для створення REST API (Representational State Transfer, підхід в якому дані передаються у таких стандартних

форматах як XML та JSON). Він надає можливість створити класи-серіалізатори, завдяки яким спрощується реалізація логіки та маршрутів

Перший тиждень практики зайняв вивчення цих технологій, інші два – створення сайту по завданню. База практики погодилась з завданням від факультету.

Необхідно було розподілитися по групах між студентами, у зв'язку з чим я виконував завдання з Шилом Павлом, та розробити сайт за завданням :

Розробити сайт - каталог компаній, що займаються інтернет доставкою. Компанії можуть працювати як в одному місті, так і в декількох містах.

Користувач може відфільтрувати компанії за певними параметрами. В адмін-панелі надати можливість ведення нових компаній з описом їх послуг та їх тарифами

За відповідні елементи відповідали:

Frontend

1. Розробка дизайну сайту. Дизайн повинен відповідати усім сучасним вимогам та трендам передбачати, що сайт буде адаптивним. ()
2. Адаптивне верстання сайту ()

Backend

1. Розробка реляційної БД, а саме створення концептуальної, логічної та фізичної моделей БД (). БД повинна включати не менше трьох таблиць.
2. 2. Створення таблиць БД у конкретній СУБД, наприклад MySQL().
3. 3. Розробка серверної частини для роботи з реляційною БД ()

Для розробки були використані наступні технології

- React
- MySQL
- Python

Також використані наступні інструменти

- PyCharm
- Rest Framework
- Figma

Pycharm – це один з найбільш популярних редакторів коду, розроблений корпорацією Jetbrains.

Figma – це графічний онлайн-редактор для спільної роботи. У ньому можна створити прототип сайту, інтерфейс програми та обговорити правки з колегами в реальному часі. У Figma можна відмалювати елементи інтерфейсу, створити інтерактивний прототип сайту і додатку, ілюстрації, векторну графіку.

Основними перевагами для вибору цього інструменту для створення дизайну сайту були: Figma вирішує проблему зберігання робочих файлів, оскільки вони знаходяться на власній хмарі; можливість спільного редагування; зручність та зрозумілість інтерфейсу.

Rest Framework- це інструмент для створення API, який також дозволяє надсилати запити до свого серверу та працювати з ним через вбудований графічний інтерфейс. З його допомогою можна протестувати бекенд та переконатися, що він коректно працює.

Інформаційна архітектура веб-додатку

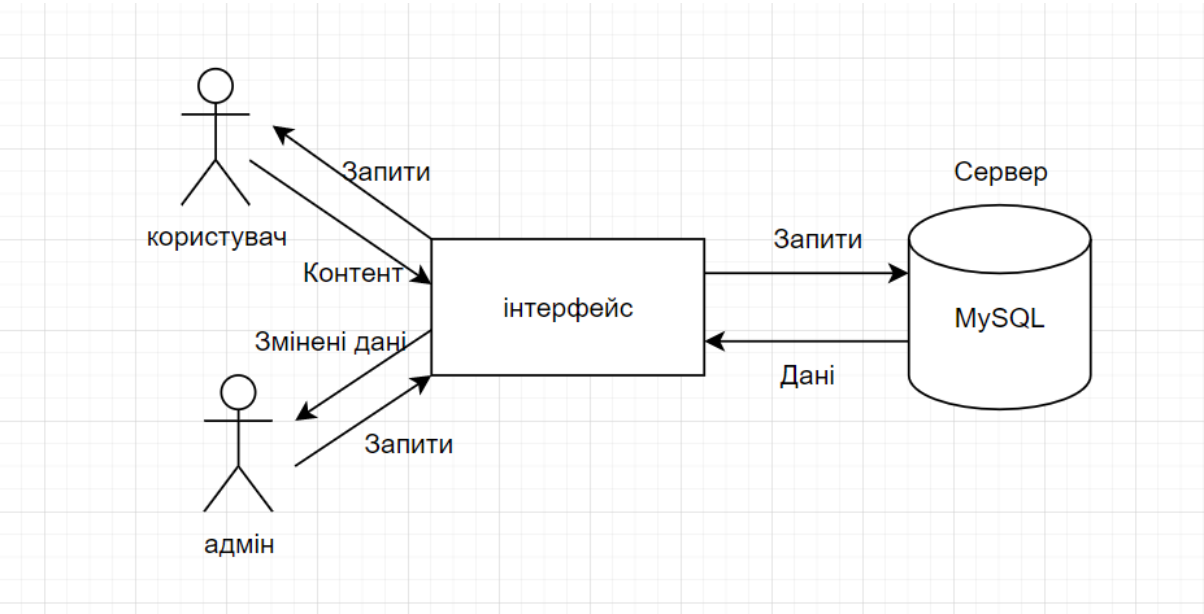

Рис. 1 - Архітектура розробленого веб-додатку

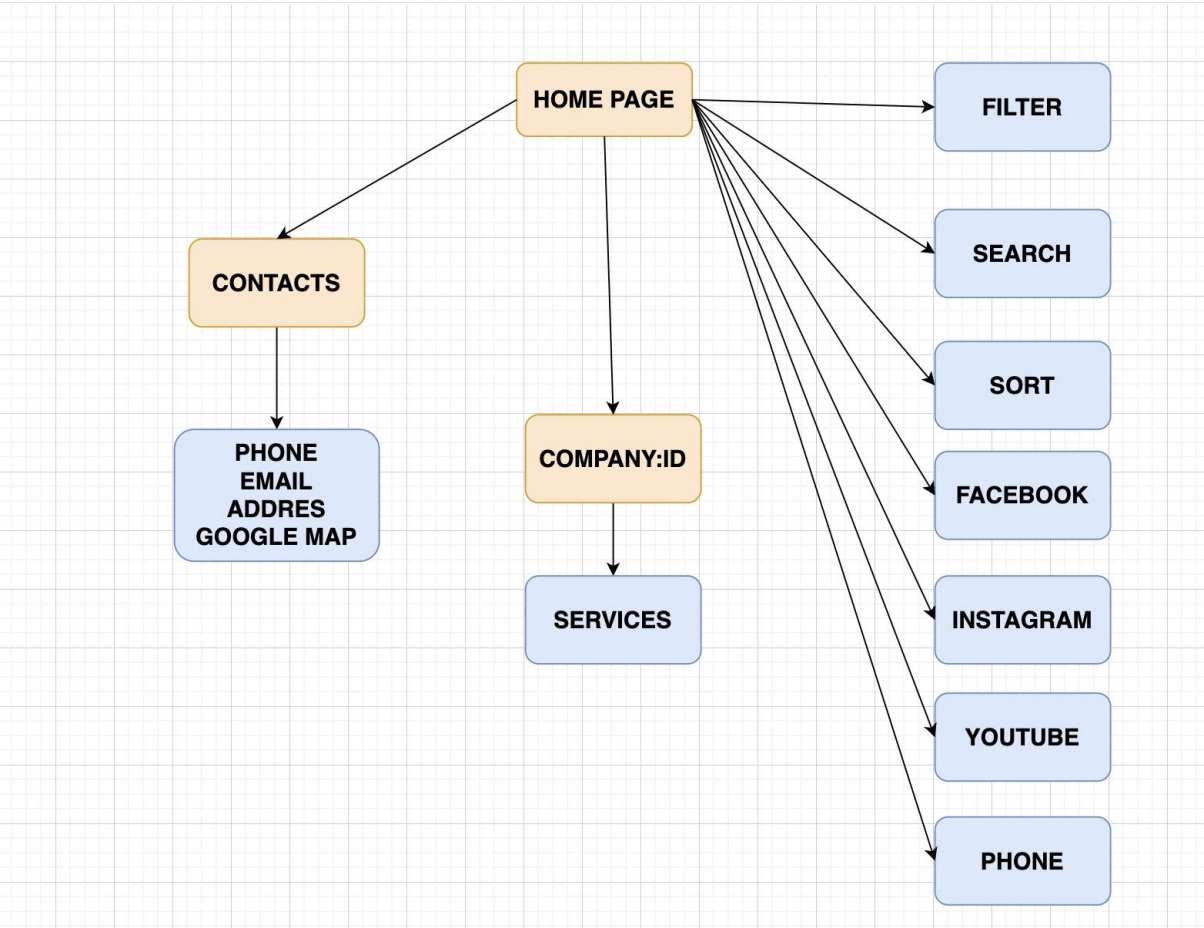

Рис. 2 - Карта сайту

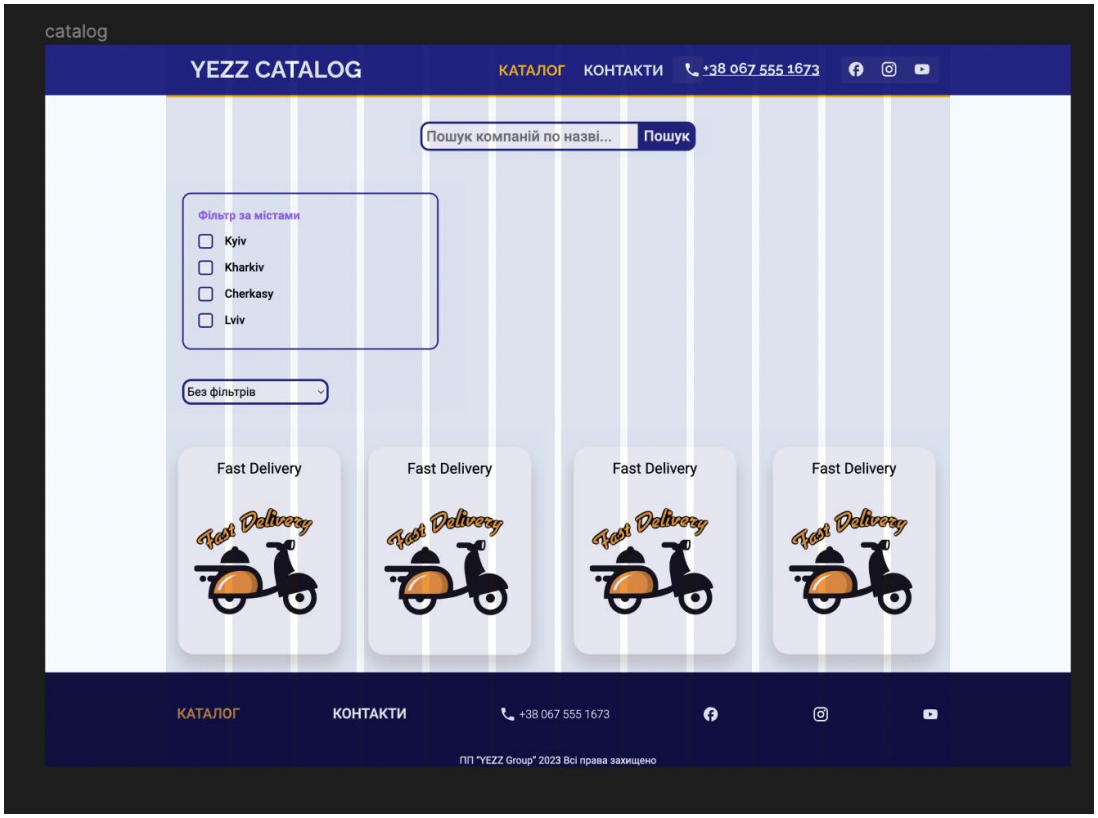

Рис.3 - Дизайн головної сторінки в Figma

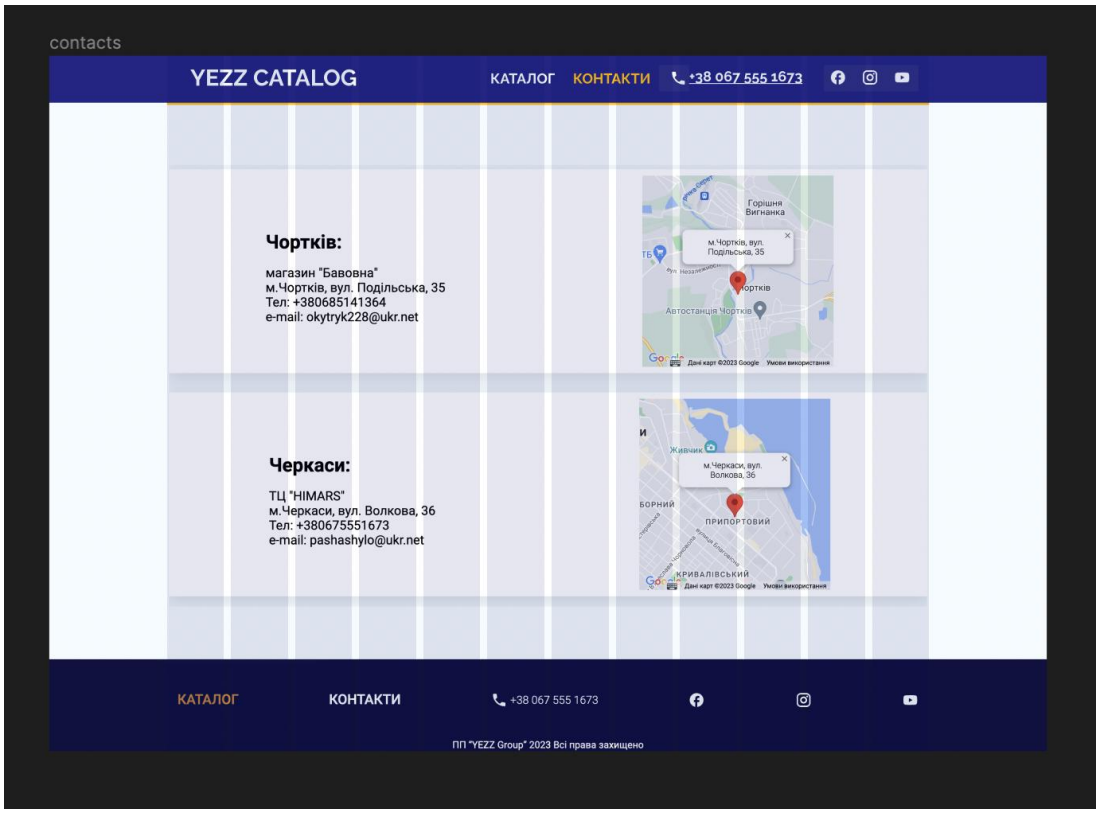

Рис.4 - Дизайн сторінки контактів в Figma

Рис.5 - Дизайн сторінки окремої компанії в Figma

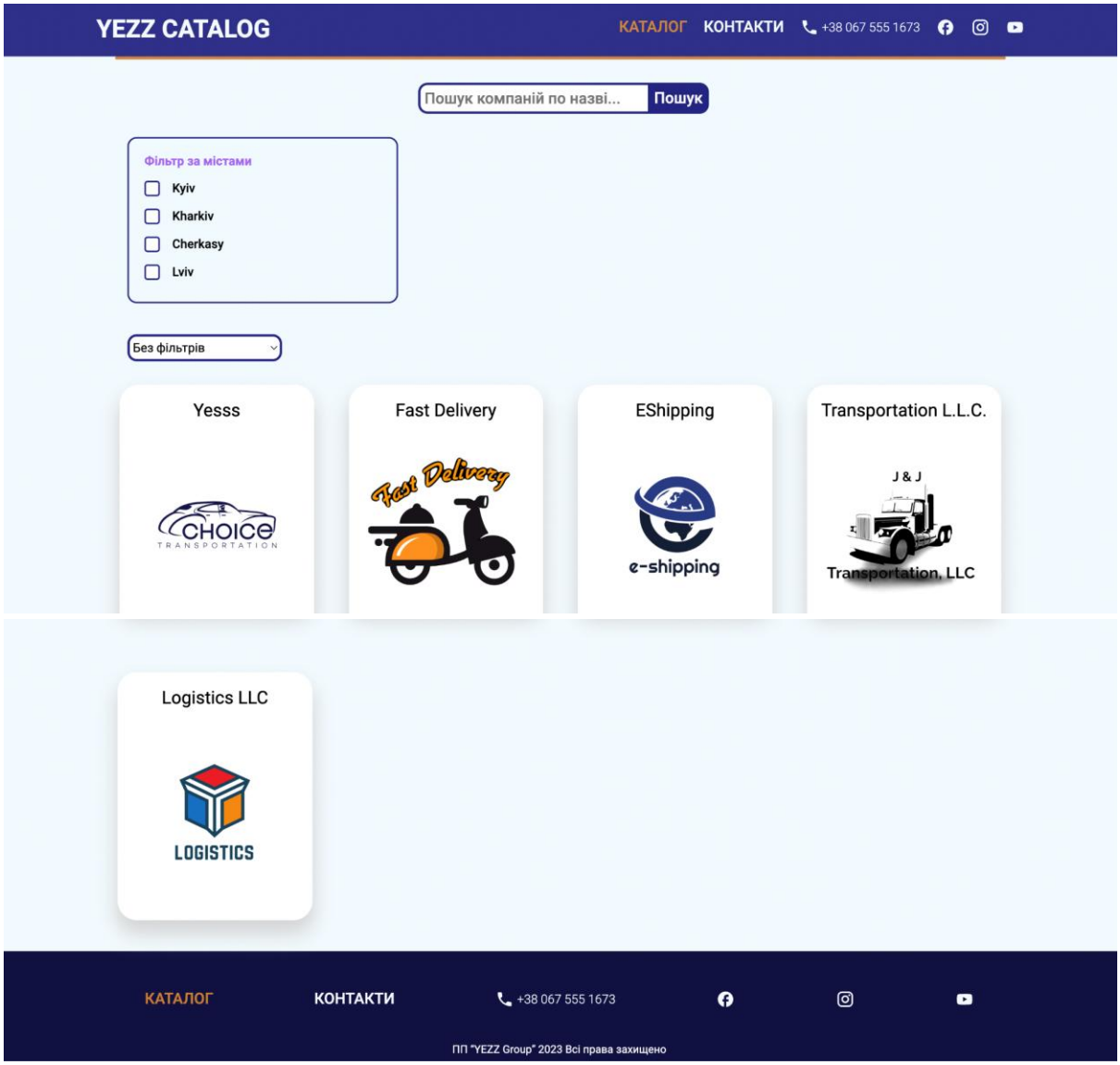

Рис.6, 7 - головна сторінка

На фотографіях зображені зверстана головна сторінка. Вона має header, footer та main блок, у якому знаходиться контент. В блоці main зверстане пошукове поле і кнопка для нього, зверстаний фільтр за містами, кнопки типу checkbox, зверстане dropdown сортування. За допомогою grid зверстав адаптивний вивід компаній на екран.

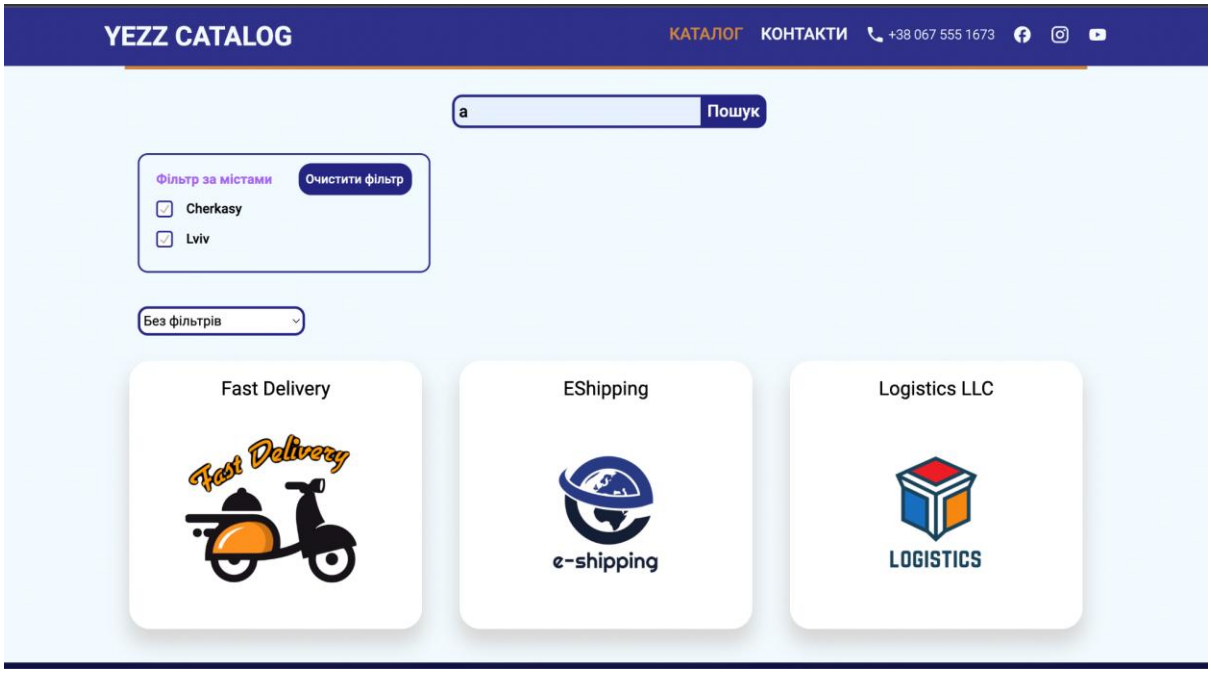

Рис.8 - фільтри

При обиранні фільтрів, головна сторінка змінює свій вигляд і виводить товари за фільтром.

Рис.9 - приклад фільтру

Також змінюється посилання сторінки після обрання фільтрів, це дозволяє ділитися сторінкою з вибраними фільтрами з іншими користувачами.

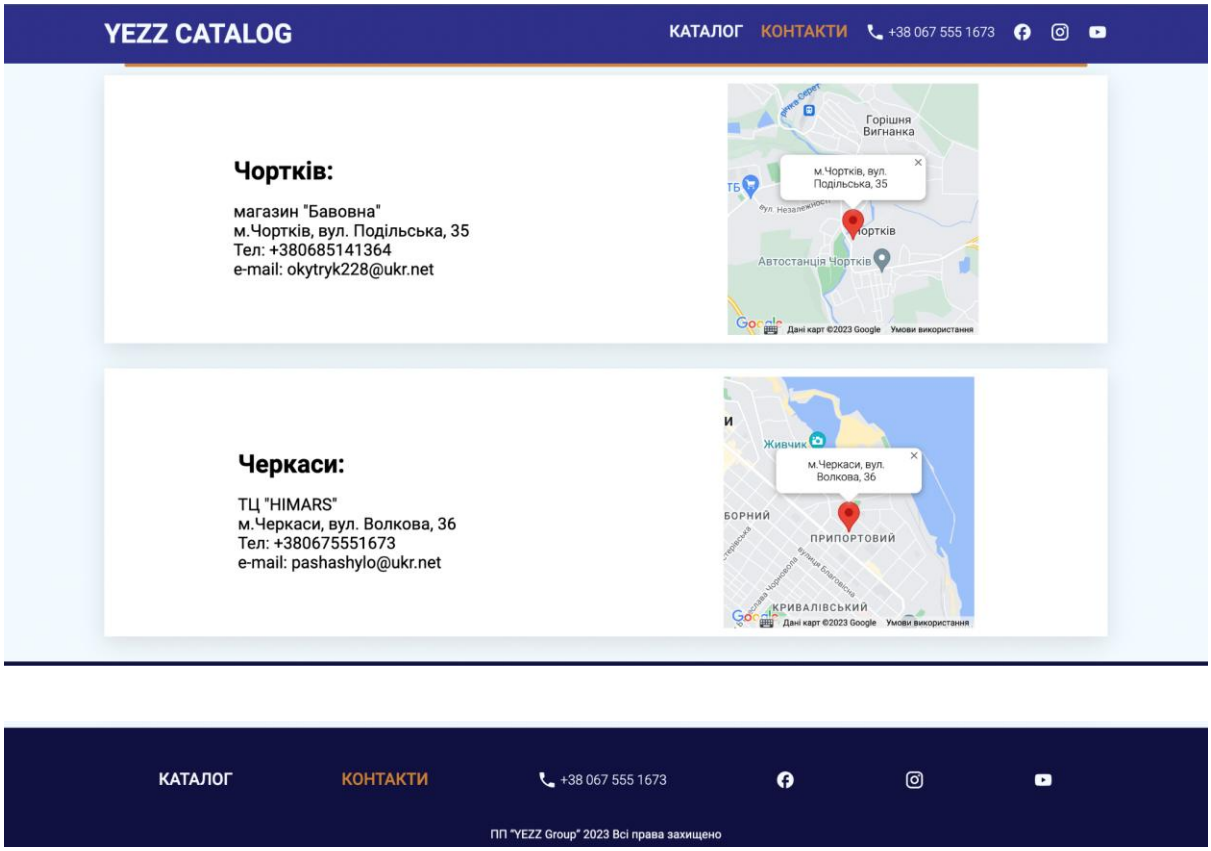

Рис.10, 11 - Контакти сайту

Сторінка “контакти” має такі ж header і footer. В блоці main імплементована кастомна гугл карта з нашими адресами.

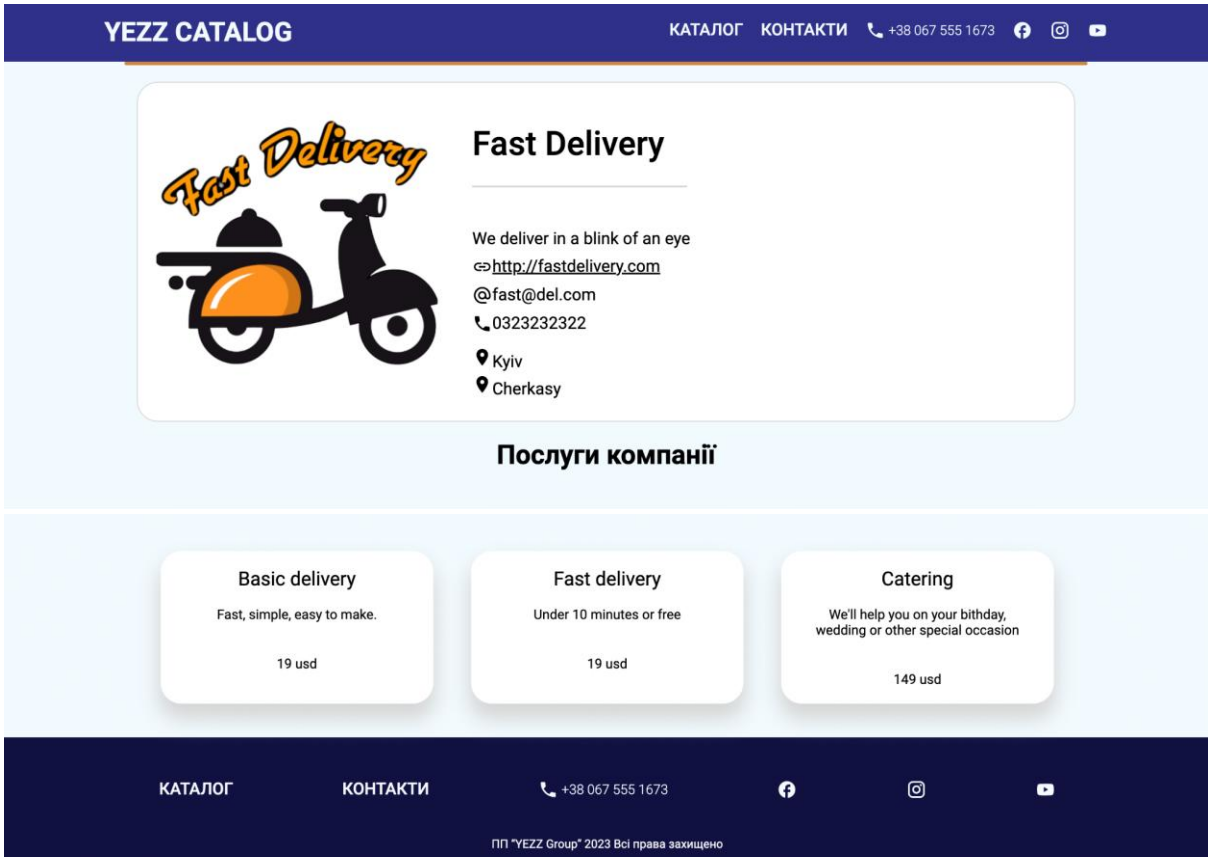

Рис.12, 13 - сторінка компанії

Окрема сторінка компанії має такі ж header і footer. В блоці main детальна інформація про компанію, її фото та контакти і послуги.

Company Api

```
GET /company/
```

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
```


```
[
    {
        "id": 2,
        "name": "Yesss",
        "description": "we make yess",
        "website": "https://yess.com",
        "email": "ye@ss.com",
        "phonenum": "0632632632",
        "photo": "https://fit-practice-backend.herokuapp.com/media/photos/2023/02/10/yess.png",
        "city": [
            {
                "name": "Kyiv"
            }
        ],
        "service": [
            {
                "name": "Yess",
                "description": "We provide the finest yess on the market",
                "price": 322
            },
            {
                "name": "YEESSSS",
                "description": "LET THE YESS BEGIN",
                "price": 777
            },
            {
                "name": "World of Tanks",
                "description": "when yes got you bored",
                "price": 99999999
            }
        ]
    },
```


Рис. 14 - GET запит на отримання списку всіх компаній

```python
class CompanyAPIView(generics.ListAPIView):
    serializer_class = CompanySerializer
    filter_backends = (DjangoFilterBackend, SearchFilter, OrderingFilter)
    filter_fields = ('id',)
    search_fields = ('name', 'description', "service__name")
    filterset_fields = ('city', )
    ordering_fields = ('name',)

    alezhuq
    def get_queryset(self):
        queryset = Company.objects.prefetch_related("city").prefetch_related("service").all()
        return queryset


alezhuq
class CompanyRetrieveUpdateDestroyAPIView(generics.RetrieveUpdateDestroyAPIView):
    serializer_class = CompanySerializer
    permission_classes = (IsStaffOrReadOnly,)
    queryset = Company.objects.prefetch_related("city").all()


alezhuq
class CityAPIView(generics.ListAPIView):
    serializer_class = CitySerializer
    queryset = City.objects.all()


alezhuq
class ServiceAPIView(generics.ListAPIView):
    filter_backends = (SearchFilter, OrderingFilter)
    serializer_class = ServiceSerializer
    queryset = Service.objects.select_related("company").all()
```

```python
urlpatterns = [
    path('city/', CityAPIView.as_view()),

    path('service/', ServiceAPIView.as_view()),

    path('company/', CompanyAPIView.as_view()),
    path('company/<int:pk>/', CompanyRetrieveUpdateDestroyAPIView.as_view()),
]
```

Рис. 15, 16 - програмний код реалізації логіки API застосунку та маршрутів

```python
class CitySerializer(serializers.ModelSerializer):
    alezhuq
    class Meta:
        model = City
        fields = ("id", "name",)


alezhuq
class PartialServiceSerializer(serializers.ModelSerializer):
    alezhuq
    class Meta:
        model = Service
        fields = ['name', 'description', 'price']


alezhuq
class ServiceSerializer(serializers.ModelSerializer):
    alezhuq
    class Meta:
        model = Service
        fields = ['id', 'name', 'description', 'price']


alezhuq
class CompanySerializer(serializers.ModelSerializer):
    city = PartialCitySerializer(many=True, required=False)
    service = PartialServiceSerializer(many=True, required=False)

    alezhuq
    class Meta:
        model = Company
        fields = ['id', 'name', 'description', 'website', 'email', 'phonenum', 'photo', 'city', 'service']
```

Рис. 17 - реалізація класів-серіалізаторів

```
import django_heroku
import dj_database_url

django_heroku.settings(locals())
del DATABASES['default']['OPTIONS']['sslmode']
```

```
web gunicorn backend.wsgi:application --log-file -
```

Рис. 18 - підготовка налаштувань та створення procfile для хостингу на платформу heroku

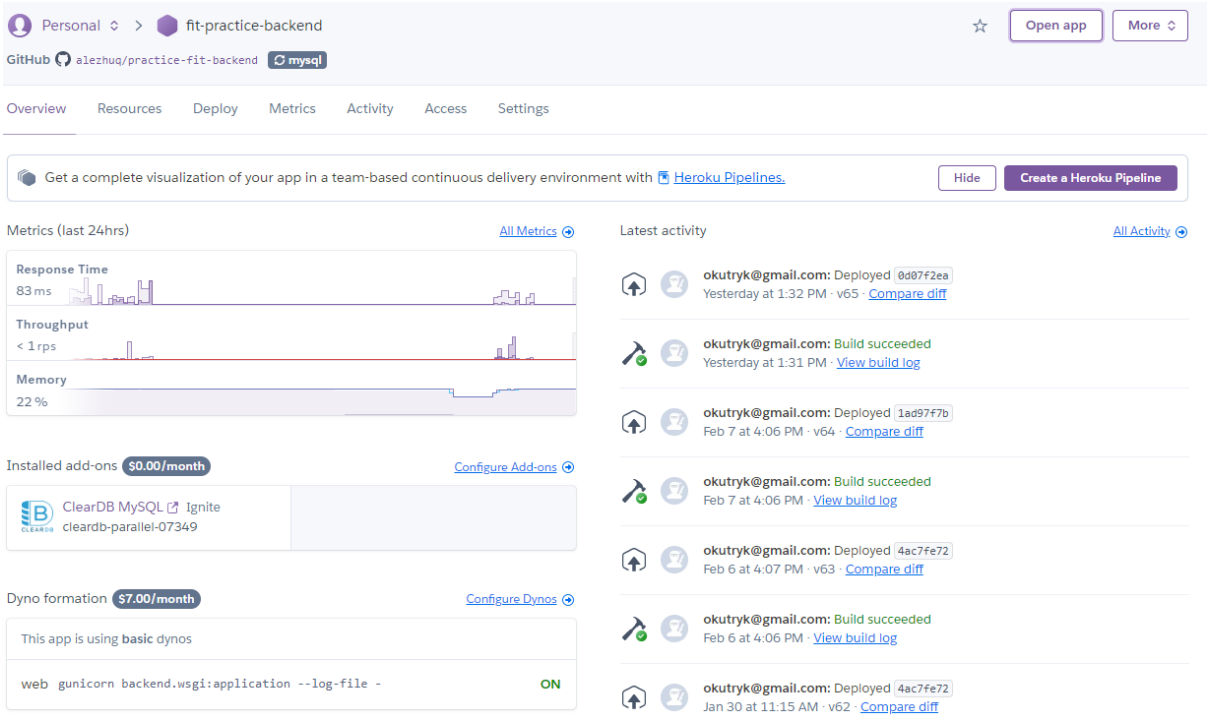

Рис. 19 - сервер, розміщений на хостингу Heroku

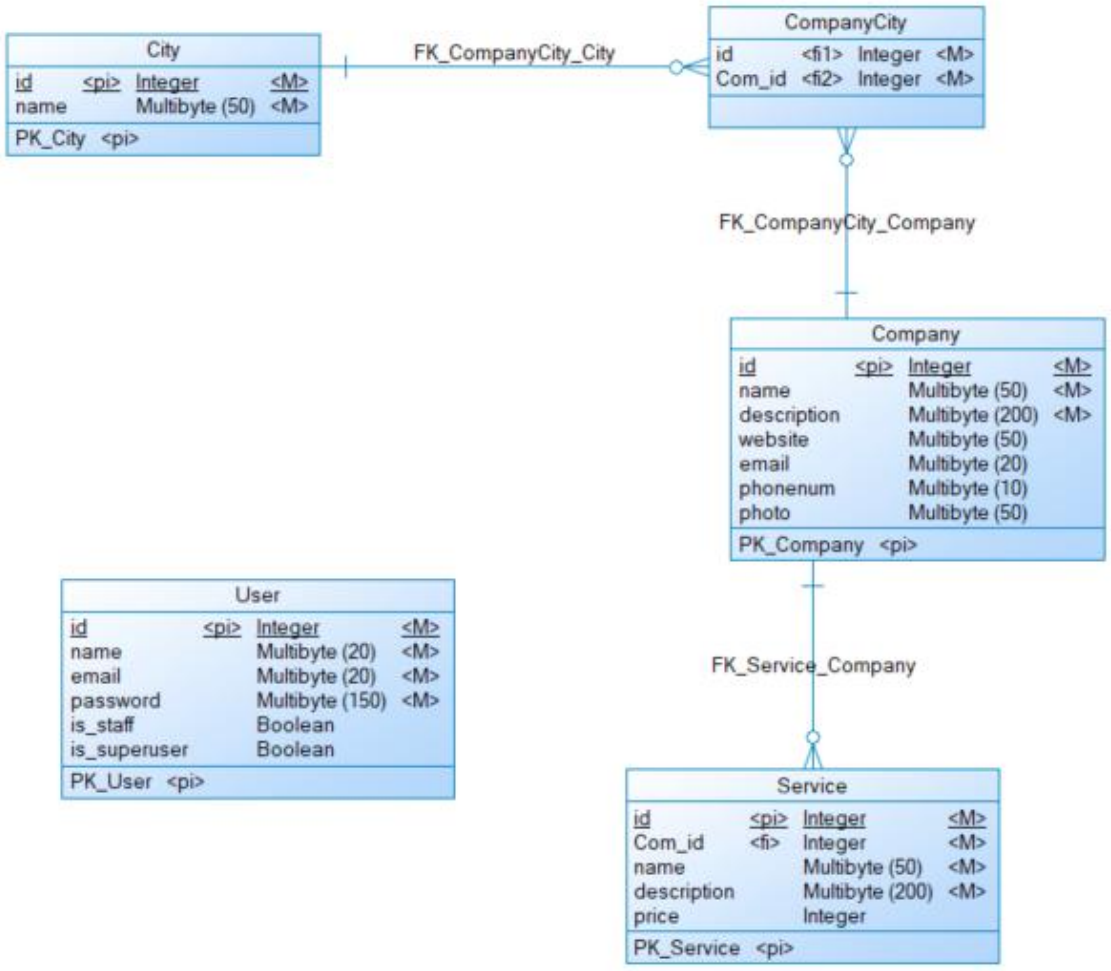

Рис. 20 -Даталогічна модель бази даних

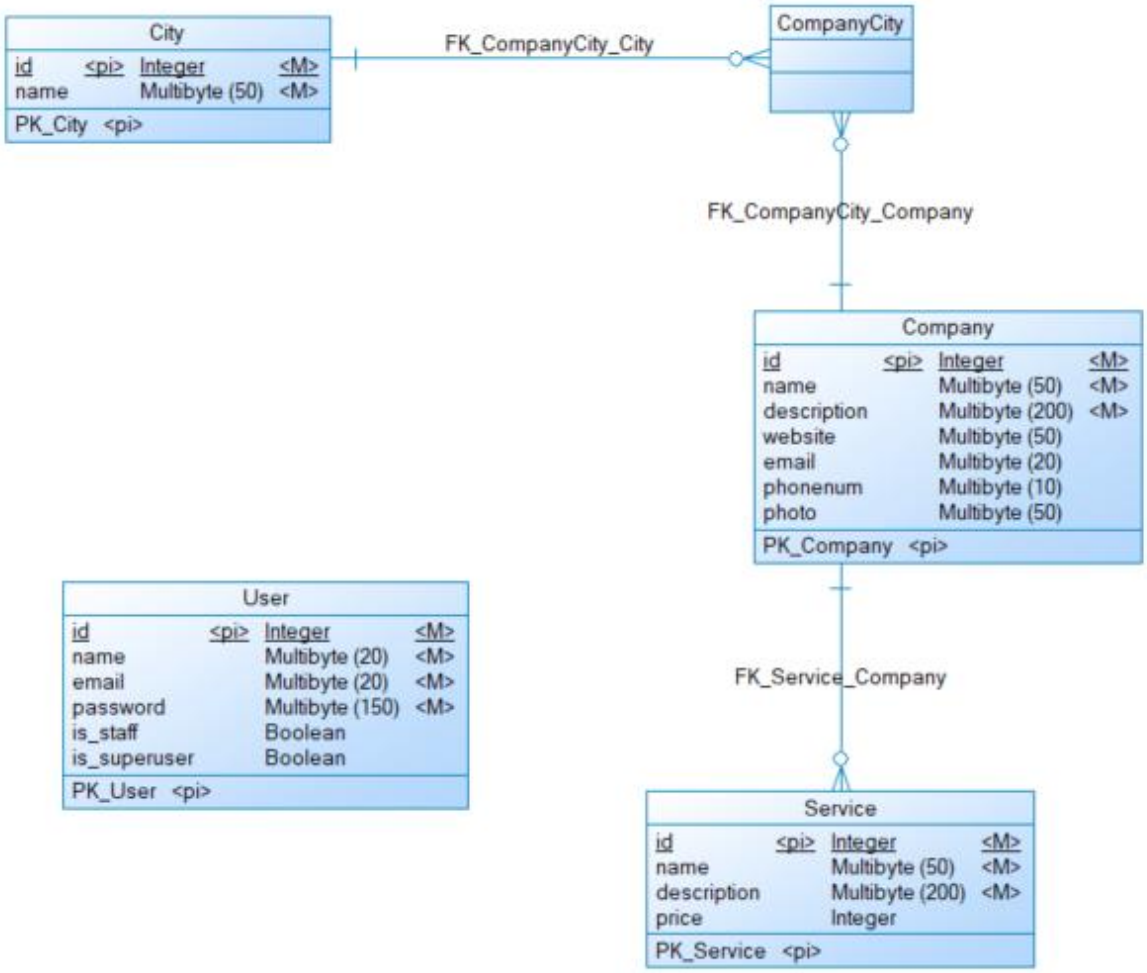

Рис. 21 -Концептуальна модель бази даних

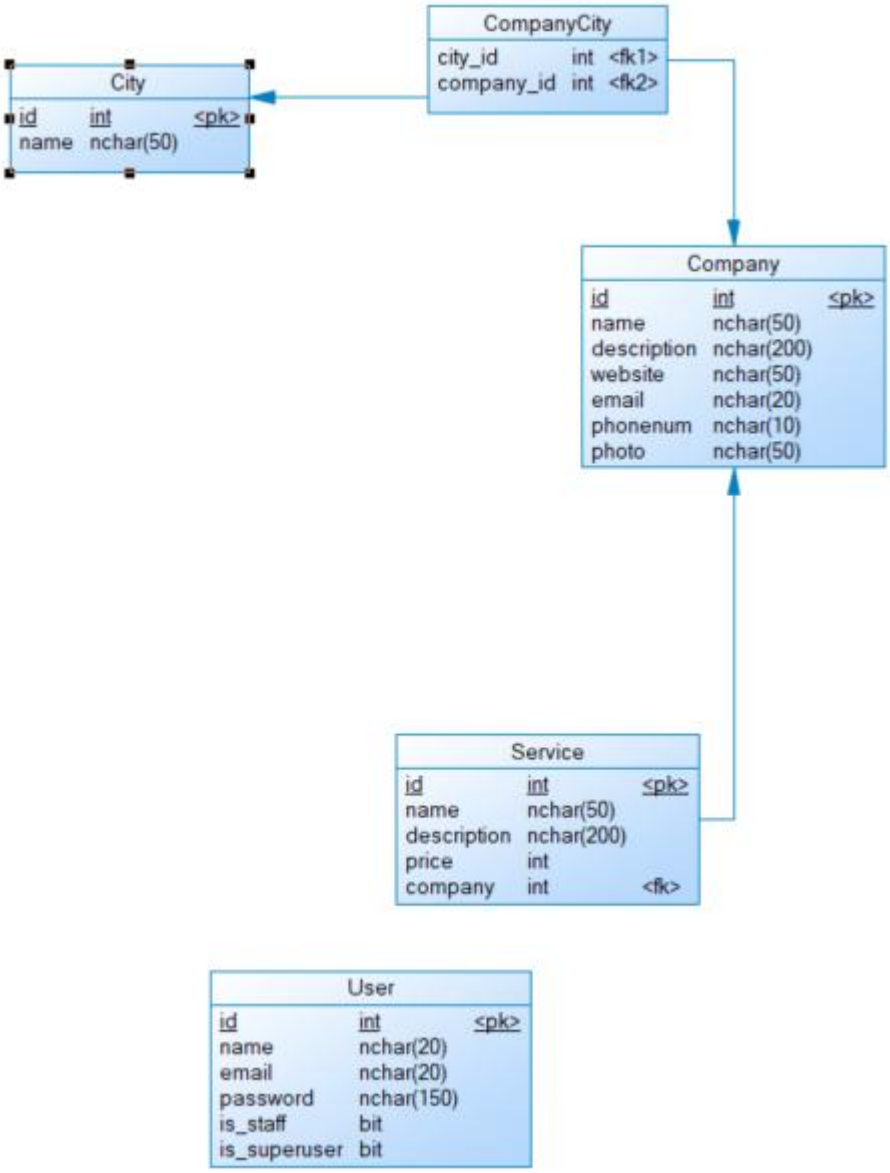

Рис. 22 -Фізична модель бази даних

Для створення таблиць було використано вбудований інструмент migrations (міграції). На даний час це краща альтернатива прямим змінам в СУБД, яка має свої переваги.

Цей інструмент дивиться у файл models та , з допомогою команди python manage.py makemigrations, генерує міграційний файл, якщо побачить зміни. Далі, з допомогою цих файлів можна змінити структуру бази даних, виконавши команду python manage.py migrate.

З допомогою міграцій можна зберігати всі зміни бази даних, і за потреби перемикатись між ними

Створені моделі :

```
class Company(models.Model):
  name = models.CharField(max_length=50, unique=True)
  description = models.CharField(max_length=200)
  website = models.CharField(max_length=50)
  email = models.EmailField(max_length=20, unique=True)
  phonenum = models.CharField(max_length=15)
  photo = models.ImageField(upload_to='photos/%Y/%m/%d/', blank="True")
  city = models.ManyToManyField("City", related_name='city')


class Service(models.Model):
  company = models.ForeignKey(Company, on_delete=models.CASCADE,
related_name='service')
  name = models.CharField(max_length=50)
  description = models.CharField(max_length=200)
  price = models.IntegerField()


class City(models.Model):
  name = models.CharField(max_length=50)
```

Приклад створеної міграції

```
class Migration(migrations.Migration):

    initial = True

    dependencies = [
    ]

    operations = [
        migrations.CreateModel(
            name='City',
            fields=[
```

```
            ('id', models.BigAutoField(auto_created=True, primary_key=True,
serialize=False, verbose_name='ID')),
            ('name', models.CharField(max_length=50)),
        ],
    ),
    migrations.CreateModel(
        name='Company',
        fields=[
            ('id', models.BigAutoField(auto_created=True, primary_key=True,
serialize=False, verbose_name='ID')),
            ('name', models.CharField(max_length=50)),
            ('description', models.CharField(max_length=200)),
            ('website', models.CharField(max_length=50)),
            ('email', models.CharField(max_length=20)),
            ('phonenum', models.CharField(max_length=15)),
            ('photo', models.ImageField(blank='True',
upload_to='photos/%Y/%m/%d/')),
            ('city', models.ManyToManyField(to='CompanyCatalogue.city')),
        ],
    ),
    migrations.CreateModel(
        name='Service',
        fields=[
            ('id', models.BigAutoField(auto_created=True, primary_key=True,
serialize=False, verbose_name='ID')),
            ('name', models.CharField(max_length=50)),
            ('description', models.CharField(max_length=200)),
            ('price', models.IntegerField()),
            ('company',
models.ForeignKey(on_delete=django.db.models.deletion.CASCADE,
to='CompanyCatalogue.company')),
        ],
    ),
]
```

Сайт розміщено на веб-хостингу Heroku за посиланням :

https://fit-practice-backend.herokuapp.com/company/

Висновок : Під час проходження практики, за три тижні, було досліджено підприємство ТОВ «АДВАНСЕД СОФТВЕР СОЛЮШНС».

Для створення готового програмного продукту було використано наступні технології :

- Django
- MySQL
- Heroku

На практиці я ознайомився з реалізацією REST api, підготовкою та деплойментом програмного продукту на веб-хостинг.