

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ПОЛІТЕХНІЧНИЙ УНІВЕРСИТЕТ

Протокол

Лабораторна робота №10

На тему: ““Организация работы с базой данных. CRUD”

По предмету: “Інженерія програмного забезпечення”

Виконав:

студент групи АМ-182

Борщов М. І.

Перевірила:

Шапоріна О.Л.

Одеса 2021

Перелік завдань до лабораторної роботи

1. На основе спроектированной в предыдущих лаб. Работах данных разработать непосредственно само приложение.

Хід роботи

Предметна область - Проектирование приложения для личного кабинета клиента банка.

Для написания программы использовался язык программирования Python3.9 и IDE PyCharm

```
PS C:\Users\kolya\Documents\GitHub\onpu\4_year\ing_prog_zab\lab_10> python manage.py startapp bank_structure
PS C:\Users\kolya\Documents\GitHub\onpu\4_year\ing_prog_zab\lab_10> 
```

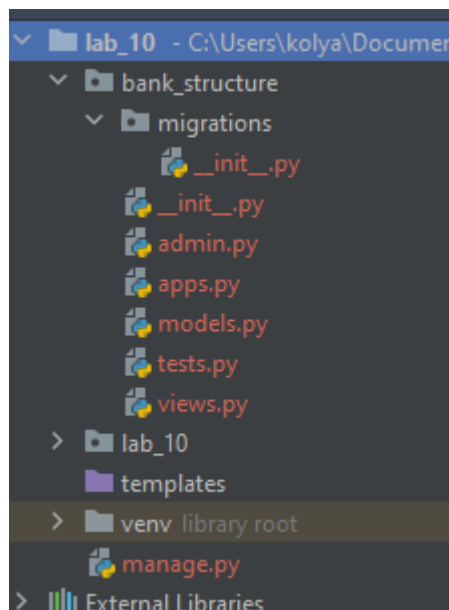


Рис. 1. Создание приложения и Django проекта

```

class Transaction(models.Model):
    sender = models.ForeignKey(
        Client,
        on_delete=models.CASCADE,
        verbose_name="Money Sender",
    )
    receiver = models.ForeignKey(
        CreditCard,
        on_delete=models.CASCADE,
        verbose_name="Money Receiver",
    )
    value = models.BigIntegerField(
        default=0,
        verbose_name='Transaction money value',
    )
    status = models.CharField(
        default=TransactionStatus.UNKNOWN_ERROR,
        max_length=15,
        verbose_name='Transaction status'
    )
    date = models.DateTimeField(
        auto_now_add=True,
        blank=True,
        verbose_name='Transaction date and time',
    )

```

Рис. 2. Класс транзакции

Django является мощным фреймворком для работы с базами данных, и в нём существует базовая модель пользователя, которая имеет все элементы класса Person, Client, Administrator

```

class Client(models.Model):
    user = models.ForeignKey(
        User,
        on_delete=models.CASCADE,
        verbose_name="Client",
    )

    def __str__(self):
        return self.user.username

```

Рис. 3. Класс клиента и Администратора

```

class Question(models.Model):
    question = models.TextField(
        verbose_name="Client Issue text",
        blank=True,
    )
    answer = models.TextField(
        verbose_name="Admin Answer text",
        blank=True,
    )
    state = models.CharField(
        default=QuestionState.NOT_SOLVED,
        max_length=15,
        verbose_name='Transaction status'
    )
    author = models.ForeignKey(
        Client,
        on_delete=models.CASCADE,
        verbose_name="Question Author",
    )
    receiver = models.ForeignKey(
        Admin,
        on_delete=models.CASCADE,
        verbose_name="Who answered question",
    )
    date = models.DateTimeField(
        auto_now_add=True,
        blank=True,
        verbose_name='Question date and time',
    )

```

```

def stringed_date(self) → str:
    return self.date.strftime(DATE__FORMAT)

def update_state(self, state: QuestionState, admin: Client):
    self.state = state
    self.changed_by = admin

def update_answer(self, answer: str, admin: Client):
    self.answer = answer
    self.changed_by = admin

def print_question(self):
    print(f'\nВопрос: {self.question}'
          f'\nОтвет: {self.answer}'
          f'\nСтатус: {self.state}'
          f'\nАвтор: {self.author}'
          f'\nОтветил: {self.receiver}'
          f'\nДата: {self.stringed_date()}')

```

Рис. 4. Класс вопроса

```

class CreditCard(models.Model):
    number = models.CharField(
        max_length=16,
        default=generate_random_card_number(),
    )
    cvv2 = models.PositiveSmallIntegerField(
        default=generate_cvv_2()
    )
    owner = models.ForeignKey(
        Client,
        on_delete=models.CASCADE,
    )
    expiration_date = models.DateTimeField(
        blank=True,
        verbose_name='Card date and time',
    )
    balance = models.BigIntegerField(
        verbose_name='Card Balance',
    )

    def convert_float_to_int(self, value: float = None) → int:
        if not value:
            value = self.balance
        return math.ceil(value * 100)

    def convert_money_to_float(self) → float:
        return self.balance / 100

    def convert_money_to_str(self) → str:
        return str(self.convert_money_to_float())

    def hide_number(self) → str:
        return self.number[:4] + '*' * 8 + self.number[-4:]

    def show_balance(self) → Union[float, str]:
        return self.convert_money_to_str()

    def __str__(self):
        return self.hide_number()

```

Рис. 5. Класс кредитной карты

Django поддерживает большое кол-во СУБД, чаще всего используются или sqlite3 или PostgreSQL. В данном случае стоит значение по умолчанию - sqlite3.

```

# Database
# https://docs.djangoproject.com/en/3.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}

```

Рис. 6. Настройка БД

```

    Борщев, 31 minutes ago • init
    INSTALLED_APPS = [
        'django.contrib.admin',
        'django.contrib.auth',
        'django.contrib.contenttypes',
        'django.contrib.sessions',
        'django.contrib.messages',
        'django.contrib.staticfiles',
        'bank_structure'
    ]

```

Рис. 7. Подключение нашей структуры к джанго

Для создания таблиц для моделей необходимо подготовить модели и данные о них к записи в бд. Для эго создаются файлы миграций

```

PS C:\Users\kolya\Documents\GitHub\onpu\4_year\ing_prog_zab\lab_10> python manage.py makemigrations
Migrations for 'bank_structure':
  bank_structure\migrations\0001_initial.py
    - Create model Admin
    - Create model Client
    - Create model CreditCard
    - Create model Transaction
    - Create model Question
PS C:\Users\kolya\Documents\GitHub\onpu\4_year\ing_prog_zab\lab_10>

```

Рис. 8. Создание миграций

После чего успешно принимаются и создаются структуры в СУБД

```

PS C:\Users\kolya\Documents\GitHub\onpu\4_year\ing_prog_zab\lab_10> python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, bank_structure, contenttypes, sessions
Running migrations:
  Applying bank_structure.0001_initial... OK
PS C:\Users\kolya\Documents\GitHub\onpu\4_year\ing_prog_zab\lab_10>

```

Рис. 9. Успешная миграция всех изменений

```

PS C:\Users\kolya\Documents\GitHub\onpu\4_year\ing_prog_zab\lab_10> python manage.py createsuperuser
Username (leave blank to use 'kolya'): admin
Email address:
Password:
Password (again):
The password is too similar to the username.
This password is too common.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
PS C:\Users\kolya\Documents\GitHub\onpu\4_year\ing_prog_zab\lab_10>

```

Рис. 10. Создание админимстратора

```
superuser created successfully.  
PS C:\Users\kolya\Documents\GitHub\onpu\4_year\ing_prog_zab\lab_10> python manage.py runserver  
Watching for file changes with StatReloader  
Performing system checks...  
  
System check identified no issues (0 silenced).  
December 03, 2021 - 06:51:01  
Django version 3.2.9, using settings 'lab_10.settings'  
Starting development server at http://127.0.0.1:8000/  
Quit the server with CTRL-BREAK.
```

Рис. 11. Запуск Джанго

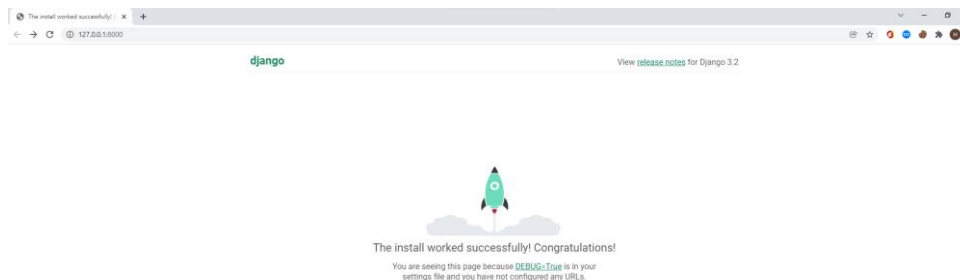


Рис. 12. Джанго работает

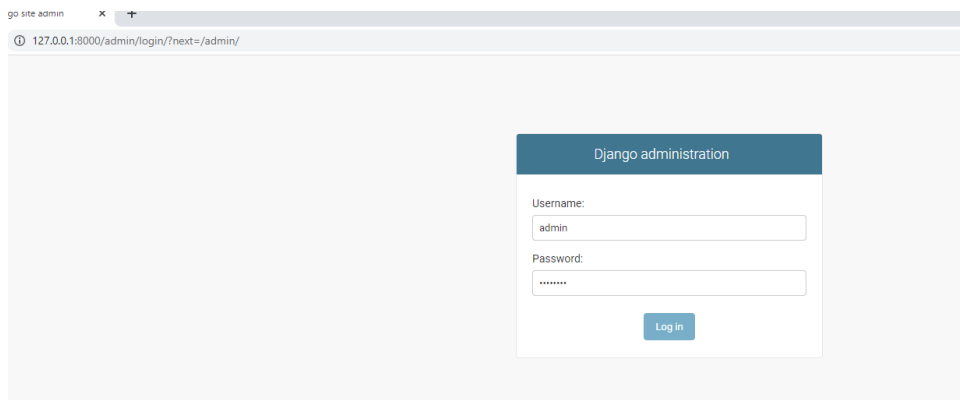


Рис. 13 вход в адним-панель

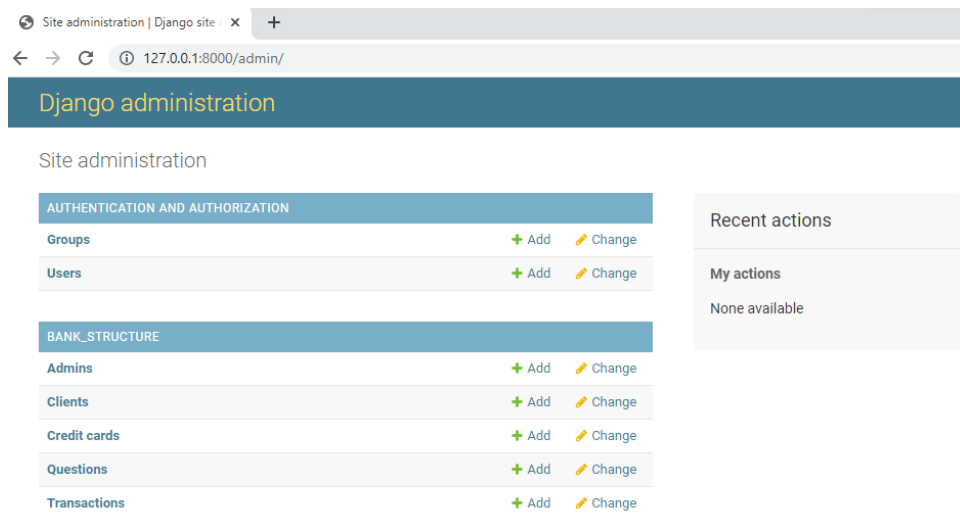
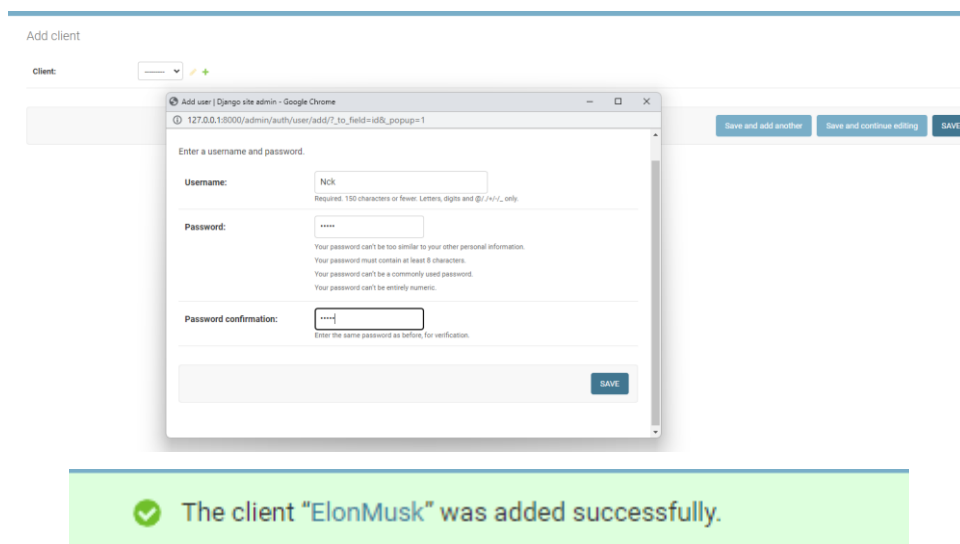


Рис. 14. Админ-панель



Select client to change

Action: 0 of 2 selected

<input type="checkbox"/>	CLIENT
<input type="checkbox"/>	ElonMusk
<input type="checkbox"/>	Nck

2 clients

Рис. 15 Создание клиента

Add credit card





Number:	<input type="text" value="517840549596934"/>
Cvv2:	<input type="text" value="889"/>
Owner:	<div><div>Nck</div><div>▼</div><div> </div></div>
Card date and time:	<div><div>Date:</div><div><input type="text" value="2022-03-18"/></div><div>Today </div></div> <div><div>Time:</div><div><input type="text" value="05:01:52"/></div><div>Now </div></div> <div>Note: You are 2 hours ahead of server time.</div>
Card Balance:	<input type="text" value="12300"/>

Рис. 16 Создание карты

Рис. 17. Созлание транзакции

Описание программы

В данной работе была создана база данных в фреймворке Django. Реализованы все кассы для пользовательского приложения для пользования электронными средствами.

Так же сделана конвертация валюты из дробного числа (гривны и копейки) в целое (копейки) ля ускорения работы транзакций и оптимизации БД.

Код написан согласно архитектурному стилю MVC и согласно стандартам PEP8 для языка программирования Python3

Исхоный код программызагружен на гитхаб репозиторий:
https://github.com/zeinlol/onpu/tree/master/4_year/ing_prog_zab/lab_10

Висновок:

В результаті виконання даної лабораторної роботи я ознайомився з процесом створення Django серверу, написав працюючий код та ознайомився з архітектурним стилем MVC. Були створені різноманітні моделі та зв'язки між ними.