МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет комп'ютерних наук, фізики та математики
Кафедра комп'ютерних наук та програмної інженерії

ПРОЄКТУВАННЯ WEB-ДОДАТКУ «ВЕДЕННЯ ТА СИСТЕМАТИЗАЦІЯ БІЗНЕСУ» НА PYTHON DJANGO

Кваліфікаційна робота
на здобуття ступеня вищої освіти «бакалавр»

Виконавець: студент 4 курсу 441групи
Спеціальність 121 Інженерія програмного забезпечення
Освітньо-професійної програми:
Інженерія програмного забезпечення
Новиков О. О.
Керівник: старший викладач Черненко І.Є.
Рецензент:

_____

Херсон – 2022

ЗМІСТ

ВСТУП 3

РОЗДІЛ 1. Історія та архітектура Python та його фреймворку Django 4

1. Історія створення мови програмування Python 4
2. Характеристики та особливості Python 7
3. Історія створення Django 15
4. Особливості та структура Django 16

РОЗДІЛ 2. Поєктування клієнт-серверного додатку 19

Розробка технічного завдання 19

Розробка макету сайту 24

РОЗДІЛ 3. Програмування клієнт-серверного додатку 30

ВИСНОВКИ 41

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 42

ВСТУП

Побудова системи обліку для ведення справи на початковому етапі створення бізнесу є необхідним для кожної компанії. Зручний і багатофункціональний інструмент з адміністрування справи дозволить у певній мірі автоматизувати фінансові та облікові процеси, що у свою чергу збільшить час для креативного розвитку справи. Представлена перевага є дуже актуальна у сучасному малому інтернет-бізнесі, що обумовлено дефіцитом кадрів на перших етапах підприємницької діяльності.

Метою даної роботи є створення веб-додатку з адміністрування та ведення бізнесу у сфері інтернет продажів малого бізнесу для часткової автоматизації бухгалтерських та облікових процесів, підтримання та керування асортиментом товарів, керування списком замовлень, переліком клієнтів та співробітників компанії.

До списку поставлених завдань входить:

1. вивчити історію створення та архітектуру Python, а також його фреймворку Django;

2. проєктування клієнт-серверного додатку;

3. розробка додатку з використанням фреймворку Python Django.

Об’єктом дослідження даної курсової роботи виступає web-сайт, як спосіб відображення інформації.

Предметом дослідження є клієнт-серверний веб-додаток націлений на адміністрування бізнесу з виготовлення і продажу товарів.

РОЗДІЛ 1

ІСТОРІЯ ТА АРХІТЕКТУРА PYTHON ТА ЙОГО ФРЕЙМВОРКУ DJANGO

1.1 Історія створення мови програмування Python

Комп’ютерна програма у спрощеному вигляді – це набір послідовних дії, виконання яких приведе до поставленої мети.

Кожна людина протягом дня виконує певні дії, які можна представити у вигляді послідовності команд. Виходячи з цього, можна сказати, що людина біо-комп’ютер який виконує алгоритми запрограмовані на біологічному рівні. Тут виникає проблема «розуміння» комп’ютерами (електронними обчислювальними машинами) природньої, людської мови програмування. Цю проблему було вирішено запровадженням машинної мови програмування, яка складається з цифр.

Спочатку програмування відбувалося з використанням цифр, проте такий спосіб виявився незручним з декількох причин:

- цифрова форма запису не є природньою для людей;
- опис елементарного алгоритму буде складатися з великого об’єму цифрових значень.

Висвітлені проблеми потрібно було вирішувати. На першому етапі вирішення проблеми було створено спеціальні мови програмування-асемблери (мови з словесно-літерним позначенням команд). Надалі було розроблено мови високого рівня, які були більш функціональні та прості для сприйняття людини. Але питання переводу зрозумілої мови програмування для людини у машинний код залишалося відкритим. У цей час почалася розробка трансляторів – програм, для перекладу

програмного коду з мови програмування у машинний код. Виділяється два способи трансляції – інтерпретація програми або компіляція.

Компіляція передбачає створення окремого файлу не пов’язаного з вихідним кодом, який містить вже перетворену у машинний код інформацію. Виконання файлу забезпечує операційна система.

Трансляція коду з використанням інтерпретатора проходить рядок за рядком (послідовно). Операційна система читає код з використанням посередництва інтерпретатора.

Виконання програми, яка була оброблена компілятором, буде відбуватися швидше (компілятор переписує програму у машинний код). З огляду на можливості сучасних комп’ютерів зниження швидкості виконання програм оброблених інтерпретатором нівелюється обчислювальними можливостями комплектуючих частин. [1]

Розробником мови програмування Python є Нідерландський програміст Гвідо Ван Россум. Ідея створення Python-а зародилася у 1980-х роках, проте реалізація мови програмування почалася у грудні 1989 року. Виконання проєкту проходило у Національному науково-дослідному інституті математики та інформатики.

Назва мови виникла під впливом та на честь британського комедійного шоу «Літаючий цирк Монті-Пайтона», який виходив на початку 1970-х років.

Перша публікація нової мови програмування Гвідо Ван Россума відбулася у лютому 1991 року. Вже тоді, перша версія мови Python (0.9.0) мала достатньо великі функціональні можливості такі як обробка винятків на основі мови Модула-3, класи з успадкуванням, основні типи даних: str, list, dict. [4]

На початку 1994 року була опублікована версія Python 1.0. У представленій версії мови були додані нові можливості, зокрема засоби

функціонального програмування: функції вищих порядків, згортка списку, лямбда-числення та інші.

Версія 1.2 була наступною. Це була остання розробка Ван Россума, яка була випущена за час його роботи у центрі математики та інформатики.

Починаючи з 1995 року Гвідо Ван Россум продовжив роботу над мовою програмування Python. У цей період він працював у корпорації національних дослідницьких ініціатив (CNRI) у містечку Рестон. За час роботи у CNRI було випущено декілька версій мови Python. Також саме там зародилася ідея запуску проєкту «Програмування для всіх». Метою ініціативи було поширення можливості отримання базової освіти з комп'ютерної грамотності (по аналогії з базовим знанням мови та математики), що у подальшому дозволить вивести програмування у широкі маси. Центральною мовою програмування у освіті виступав Python, завдяки своїй спрямованості на ясний синтаксис. Наразі проєкт закритий.[4]

Починаючи з 2000-х років велика частина команди розробників приєдналась до BeOpen.com, де було організовано нову команду BeOpen Python Lab. Командою було заплановано реліз версії 2.0, однак цього не сталося. Корпорація CNRI наполягла на випуску версії, яка буде містити усі напрацювання, створені розробниками під час роботи у фірмі. Тоді було випущено версію 1.6 з угодою від корпорації національних дослідницьких ініціатив. Але створена угода суперечила ліцензії на вільне програмне забезпечення (GNU GPL). Тоді було прийняте спільне рішення фонду вільного програмного забезпечення, корпорації національних дослідницьких ініціатив та BeOpen Python Lab на зміну поточної ліцензії та випуску версії 1.6.1, що включала незначні доопрацювання.

16 жовтня 2000 року на BeOpen.com було опубліковано Python 2.0. Нововведеннями цієї версії стали спискові включення, запозичені з функціональних мов програмування та системи складання сміття.[4] Згодом було створено версії 2.1 і 2.2, у які додали генератори та провели об'єднання в одній ієрархії всіх базових типів і класів, завдячуючи чому Python став об'єктно орієнтованою мовою програмування.

Новий виток у розвитку мови відбувся 3 грудня 2008, тоді було випущено Python 3.0 (версії 2.6, 2,7 розвивалися паралельно). У новій версії мови було усунуто попередні недоліки. При цьому підтримувалася зворотня сумісність з версіями 2.x. Також була усунута важлива проблема дублювання конструкцій які вирішували однакові задачі.

З подальшим розвитком мови та збільшенням змін (додавання функцій, динамічна типізація, перехід на Unicode для радків та ін.) ускладнювався процес автоматичної трансляції версії 2 до 3. Проблему було вирішено за допомогою інструменту під назвою «2To3», який швидко та якісно виконує роботу з перекладу. [4]

1.2 Характеристики та особливості Python

Python - високорівнева мова програмування загального призначення, що має великий стандартний набір функцій і підтримує кілька парадигм програмування. [2]

У розробників які використовують мову програмування Python є певна філософія, автором якої є Тім Петерс. Її зміст можна переглянути, запустивши інтерпретатор і ввівши в ньому команду import this (рис 1.1).

```
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
PS C:\Users\Новиков Олег\Desktop\1>
```

Рисунок 1.1 - Філософія мови програмування Python

Загалом, розробники націлені на красу, явність та простоту коду.

Філософія Тіма Петерса включає багато фундаментальних засад мови Pethon:

1. «Речі мають бути настільки простими, наскільки це можливо, але не більше того»;
2. «Складне краще, ніж заплутане.»;
3. «Не робіть ідеального, бо «достатньо хороший» це інколи саме те, що треба»;
4. «Не переймайтеся досконалістю – оптимізуйте, коли це буде необхідно».

Python є інтерпретованою мовою програмування, яка має інтегроване середовище розробки (IDLE). Наявність інтерактивного режиму у IDLE є дуже зручним доповненням для тестування різноманітних конструкцій, частин коду, що є дуже важливим для розробників будь-якого рівня. Написання коду програми також можливе у будь-якому текстовому редакторі, але важливо правильно зазначати формат файлу при його збереженні. Після запуску програми її виконання буде відбуватися у інтерпретаторі.

Помилки. Суттєвою особливістю мови програмування Python є автоматичне підсвічування синтаксису. Це дозволяє контролювати правильність написання вбудованих команд, функцій.

Виникнення помилок при написанні коду- це звична справа. Вони можуть бути різними:

5. синтаксична помилка - це неточність в синтаксисі кодування, введена програмістом;
6. помилка виконання програми - виникає у разі виявлення інтерпретатором вийняткової ситуації;
7. семантична помилка – виникає при застосуванні оператора, який не дає очікуваного результату, або похибка у структурі алгоритму та ін.

Помилки синтаксису виявляються програмою, яка називається компілятором, і програміст повинен виправити їх до того, як програма буде скомпільована, а потім запущена.

Помилки які виникають під час виконання програми коректніше називати винятками. Вони допомагають при знаходженні причини помилки яка виникла в ході роботи програми. (рис. 1.2).

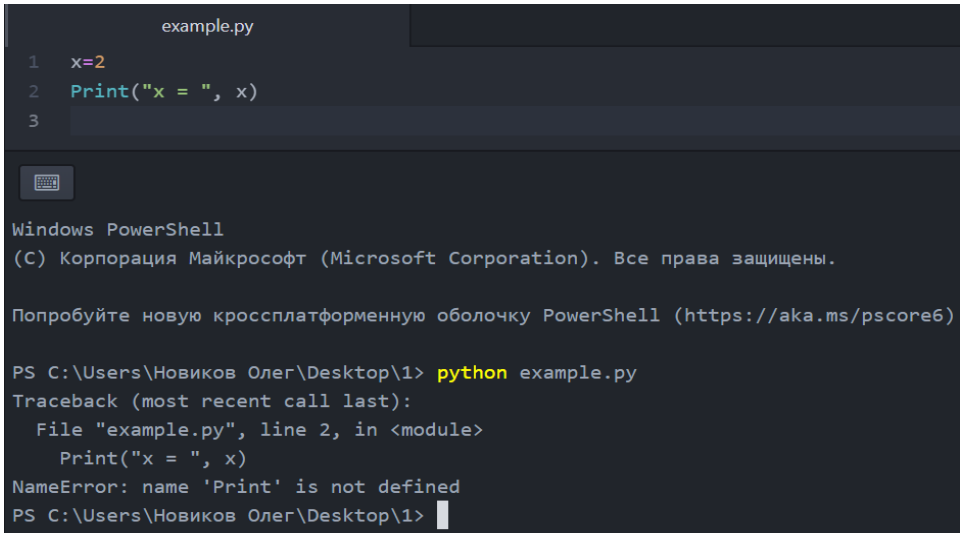

Рисунок 1.2- Виняток «NameError»

У представленому випадку зображено виникнення винятку "NameError" – не знайдено змінну/функцію з таким ім’ям. Другий рядок сповіщення про виняток вказує на рядок з помилкою.

Основними винятками у Python є "NameError", "ZeroDivisionError" та "TypeError" і вони є стандартними для мови програмування Python. Розробники створили функціонал для створення власних винятків, що дозволяє не припиняти хід виконання програми у разі виникнення передбаченого автором коду випадку.

Змінні. Змінні у Python-і не потребують додаткового опису, визначення відбувається у момент надання їм значення. Тип змінної буде залежати від типу присвоєного значення (рис. 1.3).

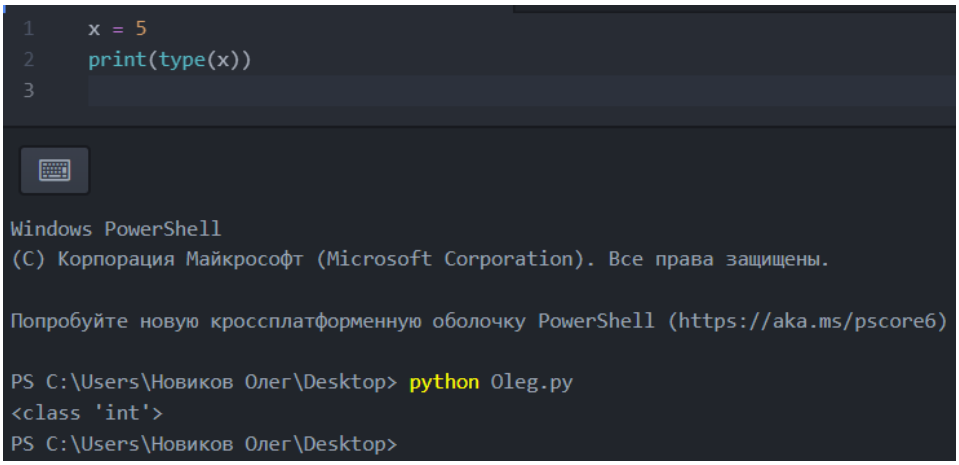

Рисунок 1.3 - Визначення типу змінної під час присвоєння

На рисунку 1.3 зображено процес присвоєння значення 5 змінній x. З огляду на те, що тип значення є числовим – отже і тип змінної також стає числовим. Тип змінної не є постійним значенням, він залежить від типу присвоєного значення. У Python-і говорять , що змінна посилається на об’єкт, який знаходиться у пам’яті комп’ютера. Це означає, що при наданні значення змінній їй надається лише посилання на значення. І в момент виклику функцій їм передається посилання. Цікавим випадком є присвоєння однієї змінної іншій. Тоді нова змінна буде посилатися на ту саму область пам’яті, але вже під іншим ім’ям.

Групування операторів. У мові програмування Python групування елементів коду здійснюється з використанням відступів (рис.1.4)

example.py

```
print("Приклад групування операторів")
age = int(input("Напишіть ваш вік"))
if age < 12:
    print("Залиште сайт")
else:
    print("Оберіть потрібну вам інформацію")
```

Рисунок 1.4 - Групування блоків операторів

У представленому прикладі 1.4 блок if, який містить один рядок коду, зміщений на ширину чотирьох пробілів праворуч.

Зразок вкладених конструкцій (рис.1.5):

example.py

```
print("Приклад групування операторів")
age = int(input("Напишіть ваш вік"))
if age < 12:
    if age > 6:
        print("Для вас є дитяча інформація")
    else:
        print("Залиште сайт")
else:
    print("Оберіть потрібну вам інформацію")
```

Рисунок 1.5 - Приклад вкладених конструкцій

В даному зразку зображено вкладення однієї команди if у іншу таку ж команду. Тут ми можемо бачити два види відступів: для головної конструкції одинарна табуляція, для вкладеної – подвійна.

Цикл for. Цикл for, або цикл з параметром, у мові програмування Python має великі можливості. У циклі вказується змінна, а також множина по якій буде проходитись змінна. Множина значень може бути задана списком, кортежем, рядком, діапазоном.

На малюнку 1.6 зображено приклад використання циклу for, де у якості множини значень використовується діапазон:

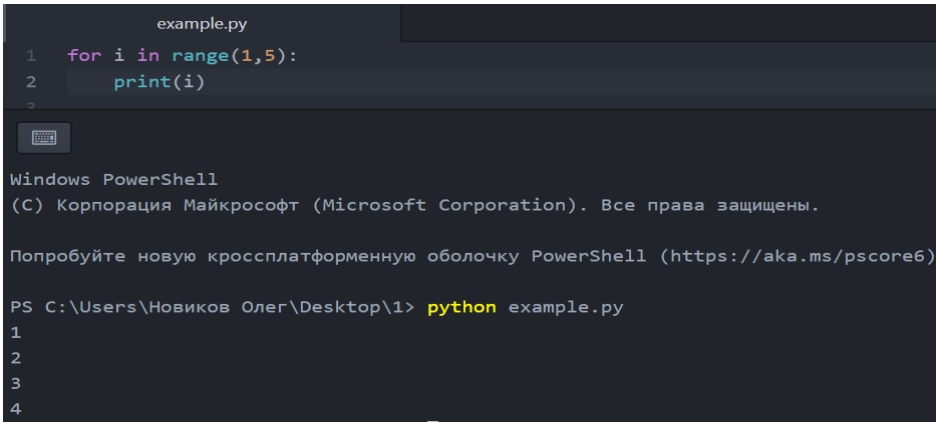

Рисунок 1.6 - Цикл for

У цьому прикладі змінна і послідовно приймає значення діапазону range(1,5). У цьому циклі виводиться повідомлення, яке містить значення діапазону (1,2,3,4).

Списки. У таких мовах програмування, як C++ та C організація списків досить складна. Для використання списків необхідно розумітися на функціонуванні покажчиків та динамічної пам’яті. Таке ускладнення часто спричиняло помилки навіть у розробників з великим досвідом роботи. Для полегшення використання списків були створені спеціальні інструменти.

У Python-і на відміну від зазначених мов не використовується робота з динамічною пам’яттю та покажчиками, що зумовлює його простоту та надійність. [5]

Працювати зі списками досить просто, створюються вони за допомогою квадратних дужок (рис.1.7).

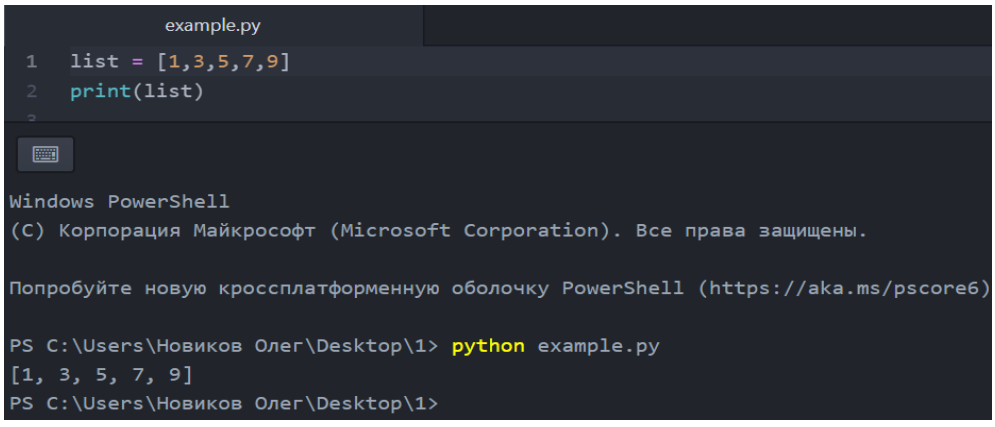

Рисунок 1.7 - Створення списку

У прикладі було створено список list, який містить 5 елементів. Скориставшись функцією print() ми можемо вивести як весь вміст списку, так і окремий його елемент. Для виводу обраного елементу списку слід вказати його порядковий номер у квадратних дужках (рис.1.8).

example.py

```
1  list = [1,3,5,7,9]
2  print(list)
3  print(list[0])
```

```
Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

Попробуйте новую кроссплатформенную оболочку PowerShell (https://aka.ms/pscore6)

PS C:\Users\Новиков Олег\Desktop\1> python example.py
[1, 3, 5, 7, 9]
1
```

Рисунок 1.8 - Вивід окремих елементів списку

Під час виводу елементів списку слід мати на увазі, що індексація починається з нуля, отже максимальний індекс буде мати значення N-1, де N – це кількість елементів списку.

Звернення до елементів списку може здійснюватися з використанням інтервальної задачі індексів, що робиться за допомогою двокрапки (рис.1.9).

example.py

```
1  list = [1,3,5,7,9]
2  print(list)
3  print(list[0:3])
```

```
Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

Попробуйте новую кроссплатформенную оболочку PowerShell (https://aka.ms/pscore6)

PS C:\Users\Новиков Олег\Desktop\1> python example.py
[1, 3, 5, 7, 9]
[1, 3, 5]
PS C:\Users\Новиков Олег\Desktop\1>
```

Рисунок 1.9 - Вивід проміжку списку

При зверненні до елементів списку з використанням інтервалу, ліворуч вказується початок проміжку, а праворуч – кінець. У випадку коли не вказане початкове/кінцеве значення, береться нульовий/максимальний індекс.

Елементи списку у Python-і можуть бути різних типів даних. Така особливість можлива завдяки відсутності опису змінних.

Особлива математична структура, яку можна організувати за допомогою списків - це матриця. Реалізація її досить проста, вкладення списку у список (рис. 1.10).

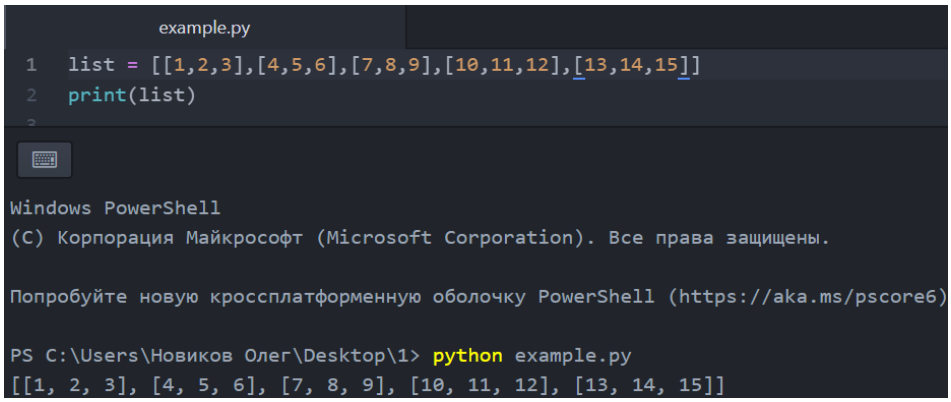

Рисунок 1.10 - Створення матриці

У прикладі кожен елемент списку є списком з власними елементами (рядками матриці).

Цікавою особливістю рядків, кортежів та списків є можливість доступу до елементів з негативним індексом. Це досить корисно у програмуванні, бо з’являється можливість доступу до елементів кінця списку, не знаючи при цьому його довжини.

Функціональність списків забезпечується великою кількістю методів. Перерахуємо найважливіші:

8. append(x), дозволяє вставити елемент у кінець списку;
9. insert(i,x) вставляє елемент x на позицію i;
10. extend(L) вставляє список L в кінець вихідного списку;
11. clear() очищає список;

12. remove(x), видаляє зі списку елемент зі значенням x.

Кортежі. Кортежі досить схожі на списки. Основні відмінності полягають у способі ініціалізації (у круглих дужках), та незмінності значень.

Основною метою кортежів є передача параметрів функціям і унеможливлення зміни їх вмісту іншими функціями.

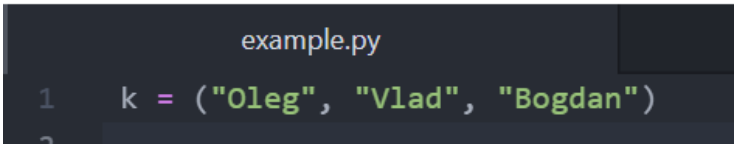

Рисунок 1.11 - Кортеж

У зображеному прикладі з малюнку 1.11 змінній k надається посилання на кортеж з трьох елементів. У разі визначення кортежу, який складається з одного елементу, неодмінно потрібно в кінці ставити кому. У кортежі не дужки, а саме коми надають коректність структурі визначення. [6]

Словники. Словник – це невпорядкована структура даних, яка дає можливість зберігати пари «ключ : значення» (Рисунок 1.12).

Синтаксична побудова словників у певній мірі нагадує списки, але для доступу до елементів застосовуються ключі, а квадратні дужки змінені на фігурні. Доступ до елементів словника здійснюється за допомогою квадратних дужок у які вписується назва необхідного елементу.

Словники є чи не найважливішою структурою даних у мові програмування Python. Більшість об’єктів мови у своїй структурі містять словники. Незважаючи на тип об’єкту та спосіб його використання є висока вірогідність, що у своїй структурі він містить словники, які використовуються для керування атрибутами об’єкта. [6]

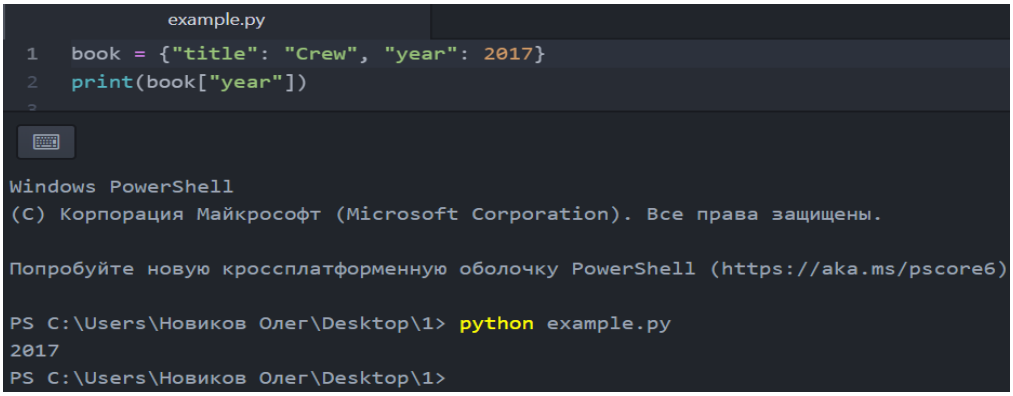

Рисунок 1.12 - Створення та доступ до елементів словника

1.3 Історія створення Django

Еволюція Django відбувалася природним шляхом, у ході програмування реальних веб-додатків, розроблених командою програмістів в Лоуренсі, США. Зародження фреймворку відбулося у 2003 році, коли команда програмістів газети «Lawrence Journal-World», Сімон Віллісон та Едріан Холоваті почали у свої роботі використовувати мову програмування Python. В умовах скороченого часу спричиненого особливістю журналістської професії, команда з розробки та підтримки новинних сайті World Online займала лідируючі позиції у середовищі створення новинних інтернет ресурсів. Зростання вимог щодо швидкості впровадження нових цілей та можливостей програми вимагали запровадження та втілення нових концепцій. У цей час Віллісон і Холоваті втілюють усі вимоги актуальні у журналістській професії у нове середовище розробки, яке спроможне створювати інформаційні програми з можливістю їх динамічної зміни у стислі терміни.

Розвиток середовища не зупинився і влітку 2005 році більшість сайтів World Online було розроблено з використанням раніше згаданого інструменту, тоді було вирішено створити середовище у вигляді програмного забезпечення з відкритим вихідним кодом. Воно було

опубліковано у липні 2005 року під назвою Django (назву було обрано на честь гітариста Джанго Рейнхарда).

Наразі, Django це програмний каркас з відкритим вихідним кодом, який має широкий спектр можливостей у розробці складних сайтів та веб-додатків. Представлений фреймворк має десятки тисяч користувачів, та розробників які працюють над його удосконаленням.

З команди World залишилося два автори, які досі здійснюють загальне керівництво над розвитком проєкту, але у більшій мірі задачі по удосконаленню середовища розробки виконує широкий загал програмістів з усього світу.[7]

1.4 Особливості та структура Django

Основною задачею Django є швидка розробка веб-додатків різного рівня складності. Зважаючи на те, що фреймворк зародився у новинному середовищі, він має інструментарій для розробки та підтримки контент-орієнтованих веб-джерел, які оперують динамічною інформацією з бази даних.

Культура спільноти відкритого вихідного коду була сформована під впливом витоків фреймворку. Django не розроблявся як академічний чи комерційний проєкт, а був отриманий з реального коду, тому він повністю спрямований на вирішення проблем програмування, з якими стикалися його автори. Розробники проєкту мали свій інтерес у швидкісному та якісному створенні додатків з використанням фреймворку, що заощаджувало їх час. Не дивлячись на досить простий процес програмування, розроблені додатки добре працюють під навантаженням та легкі в обслуговуванні. Відкритий вихідний код

зумовлює постійний розвиток продукту за участю програмістів, які бажають приймати участь в удосконаленні середовища розробки.[7]

Django дозволяє писати програмне забезпечення, яке буде:

13. повним: фреймворк надає повний функціонал, який не потрібно доповнювати сторонніми програмами, все відповідає послідовним принципам проєктування та має велику і актуальну документацію;
14. різностороннім: Django може бути використаний для створення майже будь-якого типу веб-сайту – від системи керування контентом до соціальних мереж;
15. безпечним: Django допомагає розробникам уникнути багатьох поширених помилок безпеки, має функціонал для автоматичного захисту сайту;
16. масштабованість: фреймворк використовує компонентну архітектуру (кожна її частина незалежна від інших, тому може бути зміненою, або заміненою, якщо це необхідно);
17. зручний у супроводі: код Django написаний з використанням принципів і шаблонів проєктування, які заохочують створення коду, який можна повторно використовувати;
18. переносним: Django написаний на Python, який працює на багатьох платформах.

Кожен Django-додаток містить чотири базові компоненти:

1. Модель даних: основою будь-якого веб-додатку є дані. Однією з найважливіших частин програми, яка безупинно оперує даними являється модель, яка у свою чергу є стандартним класом мови Python. Використання об’єктно-орієнтованого мапера (ORM) надає класам безпосередній доступ до бази даних (БД);

2. Представлення: view у фреймоврці виконує різні задачі, зокрема контролюють запити користувача, видаючи результат у залежності від його статусу. Представлення – це функція, яка викликається під час запиту певної адреси і повертає результат (контекст);
3. Шаблон: шаблон є конструкцією у відповідності з якою виводяться дані. Вони мають спеціальну метамову і є основними засобами для виводу інформації на екран;
4. URL: url є механізмом зовнішнього доступу до представлень. Регулярні вирази вбудовані у url додають гнучкості механізму. Також слід знати, що одне представлення може бути налаштованим на кілька url, надаючи доступ декільком додаткам.

Django має доступ до всіх стандартних бібліотек Python, а також має власний вбудований функціонал фреймворку .[8]

ОЗДІЛ 2

ПРЄКТУВАННЯ КЛІЄНТ-СЕРВЕРНОГО ДОДАТКУ

2.1 Розробка технічного завдання

Розробка технічного завдання (ТЗ) є одним з найважливіших етапів при створенні продукту. Документ ТЗ є інструкцією для розробників, конструкторів та інших безпосередніх творців кінцевого продукту, який визначає чіткі вимоги до кожної деталі, що робить співробітництво замовника та виконавця безпечним та комфортним.

Технічне завдання – це документ або декілька документів, які визначають ціль, структуру, властивості та методи проєкту і виключають двозначне тлумачення різними виконавцями.

Технічне завдання є документом, який необхідний обом сторонам, як замовнику так і виконавцю. Замовник у процесі створення ТЗ зможе більш точно визначити цілі та завдання проєкту, детально продумати стратегії, уникнути помилок допущених раніше, а також визначитися з очікуваними результатами від проєкту. За допомогою ТЗ всі ідеї, думки та вимоги замовника будуть зібрані та впорядковані у одному документі, що зменшить імовірність упущення важливої інформації. Перш ніж починати виконувати об’ємну, складну у структурному плані роботу її слід обдумати та спланувати і тільки після цього приступати до її виконання.

Виконавцю проєкту ТЗ буде слугувати вихідною інформацією для виконання роботи. Тільки після ознайомлення з проєктом з використанням ТЗ можна визначити реальний обсяг роботи, терміни її виконання та вартість. Слід зазначити, що ТЗ слугує опорним

матеріалом у процесі підготовки та виконання роботи. Досить суттєвим є те, що якість виконання завдання напряму залежить від повноти інформації наданої замовником.

З метою опрацювання всіх етапів розробки веб-додатку з ведення та систематизацію бізнесу створимо технічне завдання.

Технічне завдання

1. Інформація про замовника.

   ФОП «Bookinist». Підприємство займається реалізацією літератури різного спрямування по всій Україні. ФОП «Bookinist» засноване у 2021 році. Конкуренти: «Гуверненр», «Глобус», «Книжковий ряд».

2. Цільова аудиторія.

   Цільовою аудиторією веб-додатку є люди віком від 18 до 50 років. Середній дохід від 20 до 50 тисяч гривень на місяць. Мають власне житло. Мають малий бізнес. Сімейний стан: одружений/у шлюбі. Мають середню, вищу освіту. Схильні до саморозвитку. Проживають на території України.

3. Цілі веб-додатку:

1. реалізація чіткого документування замовлень клієнтів;
2. автоматичне ведення статистичної інформації;
3. відображення асортименту товарів з додатковою інформацією;
4. реалізація можливості створення замовлення;
5. надання можливості додавання нових позицій у асортимент;
6. зворотній зв’язок;
7. Попередня структура сайту.

   Головна сторінка сайту:

8. горизонтальна навігаційна панель (логотип, аудиторія (інформація з’являється при наведенні), можливості (інформація з’являється при наведенні), ціни, про нас,);

9. кнопка входу (розташована з права на навігаційній панелі)
10. блоки з інформацією про сервіс;
11. футер (контактні номери та логотипи соціальних мереж у яких можна з нами зв’язатися).

Сторінка «Ціни».

Сторінка «Про нас»

Сторінка входу:

12. форма для входу (поле email, поле пароль, кнопка «забули пароль», кнопка «вхід»);
13. посилання на головну сторінку та на сторінку реєстрації.

Робоча сторінка:

14. ліва бокова панель ;
15. хедер ;
16. основна частина сторінки з відображенням зроблених замовлень відсортованих згідно обраного фільтру;
17. футер (контактні номери та логотипи соціальних мереж у яких можна з нами зв’язатися).

Сторінка «Статистика» (відображення статистики стосовно доходів, якості роботи персоналу, кількості замовлень(опрацьованих, на опрацюванні, загальна кількість замовлень за весь період, кількість замовлень за обраний термін).

18. Орієнтована структура веб-сайту
    1. Головна сторінка сайту
    2. Робоча сторінка:
        1. блок виводу прийнятих за день замовлень;
        2. бічна навігаційна панель(не опрацьовані замовлення, не оплачені замовлення, оплачені замовлення, архів);
        3. форма оплати за рекламу;

4. блок виводу прийнятих за день замовлень;
5. блок виводу прийнятих за день замовлень;
6. форма пошуку по прізвищу;

3. сторінка «Про нас»
4. сторінка «Статистика»
    1. відображення статистики стосовно доходів;
    2. відображення статистики стосовно якості роботи персоналу;
    3. відображення статистики замовлень (опрацьованих, на опрацюванні, загальна кількість замовлень за весь період, кількість замовлень за обраний термін)
5. сторінка виводу не оплачених замовлень,
6. сторінка виводу не опрацьованих замовлень,
7. сторінка виводу оплачених замовлень,
8. архів.
9. сторінка «Вхід»
10. сторінка «Реєстрація»

19. Вимоги до оформлення веб-додатку.

    Сайт повинен бути оформлений у офіційному стилі. Корпоративними тонами мають бути сірий, білий, бежевий. Білий колір має використовуватися як основний фоновий окрім оформлення навігаційних панелей та кнопок, які мають бути у сірому та бежевому тонах відповідно. Основні шрифти: Intro Condensed, Bravo, можливе використання схожих шрифтів.

20. Пристрої на які повинен бути адаптований сайт.
    1. монітори ПК від19 до 27 дюймів;
    2. планшети від 7 до 12 дюймів.

3. смартфони від 3,5 до 6 дюймів;
4. ноутбуки від 15,6 до 17,3 дюймів;

21. Потрібна мобільна версія.

На даному етапі розвитку справи обмежимося створенням веб-сайту.

22. Орієнтований набір модулів.

1. фільтр замовлень по критерію – сплачено/ні, виконано замовлення/ні, надіслане замовлення/ні ;
2. форма пошуку замовлення за прізвищем, номером телефону, email-ом;
3. форма оплати реклами;
4. кнопка для додавання замовлення;
5. сторінка з відображенням архіву замовлень;
6. сторінка «Статистика» повинна містити наступні модулі: модуль відображення статистики стосовно доходів, модуль відображення статистики стосовно якості роботи персоналу, модуль відображення статистики замовлень (опрацьованих, на опрацюванні, загальна кількість замовлень за весь період, кількість замовлень за обраний термін)
7. Можливість адміністрування.
8. створення, редагування, видалення замовлень;
9. можливість для адміністратора впливати на статистичні дані;
10. можливість видалення користувачів.

Якість виконання роботи може значно падати при недотриманні поетапності створення проєкту. Тому розробка ТЗ є дуже важливим етапом розробки, відсутність зазначеного документу може спричиняти

невизначеність у термінах виконання, появу правок у ході роботи, збільшення тривалості обговорення деталей проєкту.

Можна зробити висновок, що якісно зроблене технічне завдання суттєво збільшує ймовірність успішної реалізації проєкту.

2.2 Розробка макету сайту

Створення сайту неможливе без ретельного планування зв’язків та архітектури. Для цього скористаємося одним з методів структуризації концепцій з використання графічного запису mind map.

MindMap – інтелект-картка, яка допомагає у систематизації методом схем будь-якого поняття, або процесу. Представлений метод структурування даних може використовуватися для групового або самостійного аналізу питань стосовно структури сайту, систематизації візуальної інформації, створення плану роботи та ін.

Створення інтелектуальної карти у процесі проєктування сайту допоможе у визначенні з кількістю сторінок, для яких необхідно створити макет, та дозволить детально усвідомити концепцію проєкту вцілому.

Створення MindMap було проведено у онлайн-сервісі Cogle. Вибір сервісу зумовлений наступними перевагами:

11. робота сервісу у online-режимі;
12. можливість створення карти за допомогою будь-якого пристрою;
13. збереження MindMap у зручному для користувача форматі;
14. великий вибір інструментів;
15. наявність безкоштовної версії;
16. має доступ до Google Drive.
17. можливість надання доступу до карти іншим користувачам;

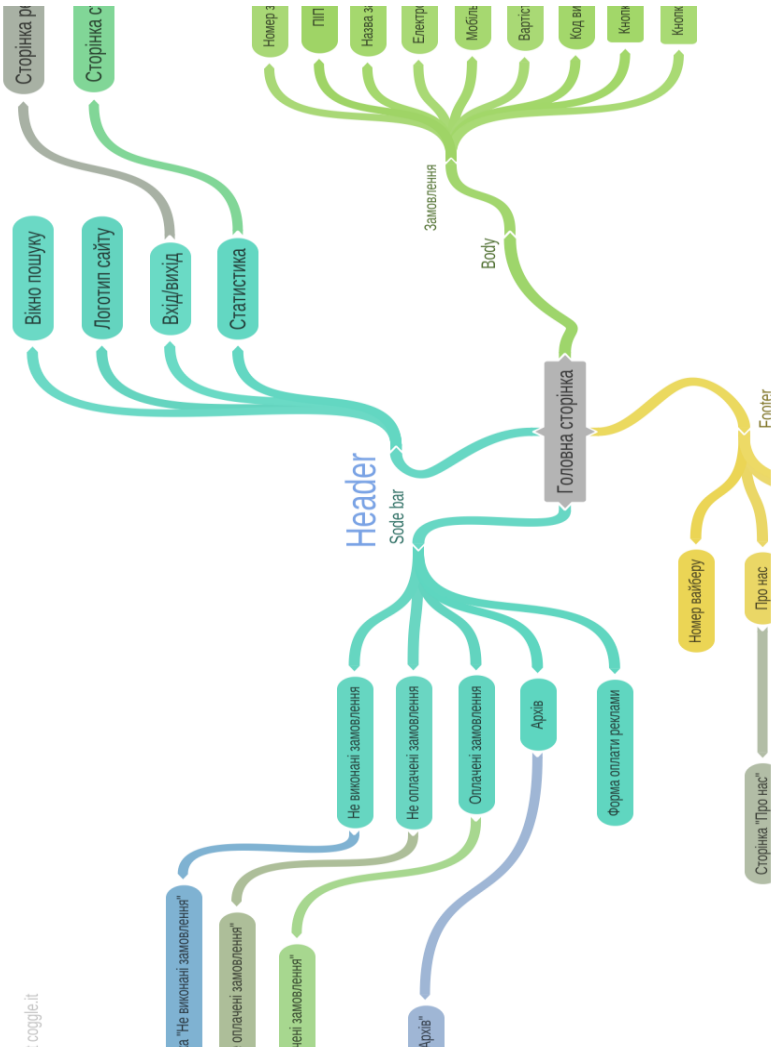

Рисунок 2.1 - Mind map сайту

Після наочної структуризації проєкту можна переходити до етапу створення прототипів основних частин веб-додатку.

Прототип – це схема усіх або декількох сторінок сайту у вигляді замальовок, ескізів, або html-документів, де відображена взаємодія та структурні елементи майбутнього сайту: меню, кнопки, форми та інше.

Наочність відображення прототипів дозволяє уникнути значних витрат часу, бо редагування загального вигляду об’єктів проводиться на етапі прототипування.

Якість прототипів може бути різною, у залежності від загального вигляду, рівня візуалізації, інтерактивності. Від звичайного ескізу на папері до інтерактивного, структурованого, багатосторінкового

прототипу. Всі моделювання веб-сайт слугують синхронізатором між уявленнями замовника та виконавця про кінцевий вигляд продукту.

Створення прототипів веб-сайту допомагає у вирішенні наступних задач:

1. створення візуального відображення ідей та уявлень про зовнішній вигляд веб-сайту на ранніх термінах його розробки;
2. можливість редагування сайту на ранніх етапах його створення;
3. оцінювання термінів та затрат на роботу;
4. ознайомлення з перспективою та можливістю подальшої модернізації.

Фінансова складова – це важливий аргумент у процесі ухвалення рішення про майбутній сайт. Завдяки прототипу можна вибрати дійсно необхідний набір функцій. Це вплине на терміни виконання завдання та дозволить чітко визначити бюджет на створення веб-додатку.

Для команди розробників прототипування вирішує питання з валідації ідей, тестування варіантів розміщення блоків, розробка use cases.

Для створення прототипів будемо використовувати онлайн-сервіс Figma.

Figma – онлайн-сервіс для розробки інтерфейсів та прототипування з можливістю організації спільної роботи у режимі реального часу.

У ході проведення практичної частини роботи було створено прототипи основних складових частин сторінок та робочої сторінки вцілому.

Одним з важливих елементів сайту є блок для виводу прийнятих замовлень.

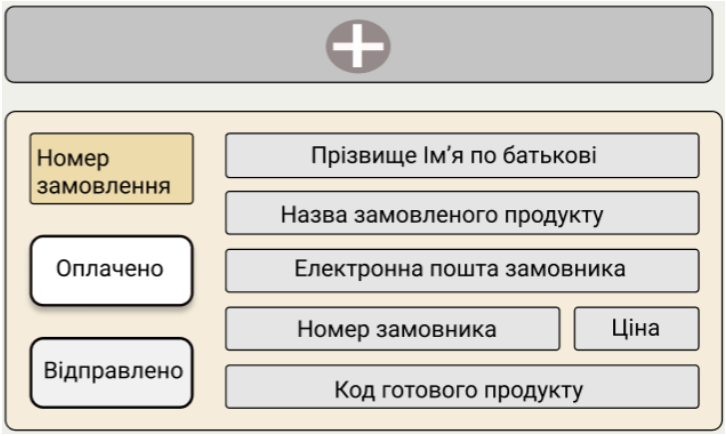

Рисунок 2.2 - Блок відображення та додавання замовлення

Блок відображення отриманих замовлень спроектований у відповідності до вимог зазначених у технічному завданні. На основі представленого блоку було розроблено макет форми додавання нових замовлень, який буде відображатися на окремій сторінці. Поля форми розташовані у відповідності до ергономічних вимог, що дозволяє поєднання максимальної інформативності з правильним розміщенням елементів блоку. Прототип у подальшому буде використаний для проведення програмування «робочої» сторінки сайту.

Рисунок 2.3 - Header сайту

Прототип хедера веб-сайту зображено на рисунку 2.3. У лівому куті буде розміщений логотип сайту та пошукове поле. Кнопки статистики та входу розміщені у правому куті блоку.

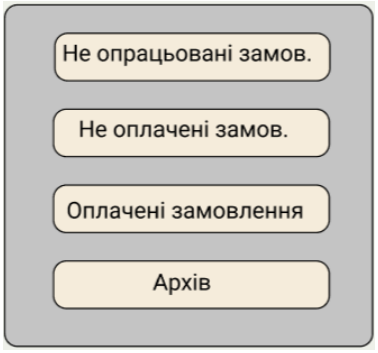

Рисунок 2.4 - Side bar

З огляду на побажання замовника було розроблено бокову панель «робочої» сторінки сайту. Side bar створений для швидкої зміни відображених замовлень згідно фільтрів, які були оговорені з замовником.

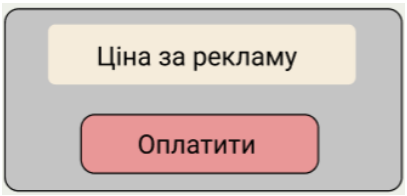

Рисунок 2.5 - Форма оплати реклами

У сучасних реаліях реклама відіграє чи не найважливішу роль у рівні прибутковості компанії. У технічному завданні зазначено про необхідність створення форми за допомогою якої буде збиратися інформація про грошові витратами на розміщення реклами. Інформація такого роду необхідна для уточнення статистичних даних стосовно прибутків та витрат компанії.

Прототип головної (робочої) сторінки (рисунок 2.6) було створено з використанням усіх вище розглянутих прототипів. Оформлення було зроблено у мінімалістичному стилі, що відповідає побажанню клієнта. Кольорова гама сторінки обрана для легкості сприйняття інформації при довгостроковій роботі (користування сайтом).

У розділі зображено та описано основні прототипи майбутнього сайту, які спроектовані у відповідності до побажань замовника. Даний етап проектування є досить важливим для обох сторін (замовник, виконавець), бо надає можливість скоректувати вартість та час розробки, що є досить важливо для замовника. Також після прототипування зменшується вірогідність внесення змін у готовий продукт.

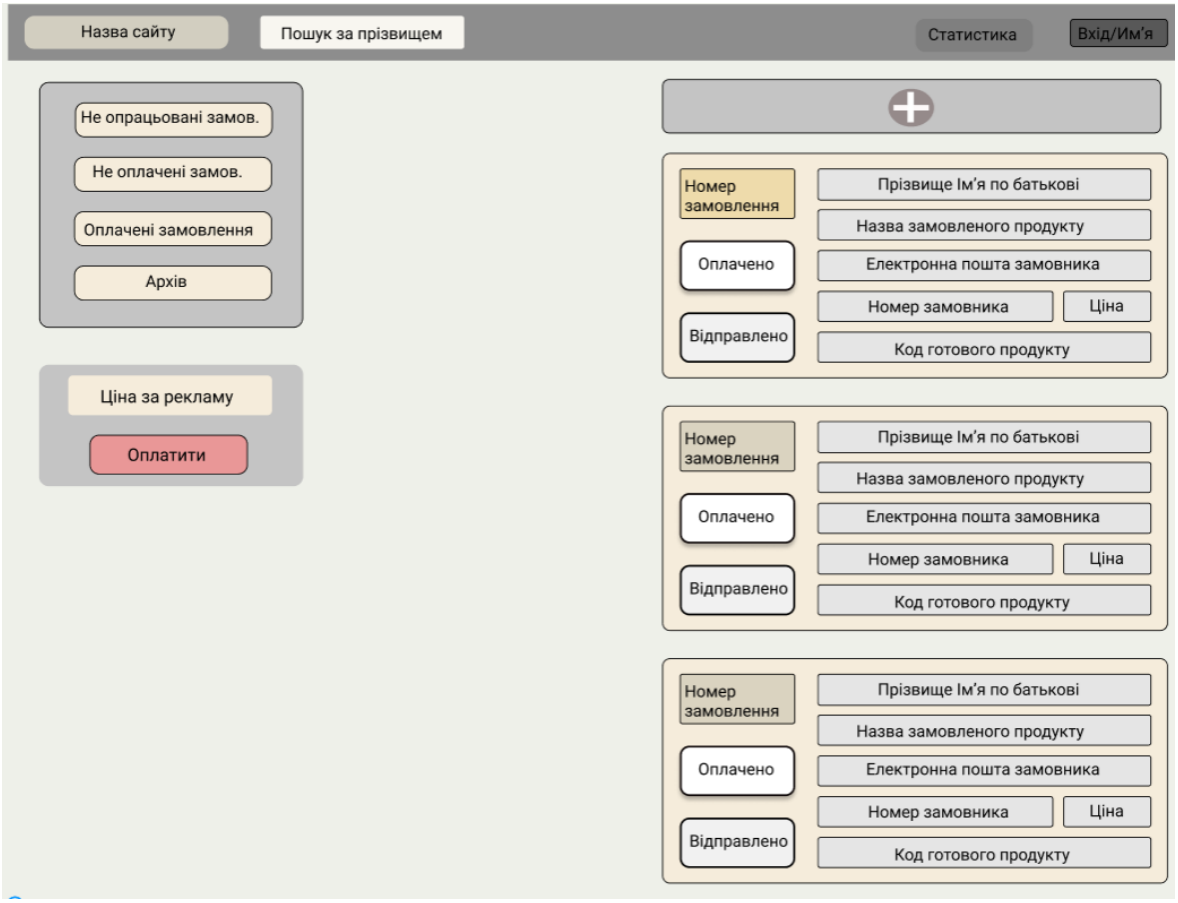

Рисунок 2.6 - Прототип головної сторінки

РОЗДІЛ 3

ПРОГРАМУВАННЯ КЛІЄНТ-СЕРВЕРНОГО ДОДАТКУ

Програмування веб додатку «Ведення та систематизація бізнесу» будемо проводити з використанням Python Django. Для спрощення роботи будемо використовувати середовище розробки Atom.

Для нашого проєкту було створене віртуальне середовище myvenv, яке зберігається у папці де буде зберігатися програмний код веб-додатку. Myvenv забезпечить ізоляцію залежності Python Django від інших проєктів (зміни одного сайту не будуть впливати на інші проєкти).

За допомогою команди activate, яку ми використовуємо у командному рядку, підключимо віртуальне оточення.

```
C:\Users\Новиков Олег\Desktop\admin_site>py -m venv myvenv

C:\Users\Новиков Олег\Desktop\admin_site>.\myvenv\Scripts\activate

(myvenv) C:\Users\Новиков Олег\Desktop\admin_site>
```

Рисунок 3.1 - Успішний перехід у віртуальне середовище

Відображення у дужках віртуального середовища говорить про його коректне підключення. Фізично віртуальне середовище є папкою, в якій містяться інші файли.

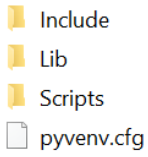

Рисунок 3.2 - Вміст папки віртуального середовища

Надалі створимо наш проєкт

```
(myvenv) C:\Users\Новиков Олег\Desktop\admin_site>django-admin startproject journal

(myvenv) C:\Users\Новиков Олег\Desktop\admin_site>
```

Рисунок 3.3 - Створення проєкту

Прєкт має назву jornal.

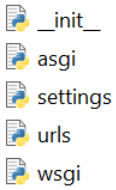

Рисунок 3.4 - Вміст папки jornal

Налаштування проєкту знаходиться у файлі settings. У цьому файлі знаходиться багато налаштувань, але змінити буде потрібно лише декілька. Рядок TIME_ZONE відповідає за годинниковий пояс який слід змінити на UTC+2. Для того щоб поставити українську мову потрібно у рядку LANGUAGE_CODE прописати ukranian, після цього всі повідомлення будуть виводитися на українській мові.

Інформація на рисунку 3.5 стосується настройки бази даних. За замовчуванням стоїть sqlite3

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

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

Рисунок 3.5 - Настройка бази даних

Щоб перевірити правильність роботи Django у командному рядку напишемо команду python manage.py runserver.

```
Run 'python manage.py migrate' to apply them.
November 12, 2021 - 01:27:00
Django version 3.2.9, using settings 'journal.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Рисунок 3.6 - Запуск сервера

Командний рядок дає відповід, що помилок немає. На сервер можна перейти за адресою http://127.0.0.1:8000.

Створимо додаток під назвою «journal» за допомогою команди python manage.py startapp journal.

Поряд із папкою проєкту з’явиться папка з назвою додатку.

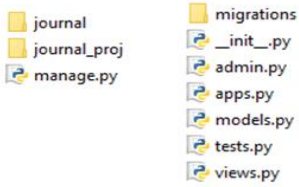

Рисунок 3.7 - Створення додатку та його вміст

Models.py - файл потрібен для створення моделей даних.

Admin.py – для присвоювання параметрів адміністрування

Views.py – цей файл викликається на запит url-адреси.

Створимо базу даних для проєкту. Спочатку слід створити суперкористувача, якого назвемо admin1 та задамо такий же пароль.

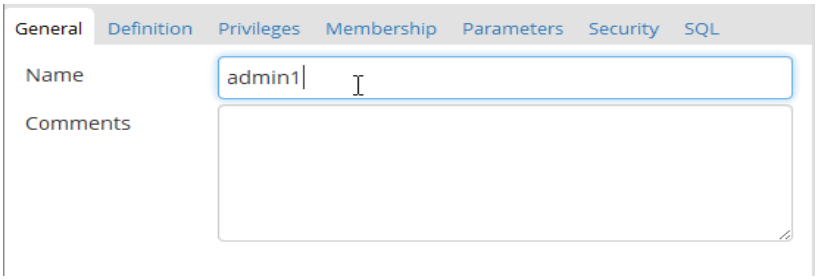

Рисунок 3.8 - Створення суперкористувача

Налаштуємо права суперкористувача.

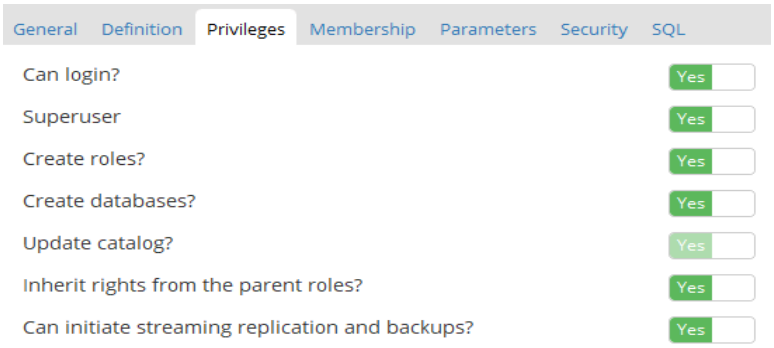

Рисунок 3.9 - Настроювання прав

Створимо базу даних, яку назвемо jour_db та у якості власника зазначимо створеного нами суперкористувача admin1.

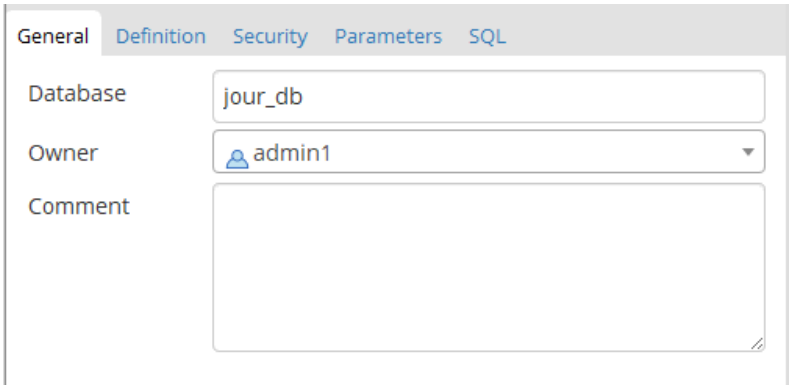

Рисунок 3.10 - Створення бази даних

У настройки проєкту додамо створену нами базу даних та вкажимо данні для суперкористувача.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'jour_db',
        'USER': 'admin1',
        'PASSWORD': 'admin1',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}
```

Рисунок 3.11 - Додавання бази даних у настройки проєкту

Настройка urls.py.

```
from django.contrib import admin
from django.urls import path

urlpatterns = [
    path('admin/', admin.site.urls),
]
```

Рисунок 3.12 - Файл urls.py

urls.py – це механізм доступу до представлень. У списку urlpatterns містяться шаблони адрес. Оскільки запити йдуть через urls.py файл проєкту, нам потрібно зробити так, щоб він посилався на urls файл нашого додатку. Для цього імпортуємо функцію include і запишемо у список необхідний код.

```
from django.contrib import admin
from django.urls import path
from django.conf.urls import include

urlpatterns = [
    path('admin/', admin.site.urls),
    path(r'', include('journal.urls')),

]
```

Рисунок 3.13 - Додавання urls створеного додатку

Django поєднують URL–адреси та представлення. URL–адреси записуються за допомогою регулярних виразів, які виділяються лапками. Якщо у них нічого немає, то мається на увазі адреса http://127.0.0.1:8000.

Перейдемо у папку нашого додатку та створимо у ньому аналогічні файли з адресами.

Початкова адреса http://127.0.0.1:8000 буде відповідати функція представлення reg – вікно з реєстрацією користувача.

Адреса http://127.0.0.1:8000/auth - функція authoriz – вікно з авторизацією.

Адреса http://127.0.0.1:8000/main - функція main – головне меню.

Адреса http://127.0.0.1:8000/write - функція writing – форма заповнення замовлення.

Адреса http://127.0.0.1:8000/jour - функція read_jour – вікно з отриманими замовленнями

Всі ці функції створимо пізніше.

```
from django.urls import path
from . import views

urlpatterns = [
    path(r'', views.reg),
    path(r'auth/', views.authoriz),
    path(r'reg/', views.reg),
    path(r'main/', views.main),
    path(r'write/', views.writing),
    path(r'jour/', views.read_jour),
]
```

Рисунок 3.14 - Створення файлу з адресами для додатку

Створення моделей (таблиці у базі даних). Для цього потрібно відкрити файл models.py і додати туди наступний текст.

```
from django.db import models
import datetime
from django.utils import timezone

class users(models.Model):
    login=models.CharField(max_length=200)
    password=models.CharField(max_length=200)


class trade(models.Model):
    login=models.ForeignKey(users, on_delete=models.CASCADE)
    secur_name=models.CharField(max_length=200)
    price_open=models.DecimalField(max_digits=8, decimal_places=2)
    quantity = models.IntegerField(default=0)
    date = models.DateTimeField('date open')


class trade_close(models.Model):
    open_trade=models.ForeignKey(trade, on_delete=models.CASCADE)
    price_close=models.DecimalField(max_digits=8, decimal_places=2)
    quantity = models.IntegerField(default=0)
    date = models.DateTimeField('date closed')
```

Рисунок 3.15 - Моделі

Таким чином, у нас буде 3 таблиці у базі даних.

Перша буде називатися users – список користувачів, яка буде містити логіни і паролі. Тип CharField – текстове значення. Максимальна довжина 200 символів.

Друга модель - trade – список відкритих угод. Вона містить номер авторизованого користувача, який є зовнішнім ключем. При видаленні користувача будуть видалені всі його угоди.

Третя модель – trade_close – список закритих угод. Містить у собі номер угоди, як зовнішній ключ, ціну, дату і час.

Далі за допомогою команди Python manage.py makemigrations journal фіксуємо список змін, а далі за допомогою команди migrate записуємо таблиці у базу даних.

Створення представлень і шаблонів. Перед створення представлень, створимо файли шаблонів – клієнтську частину (з використанням html)

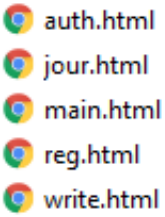

Рисунок 3.16 - Шаблони html

У нас буде п’ять функцій представлень – стільки ж, скільки і шаблонів, які ми вказали у файлі urls.py.

У першу чергу імпортуємо наші моделі з файлу models.py

```
File  Edit  Format  Run  Options  Window  Help
from django.shortcuts import render
from .models import users, trade, trade_close
```

Рисунок 3.16 - Імпорт моделей

Створимо функцію reg, яка буде відповідати за реєстрацію нових користувачів. У якості аргументу вона буде приймати POST-запит, який буде виходити з клієнтської частини. Потім ми присвоюємо змінним значення, які ми отримуємо із запиту методом get().

```
def reg(request):
    log=request.POST.get('login')
    pas=request.POST.get('password')
    print(log,'      ', pas)
    f = users.objects.all()

    if log != None:
        if len(str(log)) >= 5 and len(str(pas))>=5:
            uniq=True
            kolvo=f.count()
            for i in range(0,kolvo):
                if str(log)==str(f[i].login):
                    uniq=False
            if uniq == False:
                s='Данный логин уже используется '
            else:
                s='Успешная регистрация'
                new_user=users(login=log, password = pas)
                new_user.save()
        else:
            s='Длина логина и пароля должна быть больше 6 символов'
    else:
        s= ''
#   print('f =  ', f.password)
#   print(f[1].login)
    return render(request, 'reg.html',{'success':s})
```

Рисунок 3.17 - Функція представлення reg

Повідомлення про успішну реєстрацію користувача буде передаватися у змінній s, яка у шаблоні буде мати ім’я success. Та зазначимо, що ця функція представлення зв’язана з шаблоном reg.html.

Відкриємо reg.html, де створимо 2 форми для вводу (login, password), а також кнопку для відправки даних на сервер.

```
<h2>Регистрация</h2>

<form method="POST"> {% csrf_token %}
{{success}}

<br /><br />
 <label>
   Логин:<br />
   <input name="login"
     value="введите логин" />
 </label><br /><br />

 <label>
   Пароль:<br />
   <input name="password"
     value="введите пароль" />
 </label><br /><br />

 <input type="submit" value="Зарегистрироваться" />
  </form>

 <form action="http://127.0.0.1:8000/auth" method="GET">
  <input type="submit" value="К окну авторизации" />

  </form>
```

Рисунок 3.18 - Шаблон для функції reg

Зробимо функцію представлення для авторизації.

У змінні log і pas ми записуємо значення полів для вводу логіна і пароля. Створемо булеву змінну auth_f, у якій буде зберігатися стан авторизації. Далі ми перевіряємо, чи є запит порожнім і діємо залежно від результату.

```
def authoriz(request):
    log=request.POST.get('login')
    pas=request.POST.get('password')
    print(log,'      ', pas)
    auth_f=False
    if log != None:
        try:
            global user
            user = users.objects.get(login = log, password = pas)
            sl = 'Успешная авторизация. Пройдите в главное меню'
            auth_f = True
        except:
            sl='Неверный логин или пароль'
    else:
        sl=''

    return render(request, 'auth.html', {'auth_f':auth_f, 'succ_auth': sl})
```

Рисунок 3.19 - Функція представлення для авторизації

```
<form action="http://127.0.0.1:8000/reg" method="GET">
   <input type="submit" value="Зарегистрироваться" />
   </form>

<form action="http://127.0.0.1:8000/main" method="GET">
 {% if auth_f == True %}
   <input type="submit" value="Главное меню" />
    {% endif %}
```

Рисунок 3.19 - Шаблон для авторизації

Зробимо функцію головного меню, яку назвемо main.

Складатися буде з однієї строки, де зазначимо, що вона відповідає шаблону main.html

```
def main (request):
    return render(request, 'main.html')
```

Рисунок 3.20 - Функція для головного меню

```
<form action="http://127.0.0.1:8000/write"
 method="GET">

 <input type="submit" value="Заполнить журнал" />

  </form>

   <form action="http://127.0.0.1:8000/jour"
 method="GET">

 <input type="submit" value="Посмотреть журнал" /

  </form>

   <form action="http://127.0.0.1:8000/auth"
 method="GET">

 <input type="submit" value="Выйти" />
```

Рисунок 3.21 - Шаблон головного меню

Створимо функцію заповнення журналу.

```
def writing(request):
    if request.POST.get('price_open') != None:
        s_name=request.POST.get('secur_name')
        price_op=request.POST.get('price_open')
        quan_op=request.POST.get('quantity_open')
     #   date_op=(request.POST.get('calendar_open')).strftime('%Y-%m-%d')+(reques
        date_op=(request.POST.get('calendar_open'))+' '
                                  +(request.POST.get('calendar_time_open'))
     #   print(price_op, ' quan  ',quan_op,' dat   ',date_op)
        trade_op=user.trade_set.create(secur_name = s_name, price_open=price_op,
                                       quantity = quan_op, date= date_op)
        trade_op.save()
```

Рисунок 3.22 - Функція для заповнення журналу

```
<h2>Форма для заполнения журнала</h2>
<form method="POST"> {% csrf_token %}
<h3>Открыть сделку</h3>

<label>
    Название акции:<br />
    <input name="secur_name"
      value="" />
  </label><br />
  <label>
    Цена акции на момент покупки:<br />
    <input name="price_open"
      value="" />
  </label><br />
   <label>
    Количество акций:<br />
    <input name="quantity_open"
      value="" />
  </label><br />
    <label>
    Выберите дату и время:<br />
    <input type="date" name="calendar_open"
      value="" max="today" />

     <input type="time" name="calendar_time_open"
      value="" max="today" />
  </label><br />

  <input type="submit" value="Отправить" />
   </form>
```

Рисунок 3.23 - Шаблон заповнення журналу

Створимо останню функцію представлення, яка дозволить нам прочитати журнал угод. Зв'яжемо її з шаблоном jour.html, куди будемо відправляти tr_op, якій ми присвоємо всі угоди

```
def read_jour(request):
    tr_op=user.trade_set.all()
    return render(request, 'jour.html', {'trade_op': tr_op})
```

Рисунок 3.24 - Функція для читання журналу

```
<table border="1" width="12%" cellpadding="2">
 <tr>
        <th>Номер сделки</th>
        <th>Название акции</th>
        <th>Цена открытия</th>
        <th>Количество</th>
        <th>Дата и время открытия</th>
        <th>Цена закрытия</th>
        <th>Количество</th>
        <th>Дата и время закрытия</th>
</tr>
```

Рисунок 3.25 - Шаблон для читання журналу

За допомогою циклу зробимо перебір угод.

```
{% for i in trade_op %}
            <tr class='success1'>
                <td>{{ i.id }}</td>
                <td>{{ i.secur_name }}</td>
                <td>{{ i.price_open }}</td>
                <td>{{ i.quantity }}</td>
                <td>{{ i.date }}</td>
                {% if i.trade_close_set.count > 0 %}
                <td>{{ i.trade_close_set.all.0.price_close}}</td>
                <td>{{ i.trade_close_set.all.0.quantity }}</td>
                <td>{{ i.trade_close_set.all.0.date }}</td>
                {% else %}
                <td></td>
                <td></td>
                <td></td>
                {% endif %}
                </tr>
```

Рисунок 3.26 – Шаблон читання журналу

.

ВИСНОВКИ

В ході виконаної кваліфікаційної роботи була досягнуто поставленої мети, а саме, проєктування та розробка клієнт-серверного додатку для адміністрування та ведення бізнесу на Python Django.

Головна перевага Python – простота. Більшою мірою до спрощення написання програм наводить відсутність необхідності оголошувати змінні, тип яких визначається по ходу написання коду автоматично, а також зручна система відступів, що надає коду компактності та структурованості. Наявність великої кількості бібліотек та інструментів дозволяє вирішувати широкий спектр завдань різних галузей.

Використання Python Django дозволяє у стислі терміни розробляти якісні та багатофункціональні web-додатки, які допоможуть виконувати широкий спектр завдань, від ведення новинного сайту до адміністрування бізнесу.

Підсумовуючи практичну частину даної роботи можна зробити висновок, що зпроєктований додаток надасть можливість користувачам адмініструвати справу з виготовлення та продажу товарів дистанційно. Web-додаток скоротить час власників справи автоматизувавши ведення статистики, сформувавши списки клієнтів відповідно до актуальних для адміністратора критеріїв, буде вести архів замовлень та інше.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Коротка історія мов програмування. Транслятори. URL: http://younglinux.info/python/programminglanguage.php (дата звернення: 23.10.2021).

2. Wikipedia, Python. URL: https://ru.wikipedia.org/wiki/Python (дата звернення: 27.10.2021).

3. A Brief Timeline of Python. URL:http://python–history.blogspot.ru/2009/01/brief–timeline–of–python.html (дата звернення: 28.09.2021).

4. Wikipedia, История языка программирования Python. URL:https://ru.wikipedia.org/wiki/История_языка_программирования_Python (дата звернення: 02.11.2021).

5. Короткий огляд мови Python. URL:http://www.helloworld.ru/texts/comp/lang/python/python2/index.htm (дата звернення: 04.11.2021).

6. Форс'є Дж., Django. Разрабка веб–додатків на Python, 2009. –456 с.

7. Історія Django. URL: https://djbook.ru/ch01s03.html (дата звернення: 05.11.2021).

8. Python Django. URL: http://ep–z.ru/stroitelstvo/sayt/python/python–django (дата звернення: 07.11.2021).

9. Вигерс К. И. Разработка требований к программному обеспечению. СПб.: БХВ-Петербург, 2018. 736 с.

10. Роббинс Дж. HTML5. Карманный справочник. М.: Вильямс, 2016. 192 с.

11. Маттес Е. Пришвидшений курс Python. Практичний, проєктно-орієнтований вступ до програмування. Л.: Видавництво старого лева,2021.600с.

12. Тарасов С. В. СУБД для программиста. Базы данных изнутри. Омск: Соломон, 2015. 320 с.

13. Макдональд М. Веб-разработка. Исчерпывающее руководство. СПб: Питер, 2017. 640 с.

14. Мейер Э. А. CSS. Карманный справочник. М.: Вильямс, 2017. 288 с.

15. Казаков Г. С. Система автоматизации проектирования баз данных на основе модели «сущность-связь». // Информационные системы и измерительно-вычислительные комплексы: сборник докладов студентов и аспирантов кафедры «Измерительно вычислительные комплексы» на научно-технических конференциях / Под ред. В. В. Родионова, 2010. С. 25–31.

16. Рыжко А. Л., Рыбников Н. А., Рыжко Н. А. Информационные системы управления производственной компанией. М.: Юрайт, 2016. 356 с.

17. Дэйт К. Дж. Введение в системы баз данных. М.: Вильямс, 2017. 1328 с.

18. Кузнецов С. Д. Основы баз данных. М.: Бином, 2007. 488 с.

19. Пегат А. Нечеткое моделирование и управление. М.: Бином. Лаборатория знаний, 2017. 800 с.

20. Тысленко А. Менеджмент. Организационные структуры управления. М.: Альфа-Пресс, 2011. 320 с.