

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,

Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему РОЗРОБКА ВЕБСАЙТУ ДЛЯ ВЕДЕННЯ ОБЛІКУ ПРОДУКЦІЇ
ПІДПРИЄМСТВА З ВИГОТОВЛЕННЯ КАНЦЕЛЯРСЬКИХ ВИРОБІВ

WEBSITE DEVELOPMENT FOR STOCK TRACKING

AT A STATIONERY MANUFACTURING COMPANY

Виконав: студент(ка) 4 курсу, групи КНТ-117

Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Бойко Б.С.

(прізвище та ініціали)

Керівник Субботін С.О.

(прізвище та ініціали)

Рецензент Гофман Є.О.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Бойка Богдана Сергійовича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка вебсайту для ведення обліку продукції підприємства з виготовлення канцелярських виробів. Website Development for Stock Tracking at a Stationery Manufacturing Company
- керівник проєкту (роботи) Субботін Сергій Олександрович, зав. кафедри,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
- затверджені наказом закладу вищої освіти від “30” березня 2021 року № 103
2. Строк подання студентом проєкту (роботи) 27 травня 2021 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Проєктування інформаційних моделей і розробка алгоритмів. 2. Технологія програмної реалізації розроблених моделей та алгоритмів. 3. Програмна реалізація розроблених моделей та алгоритмів.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-3 Основна частина	Субботін С.О., зав. кафедри		
Нормоконтролер	Липовець М.В., асистент		

7. Дата видачі завдання “11” березня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2-3 тижні	Розділ 1
3	Розробка та удосконалення методів, моделей й алгоритмів вирішення задачі.	4-5 тижні	Розділ 2
4	Розробка архітектури програми.	6 тиждень	Розділ 3
5	Розробка програми.	7-8 тижні	Розділ 4
6	Тестування та експериментальне дослідження програмного забезпечення.	9 тиждень	Розділ 5
7	Розробка економіко-організаційної частини.	10 тиждень	Розділ 6
8	Розробка заходів з охорони праці та безпеки в надзвичайних ситуаціях.	11 тиждень	Розділ 7
9	Оформлення пояснювальної записки та документів до неї.	12-13 тиждень	Додатки
10	Нормоконтроль та рецензування.	14-15 тижні	
11	Захист роботи.	16 тиждень	

Студент(ка)

(підпис) Бойко Б.С.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Субботін С.О.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
80 с., 24 рис., 9 табл., 3 дод., 18 джерел.

ІНФОРМАЦІЙНА СИСТЕМА, АІС, СИСТЕМА ОБЛІКУ, БАЗА
ДАНИХ, СУБД.

Об'єкт дослідження – процес розробки WEB-орієнтованих
інформаційних систем.

Предмет дослідження – програмні засоби для обліку продукції
підприємства.

Мета роботи – створення програмної інформаційно-облікової системи
для автоматизації та прискорення обліку продукції підприємства.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване
програмування, мова програмування Python, Django web framework,
персональний комп'ютер з процесором Intel Core i7 7700k під управлінням
операційної системи Microsoft Windows 10.

Результати. Створено програмний web-додаток для обліку продукції
підприємства, що виготовляє канцелярські вироби.

Висновки. Розроблено програмну інформаційно-облікову систему для
ведення обліку продукції підприємства.

Галузь використання – виробничі підприємства, системи обліку.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 80 pages, 24 figures, 9 tables, 3 appendices, 18 sources.

INFORMATION SYSTEM, AIS, ACCOUNTING SYSTEM, DATABASE, DBMS.

The object of research is the process of developing WEB-oriented information systems

The subject of research - software for accounting for enterprise products.

The purpose of the work is to create a software information and accounting system for automation and acceleration of accounting for enterprise products.

Materials, methods and technical means: object-oriented programming, Python programming language, Django web framework, personal computer with Intel Core i7 7700k processor running Microsoft Windows 10 operating system.

Results. A software web-application for accounting of products of an enterprise that manufactures stationery has been created.

Conclusions. A software information and accounting system for accounting of enterprise products has been developed.

Field of use - manufacturing companies, accounting systems.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Проєктування інформаційних моделей і розробка алгоритмів.....	9
1.1 Вибір методів і засобів проєктування.....	9
1.2 Проєктування системи обліку.....	15
1.3 Проєктування користувацького інтерфейсу.....	20
1.4 Алгоритми роботи з системою обліку	23
2 Технологія програмної реалізації розроблених моделей та алгоритмів	24
2.1 Порівняння існуючих методів програмної реалізації	24
2.2 Обраний метод програмної реалізації.....	34
3 Програмна реалізація розроблених моделей та алгоритмів	47
3.1 Реалізація проєктованих моделей.....	47
3.2 Реалізація користувацького інтерфейсу	52
3.3 Інструкція використання реалізованої системи обліку.....	57
Висновки	63
Перелік джерел посилання	64
Додаток А Текст програми.....	66
Додаток Б Текст файлу налаштувань маршрутизації.....	71
Додаток В Ілюстрація презентації.....	73

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API	– Application Programming Interface;
ER	– Entity Relationship;
JVM	– Java Virtual Machine;
MVC	– Model View Controller;
MVT	– Model View Template;
SSN	– Social Security Number;
SQL	– Structured Query Language;
VB	– Visual Basic;
AIC	– автоматизована інформаційна система;
АСУ	– автоматизована система управління;
БД	– база даних;
ІС	– інформаційна система;
ООП	– об'єктно-орієнтоване програмування;
ПЗ	– програмне забезпечення;
СУБД	– системи управління базами даних.

ВСТУП

В умовах конкуренції, що збільшується на ринку виробництва канцелярії, підприємства змушені шукати кошти підвищення ефективності бізнесу: покращувати рівень обслуговування та якість асортименту, вдаватися до застосування дисконтних схем, надавати додаткові послуги. Впровадження автоматизації обліку діяльності підприємства - одне з основних умов підвищення ефективності бізнесу.

Численні зручності та переваги, які автоматизація дає не тільки бізнесу, а й клієнтам, свідчать про те, що кошти, вкладені в розвиток підприємства, несумірні з постійними втратами в результаті крадіжки, псування товару та ряду інших проблем обліку і управління персоналом.

Реалізація проєкту автоматизованої системи «Вебсайт для ведення обліку продукції підприємства що виготовляє канцелярські вироби» значно полегшить роботу співробітників підприємства і забезпечить можливість зменшити витрати на управління за рахунок звільнення людських ресурсів, зайнятих різними видами обробки паперових документів, зберігати і аналізувати дані за будь-який проміжок часу, здійснювати пошук потрібної інформації за різними критеріями відбору.

В ІС передбачені наступні функції:

- ведення бази даних (БД);
- формування звітів;
- формування довідок.

Відповідно ІС складається з БД і наступних підсистем:

- управління БД;
- ведення БД;
- формування довідок;
- формування звітів по продажу і надходження товарів;

Визначено, що реалізація web-орієнтованого додатку буде кращим рішенням для ведення обліку продукції підприємства, що виготовляє канцелярські вироби через низьку вимогливість до апаратного забезпечення та великої кількості існуючих бібліотек для створення додатків.

1 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНИХ МОДЕЛЕЙ І РОЗРОБКА АЛГОРИТМІВ

1.1 Вибір методів і засобів проєктування

Під проєктуванням інформаційної системи[1] можна розуміти виділення окремих її модулів і визначення зв'язків між ними з метою мінімізації залежностей між частинами майбутнього проєкту і досягнення кращого уявлення в цілому. Дана операція передуює кодування - розробці коду окремих модулів, розробці документації для користувачів, яка необхідна для опису інформаційної системи. З точки зору інженерії при створенні інформаційної системи необхідно піднімати питань експлуатації системи, внесення в неї змін, а також найперші дії - аналіз потреб користувачів і вироблення рішень, винахід функцій, що задовольняють ці потреби. Без цього неможливо, з одного боку, врахувати реальну ефективність системи в вигляді відносини отриманих результатів до всіх зроблених витрат і, з іншого боку, правильно оцінювати в ході розробки ступінь відповідності системи реальних потреб користувачів і замовників.

Весь період існування програмного забезпечення, пов'язаний з підготовкою до його розробки, розробкою, використанням і переробками, починаючи з того моменту, коли приймається рішення розробити з наявних компонентів нову систему або коли приходить сама ідея про необхідність програми певного роду, до того моменту, коли повністю припиняється всяке її використання, називають життєвим циклом програмного забезпечення.

В ході життєвого циклу програмного забезпечення воно проходить через аналіз предметної області, збір вимог, проєктування, кодування, тестування, супровід і інші види діяльності. Кожен вид діяльності являє собою досить однорідний набір дій, які виконуються для вирішення однієї задачі або групи тісно пов'язаних задач в рамках розробки та підтримки експлуатації програмного забезпечення.

На різних етапах в програмне забезпечення залучаються люди, які виконують різні ролі. Кожна роль може бути охарактеризована як абстрактна група зацікавлених осіб, що беруть участь в діяльності по створенню і експлуатації системи і вирішальних одні й ті ж завдання або мають одні і ті ж інтереси по відношенню до неї. Прикладами ролей є бізнес-аналітик, інженер за вимогами, архітектор, проєктувальник призначеного для користувача інтерфейсу, програміст і т.п. Загальну структуру у життєвого циклу будь-якого програмного забезпечення, схоже, визначити неможливо, оскільки вона істотно залежить від цілей, для яких це програмне забезпечення розробляється або набувається, і від розв'язуваних їм завдань. Структура життєвого циклу буде істотно різною у програми для форматування коду, яка спочатку робилася програмістом для себе, а згодом була визнана перспективною як продукт і перероблена, і у комплексної системи автоматизації підприємства, яка з самого початку була задумана як така.

Існує набір стандартів, що визначають різні елементи в структурі життєвих циклів програмного забезпечення і включають його програмно-апаратних систем. В якості основних таких елементів виділяються технологічні процеси - структуровані набори діяльностей, які вирішують деяку загальну задачу або пов'язану сукупність завдань, такі як процес супроводу програмного забезпечення, процес забезпечення якості, процес розробки документації та ін. Процеси можуть визначати різні етапи життєвого циклу і пов'язують їх з різними видами діяльностей, артефактами і ролями зацікавлених осіб.

Основні етапи проєктування ІС представлені в табл. 1.1.

Таблиця 1.1 – Основні етапи розробки ІС

Етап	Методи рішення
Розробка концептуальної моделі ІС	Структурно-функціональне і / або об'єктно-орієнтоване моделювання
Розробка логічної моделі ІС	Інформаційне моделювання (створення діаграми сутність-зв'язок)
Розробка фізичної моделі і програмного забезпечення ІС	Реалізація об'єктів логічної моделі, розробка програмного коду
Тестування та налагодження ІС	Коригування програмного забезпечення
Експлуатація ІС	Підтримка ІС після введення в експлуатацію

Проектування БД «Облік продукції підприємства що виробляє канцелярські вироби» складається з наступних кроків.

- аналіз предметної області;
- концептуальне проектування;
- логічне проектування;
- фізичне проектування;

Аналіз предметної області включає в себе:

- ідентифікацію функціональної діяльності предметної області;
- ідентифікацію об'єктів, які здійснюють цю функціональну діяльність;
- ідентифікацію характеристик цих сутностей;
- ієрархію між об'єктами;

Концептуальне проектування - початковий етап проектування, при якому створюється чорновий варіант продукту, що показує принцип закладання логіки для початкового узгодження і перевірки обмежень засобів розробки.

В процесі концептуального проектування проектувальник з вимог до системи і аналізу дій користувачів створює:

Інформаційну архітектуру продукту (наприклад, дерево сайту або послідовність екранів в інтернет-магазині);

Розташування контенту і функціоналу на екранах умовними блоками.

Концептуальний прототип використовується для обговорення подальшого плану робіт, порядку виконання і можливостей реалізації.

Після вирішення виниклих питань з концептом продукту проєктувальник переходить до детального проєктування і створює детальний прототип.

Логічним проєктуванням баз даних формально можна вважати процес створення моделі світу самої бази даних, без тієї системи, яка дозволить їй працювати, і інших фізичних деталей. Точність і повнота грають в цьому процесі ключову роль. Одним з головних переваг цього етапу є те, що завжди можна взяти чорновий проєкт, відкласти його в сторону і почати все заново або просто внести бажані поправки. Набагато легше змінювати ті чи інші деталі на етапі проєктування, ніж мати справу з проблемами вже реалізованої виробничої бази даних, яка погано спроектована.

Логічне проєктування полягає в концептуальному моделюванні бази даних і гарантії того, що дані в таблицях проходять перевірку на цілісність і не є надлишковими. Для задоволення цих вимог реалізуються принципи нормалізації даних.

Методика, яку найчастіше застосовують для логічного представлення та аналізу компонентів бізнес-системи і моделювання відповідного рішення після завершення аналізу вимог, називається ER-моделюванням (Entity Relationship Modeling - створення моделей типу "сутність-відношення"). ER-моделі легко конструювати, а, завдяки акценту на графічному поданні, ще й дуже легко розуміти. Проте, створювати справжню СУБД на підставі однієї тільки ER-моделі підприємства не можна. Користь ER-моделювання полягає в проєктуванні, а не в реалізації баз даних. Утворювати основу для високорівневої мови маніпулювання даними на кшталт SQL ER-

моделювання не дозволяє, тому та модель, яку проєктувальники створюють за допомогою ER-моделювання, для реалізації перетворюється в реляційну модель. Тобто за рахунок перетворення абстрактної структури з сутностями і відносинами в схему реляційної бази даних реляційна модель допомагає перетворювати ER-модель в реляційну СУБД.

У ER-моделюванні всі об'єкти ділової сфери розглядаються як сутності (entities), що володіють певними атрибутами. Сутністю вважається будь-який об'єкт, який представляє інтерес для системи і існує в діловому світі.

Атрибутами сутностей називаються властивості сутностей, які представляють інтерес. Наприклад, у сутності, що представляє студента, атрибутами можуть бути ідентифікаційний номер студента (Student ID), його адреса (Address), номер телефону (Phone Number) і т.п.

Далі в ER-моделюванні описуються відносини (relationships) між усіма бізнес-сутностями. Ці відносини відображають те, яким чином різні сутності пов'язані (або співвідносяться) між собою. Наприклад, співробітник може являти собою сутність, що володіє атрибутами на зразок імені (Name) і адреси (Address), і бути пов'язаним в моделі з іншою сутністю під назвою "Відділ" (Department) на підставі того факту, що він працює в цьому відділі.

Відносини між сутностями можуть являти собою і відносини типу "один до одного", і "один до багатьох", і "багато до багатьох" або відносини ще якогось іншого типу. Найчастіше зустрічаються відносини наступних типів (за умови наявності двох сутностей, A і B).

– один до багатьох. У таких відносинах кожен екземпляр сутності A може мати відношення з декількома членами іншої сутності B. Наприклад, сутність під назвою Клієнт може брати багато книг з бібліотеки, але кожна книга за раз може видаватися тільки одного єдиного Клієнту. Відповідно виходить, що між сутностями Клієнт і Книга має існувати відношення типу "один до багатьох". Зрозуміло, таке ставлення може і не існувати при наявності Клієнта, який ще не брав ніякої Книги. Тобто фактично

відношення має говорити наступне: "один Клієнт може брати нуль, одну або більше Книг".

– один до одного. У таких відносинах тільки один екземпляр будь-якої з сутностей може мати відношення з екземпляром іншої сутності. Наприклад, у кожної людини може бути тільки один дійсний номер картки соціального страхування (Social Security Number - SSN), а кожен номер картки соціального страхування може посилатися тільки на одну людину.

– багато до багатьох. У таких відносинах кожен екземпляр сутності А може мати відношення з одним і більше екземплярами сутності В, а кожен екземпляр сутності В - з одним і більше екземплярами сутності А. Як приклад візьмемо сутність під назвою Кінозірка і сутність під назвою Кінофільм. Кожна кінозірка може зніматися в декількох кінофільмах, і в кожному кінофільмі може брати участь кілька кінозірок. У реальному житті відносини типу "багато до багатьох" зазвичай розбиваються на більш прості відносини типу "один до багатьох", які, як склалося, є найбільш поширеною формою відносин між сутностями.

Правильне визначення потужностей відносин відіграє ключову роль для створення добре спроектованої реляційної бази даних. Некоректне моделювання відносин зумовлює появу проблем на кшталт надмірності, дублювання і аномалій даних.

Мета етапу фізичного проектування – опис конкретної реалізації БД, що розміщується в зовнішній пам'яті комп'ютера. Процедури фізичного проектування:

– проектування таблиць БД засобами обраної СУБД - здійснюється вибір реляційної СУБД[2], яка буде використовуватися для створення БД, що розміщується на машинних носіях. Вивчаються її функціональні можливості з проектування таблиць. Потім виконується проектування таблиць і схеми їх зв'язку в середовищі СУБД. Підготовлений проєкт БД описується в супроводжуваній документації.

- проектування фізичної організації БД. На цьому кроці вибирається найкраща файлова організація для таблиць. Виявляються транзакції, які будуть виконуватися в проєктованій БД, і виділяються найбільш важливі з них. Аналізується пропускна здатність транзакцій-к-ть транзакцій, які можуть бути оброблені за заданий інтервал часу, і час відповіді - проміжок часу, необхідний для виконання однієї транзакції. Прагнуть до підвищення пропускної спроможності транзакцій і зменшення часу відповіді. На підставі зазначених показників приймаються рішення про оптимізацію продуктивності БД шляхом визначення індексів в таблицях, що прискорюють вибірку даних з бази, або зниження вимог до рівня нормалізації таблиць. Проводиться оцінка дискового обсягу пам'яті, необхідного для розміщення створюваної БД. Прагнуть до його мінімізації.

- розробка стратегії захисту БД. БД- цінний корпоративний ресурс, тому організації її захисту приділяється багато уваги. Для цього проєктувальники повинні мати повне і чітке уявлення про всі засоби захисту, які подаються обраної СУБД.

- організація моніторингу функціонування БД і її налаштування. Після створення фізичного проєкту БД організовується безперервне спостереження за її функціонуванням. Отримані відомості про рівень продуктивності БД використовуються для її налаштування. Для цього залучаються і кошти обраної СУБД.

1.2 Проєктування системи обліку

Концептуальне проєктування.

Основна концепція функціоналу розроблюваної інформаційної системи «Вебсайт для ведення обліку продукції підприємства що виготовляє канцелярські вироби» полягає у веденні БД обліку створюваної продукції, що можна розкласти на такі функціональні особливості:

- додавання та редагування у БД звітності щодо виробленої продукції;
- можливість завантаження звітів щодо виробленої продукції;
- додавання та редагування у БД звітності щодо продаж продукції;

Концептуальне зображення системи зображено на рис. 1.1.



Рисунок 1.1 – Концептуальна схема системи обліку продукції

Бухгалтер(користувач), що буде заповнювати звіти, зможе створювати звіти, редагувати їх, створювати у БД конкретні продукти з конкретними унікальними ідентифікаторами(серійними номерами) задля забезпечення безпомилкового обліку продукції на підприємстві, що зменшить ризик збитків через можливі кражі/недостачі продукції, адже кожен продукт буде офіційно зареєстровано у БД.

Логічне проєктування.

Виходячи з концептуального проєктування можна виділити наступні сутності:

- продукт;
- тип продукту;
- звіт виготовленої продукції(список продуктів, що вироблено за певну дату);
- звіт проданої продукції(список продуктів, що продано за певну дату);
- замовник продукції. Звіт проданої продукції повинен посилатись на конкретного замовника, що придбав товар;
- користувач;

Продукт.

Дана сутність[3] представляє собою фізичний об'єкт конкретного виготовленого на підприємстві продукту. Список атрибутів представлено у таблиці 1.2.

Таблиця 1.2 –Атрибути сутності «Продукт»

Атрибут	Пояснення
Ідентифікаційний номер	Серійний номер продукту
Тип продукту	Тип продукту, даний атрибут посилається на сутність «Тип продукту» задля полегшення створення продуктів.

Дана сутність має універсальний набір атрибутів, що можна використовувати у випадку з кожним типом продуктів.

Тип продукту.

Дана сутність представляє собою пряму типізацію продукції. (у даному випадку прикладом слугує тип – «олівці білі»). Сутність «Продукт» буде напряду посылатись на дану сутність.

Список атрибутів представлено у табл. 1.3.

Таблиця 1.3 – Атрибути сутності «Тип продукту»

Атрибут	Пояснення
Ідентифікаційний номер	Унікальний ідентифікатор
Назва	Назва типу продуктів
Характеристики	Опис характеристик даного типу продуктів
Вартість	Роздрібна вартість продажу продукту певного типу встановлена підприємством.

Звіт виготовленої продукції.

Дана сутність відображає список виготовлених продуктів, що спрощує створення звітності обліку загальної продукції.

Список атрибутів даної сутності представлено у таблиці 1.4.

Таблиця 1.4 – Атрибути сутності «Звіт виготовленої продукції»

Атрибут	Пояснення
Ідентифікаційний номер	Унікальний ідентифікатор
Дата	Дата, за якою ведеться звіт

Кожен продукт може відноситись виключно до одного звіту виготовленої продукції, отже потрібна додаткова суміжна сутність – «Звіт виготовлення-продукт», де атрибут «Звіт» буде відноситись до звіту виготовленої продукції, а атрибут «Продукт» буде унікальним та відноситись до виготовленого продукту. Дана сутність повинна автоматично створюватись при додаванні продукту до звіту.

Звіт проданої продукції.

Дана сутність відображає список проданих продуктів, що спрощує створення звітності обліку загального продажу продукції.

Список атрибутів даної сутності представлено у таблиці 1.5.

Таблиця 1.5 – Атрибути сутності «Звіт проданої продукції»

Атрибут	Пояснення
Ідентифікаційний номер	Унікальний ідентифікатор
Дата	Дата, за якою ведеться звіт
Замовник	Посилання на сутність «Замовник»
Сума прибутку	Загальна сума прибутку

Кожен продукт може відноситись виключно до одного звіту проданої продукції, отже потрібна додаткова суміжна сутність – «Звіт продажу-продукт», де атрибут «Звіт» буде відноситись до звіту проданої продукції, а атрибут «Продукт» буде унікальним та відноситись до виготовленого продукту. Дана сутність повинна автоматично створюватись при додаванні продукту до звіту.

Замовник.

Дана сутність представляє собою набір даних про замовника продукції підприємства. Набір атрибутів представлено у таблиці 1.6.

Таблиця 1.6 – Атрибуті сутності «Замовник»

Атрибут	Пояснення
Компанія	Назва компанії-замовника
Адреса	Адреса компанії
Контактні дані	Контактні дані замовника

Схема відношень між представленими сутностями зображена на рис. 1. 2.

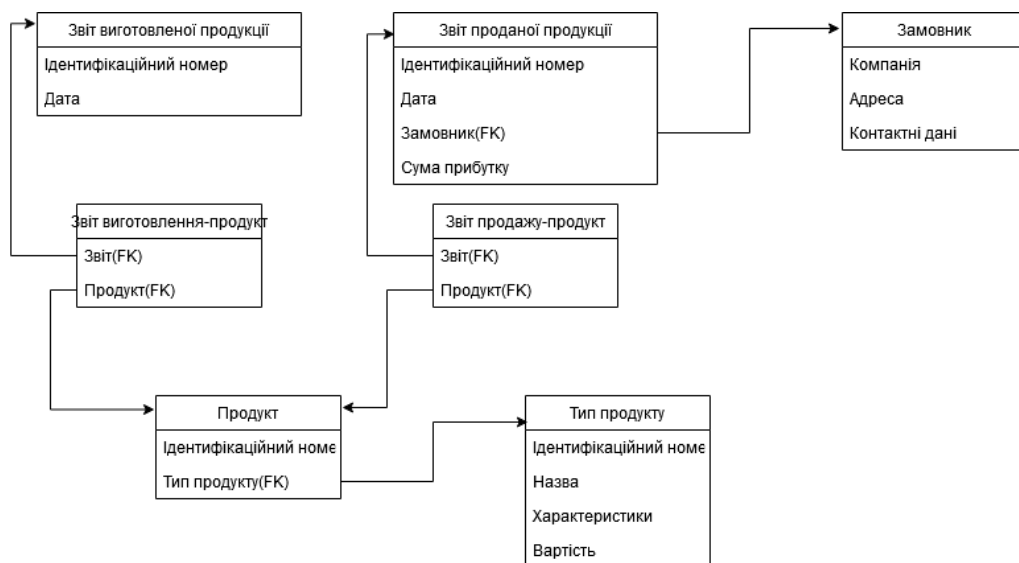


Рисунок 1.2 – Логічна схема відносин між сутностями

Користувач.

Фінальним етапом логічного проєктування стане сутність «Користувач», що представляє собою сутність користувача системи – бухгалтера. Користувач як і у будь-якій іншій системі повинен мати як мінімум два атрибути – певний ідентифікатор(логін) та пароль для забезпечення безпеки власного профілю та системи взагалі від несанкціонованого входу до системи обліку.

Набір атрибутів сутності «Користувач» представлено у таблиці 1.7.

Таблиця 1.7 – Атрибути сутності «Користувач»

Атрибут	Пояснення
Ідентифікатор	Логін для забезпечення входу в систему
Пароль	Пароль для забезпечення безпеки системи та профілю користувача

Фізичне проєктування бази даних буде розглянуто у розділі 3.1. «Реалізація проєктованих моделей».

1.3 Проєктування користувацького інтерфейсу

У розділі 1.2. встановлені функціональні вимоги до розроблюваної системи обліку[4]:

- додавання та редагування у БД звітності щодо виробленої продукції;
- можливість завантаження звітів щодо виробленої продукції;
- додавання та редагування у БД звітності щодо продаж продукції;

Опираючись на дані вимоги, визначено, що користувацький інтерфейс має бути простим та зручним для користування. У роботі системи обліку не передбачається використання зображень, отже інтерфейс прийнято розробити мінімалістичним. Загальні вимоги до інтерфейсу:

- зрозуміла та зручна навігація по всім функціональним можливостям системи обліку;
- функціональні кнопки/меню додавання/редагування обліку продуктів, звітів;
- оптимізація та мінімалізм;

Опираючись на вимоги, розроблено схематичне зображення початкової сторінки інтерфейсу системи обліку(див. рис. 1.3).

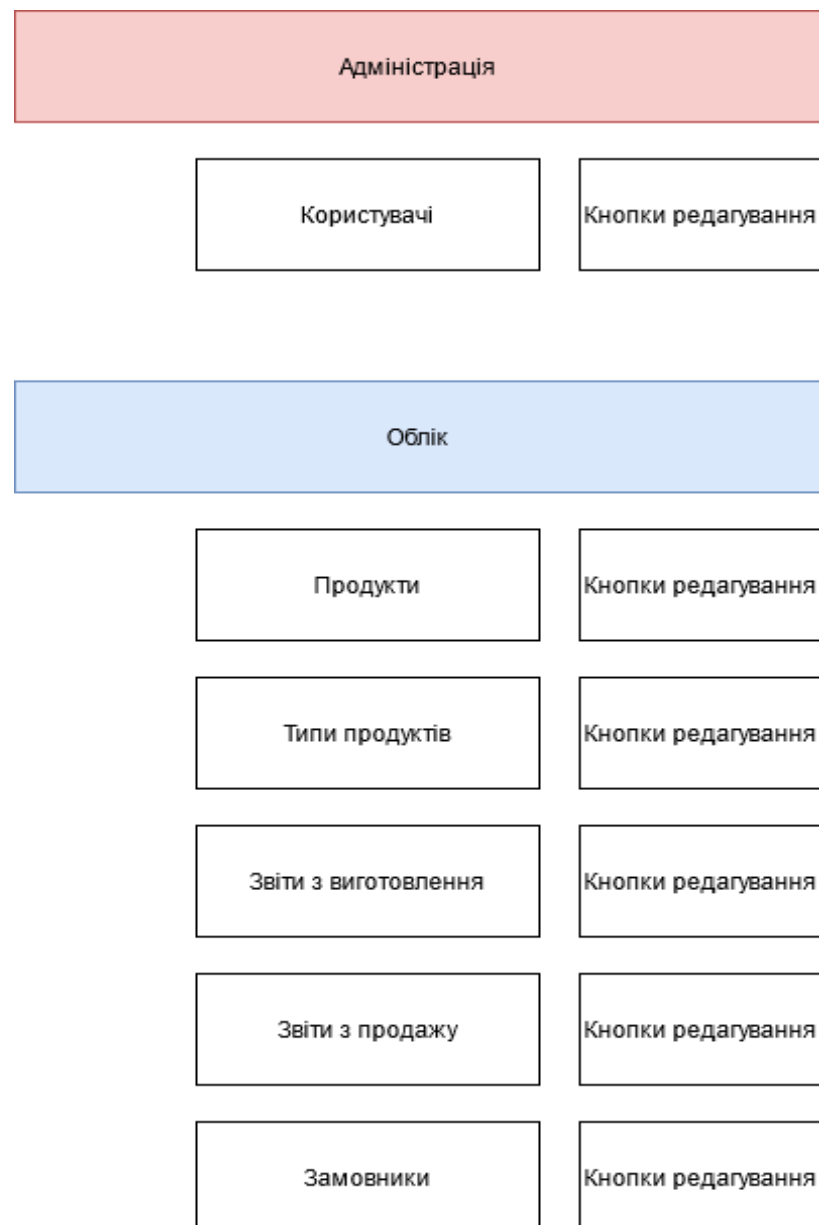


Рисунок 1.3 – Схематичне зображення інтерфейсу початкової сторінки системи обліку

Кожна сутність представлена окремою вкладкою меню. Біля кожної сутності присутні кнопки додавання нових записів або редагування існуючих. Сама сутність, як вкладка меню є посиланням на детальну сторінку керування записами даної сутності, як на рис. 1.4.

Замовники	
Фільтри/меню сортування	Додати замовника
Замовник 1	
Замовник 2	
Замовник 3	

Рисунок 1.4 – Детальна сторінка керування записами сутності «Замовники»

У детальному меню керування записами певної сутності присутні фільтри, кнопка створення нового запису. Кожен запис у даному меню являється посиланням на детальну сторінку редагування певного запису, що містить форми для заповнення, як на рис. 1.5(даний інтерфейс аналогічний і для процесу створення нового запису).

Редагування "Замовник 1"	
Замовник 1	
Компанія	Замовник 1
Адреса	-
Контактні дані	-
Видалити запис	Зберегти запис

Рисунок 1.5 – Сторінка редагування запису «Замовник 1»

Спроектований інтерфейс виконує всі вимоги перераховані вище, є простим, зручним та ефективним у рутинній роботі бухгалтера. У розроблюваному інтерфейсі передбачається використання фільтрів для полегшення знаходження існуючих записів у БД.

1.4 Алгоритми роботи з системою обліку

Загальний алгоритм з системою обліку продукції підприємства що виготовляє канцелярські вироби складається з таких кроків:

- авторизація співробітника;
- вибір сутності, записи якої підлягатимуть редакції або створенню;
- редагування існуючих записів або створення нових;
- збереження змін;

Схематично даний алгоритм зображено на рис. 1.6.

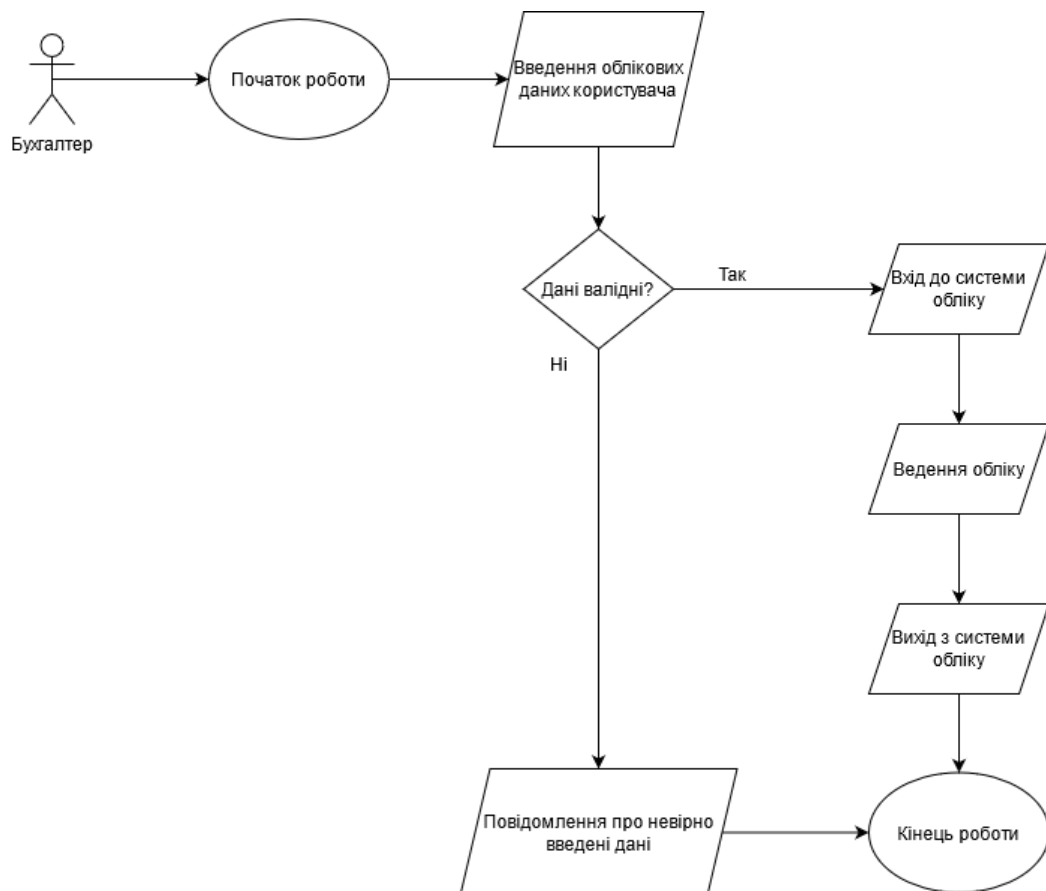


Рисунок 1.6 – Блок-схема алгоритму роботи з системою обліку

2 ТЕХНОЛОГІЯ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ РОЗРОБЛЕНИХ МОДЕЛЕЙ ТА АЛГОРИТМІВ

2.1 Порівняння існуючих методів програмної реалізації

Основним інструментом для реалізації інформаційної системи є мова програмування, це база для створення будь-якого програмного забезпечення. Оскільки у даній роботі розроблюється веб-додаток, то розглядатимуться лише серверні мови програмування – бекенд. Бекенд - це прошарок, який з'єднує інтерфейс з базою даних, дозволяє отримувати відповіді від сервера і забезпечує роботу сайту.

Розглянемо найпопулярніші мови для бекенду.

Java

Java - одна з найпопулярніших мов програмування. Вона універсальна і використовується вже більше 20 років.

Універсальність забезпечується віртуальною машиною Java (Java Virtual Machine, JVM). У багатьох мовах під час компіляції програма переводиться в код, який може працювати по-різному на різних пристроях або платформах. В Java цієї проблеми немає. JVM грає роль проміжного рівня - з програми на Java вона робить код, який може виконуватися на будь-якому комп'ютері незалежно від того, де код був скомпільований.

У Java велика спільнота, і в інтернеті можна знайти відповіді практично на будь-які питання про мову.

Що можна робити на Java

Технології Java можуть використовуватися для багатьох завдань, серед яких:

- розробка мобільних додатків під Android;
- розробка бекенд веб-сайтів;
- розробка API для роботи з базами даних;
- цифрова обробка зображень;
- створення настільних додатків;

- програмування мережевих задач;

PHP

PHP використовують близько 78% всіх сайтів. Мова з'явилась в 1995 році, коли було не так багато можливостей для створення динамічних веб-сторінок. PHP динамічно типізований, і один і той же фрагмент коду може вести себе по-різному в залежності від контексту, що робить програми на PHP складними для масштабування і іноді повільними.

PHP - відмінна мова для по ряду причин:

- він прощає помилки: ви можете запустити програму, і вона буде виконуватися, поки не досягне ділянки з проблемним кодом;
- у мови велике співтовариство, а для новачків є багато навчальних матеріалів. Мова постійно оновлюється.
- встановити і налаштувати PHP досить легко в порівнянні, наприклад, з Ruby on Rails;

Що можна робити на PHP

Згідно з офіційним сайтом PHP, використовуючи PHP можна:

- збирати дані форм (введення логіна / пароля та інше);
- створювати динамічний контент на сторінках;
- відправляти і отримувати куки;
- писати скрипти в командному рядку;
- виконувати сценарії на стороні сервера;
- розробляти настільні додатки;

NET (C #, VB)

Фреймворк з відкритим вихідним кодом ASP.NET від Microsoft використовується для створення веб-сайтів за допомогою таких мов, як Visual Basic (VB), C #, F # та інших.

.NET працює на основі архітектурного шаблону MVC (Model-View-Controller, Модель-Представлення-Контролер). Контролер приймає запити користувача і взаємодіє з моделлю для обробки даних. Потім результат передається в уявлення і відображається у вигляді інтерфейсу веб-сторінки.

Викладений у відкритий доступ в 2016 році, .NET може інтегруватися з iOS, Linux і Android через .NET Core. Він дуже стабільний і надійний, що робить його популярним вибором для бізнесу. Оскільки .NET - продукт Microsoft, у нього досить хороша підтримка.

C

C # - високорівнева мова програмування. Це означає, що розробники можуть писати на ньому програми, незалежні від архітектури процесора конкретного комп'ютера. C # популярний серед розробників, тому що він володіє деякими перевагами C ++, але на ньому простіше писати код і уникати при цьому грубих помилок.

VB

Visual Basic - це нащадок BASIC, який успадкував його стиль і поєднує в собі елементи ООП. Це проста мова для початківців: він широко поширений і має нескладним синтаксисом. VB часто застосовують для прототипування. Недоліком програмування на VB є великий обсяг пам'яті, необхідний для установки і запуску інструментів розробки.

Що можна робити на .NET

За допомогою .NET можна:

- створювати настільні додатки;
- створювати мобільні додатки;
- створювати веб-додатки та ігри;
- працювати з великими даними;

Ruby + Ruby on Rails

Ruby on Rails - це веб-фреймворк на мові програмування Ruby. Ruby on Rails має набір готових інструментів, які дозволяють швидко виконувати базові завдання.

Ruby досить лаконічний і не вимагає багато коду для бекенд, що дозволяє розробникам швидко створювати і запускати прототипи (цим Ruby

схожий на Python). Популярність Ruby виросла на початку 2000 років, але з тих пір помітно знизилася.

Що можна робити на Ruby

Ruby дозволяє:

- автоматизувати повторювані завдання;
- створювати веб-додатки;
- писати мобільні додатки та ігри;
- створювати прототипи;

Python

Python став дуже популярною мовою програмування. Він використовується як для веб-розробки, так і для створення настільних додатків. У мережі можна знайти величезну кількість навчальних сайтів, навчальних посібників і керівництв по Python, що робить його доступним мовою для новачків. Крім того, синтаксис Python простий і легкий для розуміння в порівнянні з іншими мовами. Python підтримує об'єктно-орієнтоване, функціональне і аспектно-орієнтоване програмування, а також це динамічно типізована мова з відкритим вихідним кодом.

Що можна робити на Python

Деякі з найбільш популярних застосувань Python:

- кросплатформені shell-скрипти;
- швидка автоматизація;
- веб розробка;
- data science;
- machine learning;

JavaScript

JavaScript - гнучка мова, яку можна використовувати як для фронтенду, так і для бекенду. Це гарна мова для початківців, оскільки в ньому мало налаштувань, і можна почати писати код прямо в браузері. Гнучкість JavaScript часом обходиться дорого: скрипти працюють повільно, їх складно підтримувати і масштабувати, як і в більшості динамічно типізованих мов.

У JavaScript велика спільнота, і для його вивчення в мережі є багато корисних матеріалів. Значно розширює можливості JS програмна платформа Node.js. З її допомогою код, написаний на JS, можна запускати без браузера на бекенд. А наявність величезної кількості готових рішень в пакетної екосистемі npm дозволяє розробнику не витрачати час на створення більшості типових рішень.

Що можна робити на JavaScript

З JavaScript створюються веб-додатки і програмне забезпечення, а саме:

- фронтенд і бекенд веб-сайтів;
- мобільні додатки;
- настільні додатки;
- ігри;
- веб-сервери;
- презентації;

Go

Go (часто також Golang) - компільована багато-поточна мова програмування, розроблена всередині компанії Google. Мова Go розроблялась для створення високоефективних програм, що працюють на сучасних розподілених системах і багатоядерних процесорах.

Go - мова з суворою статичною типізацією. Має досить лаконічний і простий синтаксис, заснований на Cі, але істотно доопрацьований, з великою кількістю синтаксичного цукру.

Що можна робити на Go

- писати мережеві інструменти;
- розробляти ПО для розподілених систем;
- писати ПО, що обробляє великі обсяги інформації;

Майже все сучасне програмування побудовано на принципах ООП, тому цілком актуально буде використовувати дані принципи в розробці системи обліку.

Об'єктно-орієнтоване програмування (скорочено ООП) - це парадигма розробки програмних систем, в якій додатки складаються з об'єктів.

Об'єкти - це сутності, у яких є властивості і поведінку. Зазвичай об'єкти є екземплярами якого-небудь класу. Наприклад, в грі може бути клас Character (персонаж), а його екземплярами будуть hero або prc.

Властивості - це дані, які пов'язані з конкретним об'єктом:

- здоров'я;
- окуляри;
- гроші;
- сила;
- спритність;
- інтелект;
- швидкість;
- координати;

Поведінка об'єкта визначається за допомогою методів - спеціальних блоків коду, які можна викликати з різних частин програми. Наприклад, у того ж об'єкта Character можуть бути наступні методи:

- йти;
- атакувати;
- говорити;
- підібрати;
- викинути;
- використовувати;

Використовуючи ці властивості і методи, можна значно прискорити розробку, зробити код більш читабельним. До того ж самому програмісту простіше складати код, якщо він думає з допомогою об'єктів. Тобто він не пише якусь функцію, яка буде робити щось для програми в цілому. Замість цього він подумки розділяє додаток на окремі компоненти і продумує їх властивості та поведінку.

Таку парадигму використовують багато популярних мов:

- C #;
- Java;
- Python;
- JavaScript;
- PHP;
- Kotlin;
- Swift;
- Objective-C;
- C ++;

Переваги і недоліки об'єктно-орієнтованого програмування представлені у таблиці 2.1.

Таблиця 2.1 – Переваги і недоліки об'єктно-орієнтованого програмування

Переваги	Недоліки
Легко читається. Не потрібно вишукувати в коді функції і з'ясовувати, за що вони відповідають.	Споживає більше пам'яті. Об'єкти споживають більше оперативної пам'яті, ніж примітивні типи даних.
Швидко пишеться. Можна швидко створити суті, з якими повинна працювати програма.	Знижує продуктивність. Багато речей технічно реалізовані інакше, тому вони використовують більше ресурсів.
Простіше реалізувати великий функціонал. Так як на написання коду йде менше часу, можна набагато швидше створити додаток з безліччю можливостей.	Складно почати. Парадигма ООП складніше функціонального програмування, тому на старт йде більше часу.
Менше повторень. Не потрібно писати однотипні функції для різних сутностей.	

ООП - складна, але ефективна парадигма програмування. У сучасності являється найпопулярнішою парадигмою у програмуванні. Схематично використання ООП зображено на рис. 2.1.

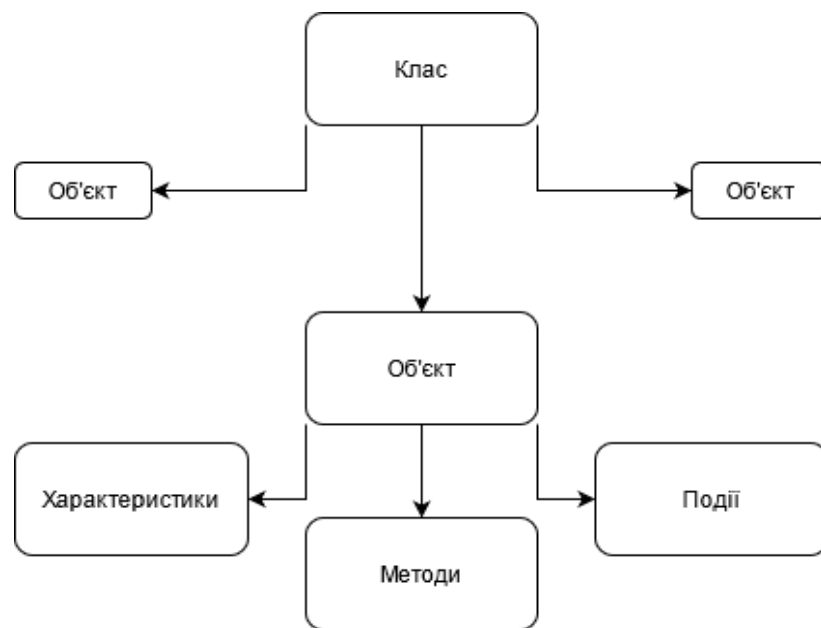


Рисунок 2.1 – Схема використання ООП

Процедурне програмування - це тип програмування, в якому інструкції для вирішення завдання виконуються одна за одною, зверху вниз, іноді виникають зміни в їх послідовності. Коли програма стає складнішою на допомогу приходять методи. Але сучасні програми настільки складні, що навіть поділ на методи не роблять програму простіше. І тут на допомогу приходить об'єктно-орієнтоване програмування.

Процедурне програмування можна порівняти з будівництвом маленького будиночка - немає необхідності витрачати час і ресурси на продумування архітектури. ООП ж схоже на будівництво складної архітектурного споруди, де дуже важливо продумати всі деталі, і тільки потім приступати до програмування (споруди).

Істотною перевагою розбиття програми на окремі процедури є те, що кожену процедуру можна писати і налагоджувати окремо, локалізую таким

чином місце появи помилки. Крім того, окремі вже налагоджені процедури можуть потім використовуватися вже в готовому вигляді в інших програмах.

Таким чином, ідеологія процедурного програмування може бути описана приблизно так. Розбиваємо вихідну задачу на сукупність підзадач меншого обсягу, потім визначаємо типи даних і алгоритми, необхідні для вирішення кожної підзадачі, після цього пишемо і налагоджуємо підпрограму для вирішення кожної підзадачі або використовуємо вже готову, і, нарешті, пишемо основну програму, що використовує окремі процедури для вирішення вихідної задачі.

Перевагами процедурного програмування є ясність і логічність розробки програмного забезпечення завдяки розподіленню вихідної задачі на підзадачі меншого обсягу, порівняльна простота проєктування процедурно-орієнтованих програм, ефективність одержуваного коду, так як всі сучасні процесори мають вбудовані апаратні засоби підтримки процедурного програмування (виклик процедур).

Слід особливо підкреслити, що поширене досі думка про те, що процедурне програмування є застарілим підходом і йому завжди слід віддати перевагу об'єктно-орієнтована стиль, є в корені не вірним. Процедурне і об'єктно-орієнтоване програмування є двома самодостатніми стилями, кожен з яких практично більш зручний і вигідний в кожній конкретній ситуації. Жоден з цих підходів, як стало очевидно до теперішнього часу виходячи з практичного досвіду, не скасовує іншого. В якості підтвердження цієї точки зору можна вказати на те, що ядра всіх сучасних операційних систем написані виключно з використанням процедурного програмування.

Для розробки інформаційних систем використовують принцип ООП, з цього виходить, що мова програмування для поставленого завдання повинна підтримувати принцип ООП. Серед вищеперерахованих мов програмування, виходячи з рис. 2.2, можна виділити три основних лідери – РНР, Ruby, Python.

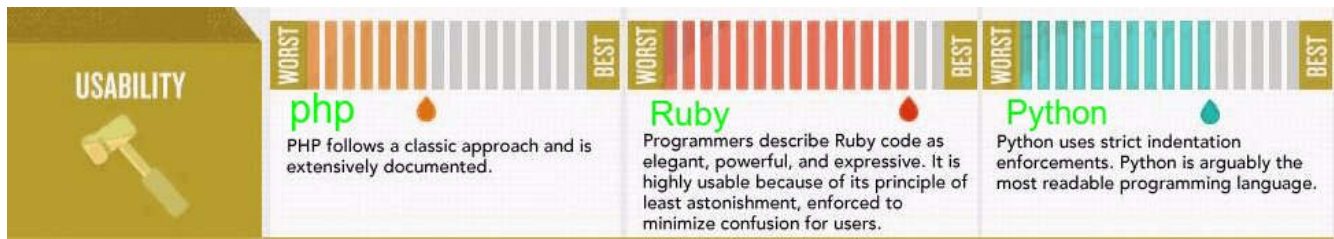


Рисунок 2.2 – Порівняльна характеристика PHP, Ruby, Python за рівнем “Usability”

Простота навчання або «крива навченості» - це ще один важливий параметр, який також враховується при виборі мови програмування для професійного зростання або наступного проєкту по розробці веб-додатків. Мова з високими значеннями кривої навченості легкий для розуміння і містить чистий вихідний код. З трьох розглянутих мов найкраща крива навченості(див. рис. 2.3) у Python, на другому місці PHP, на третьому - Ruby-on-rails.

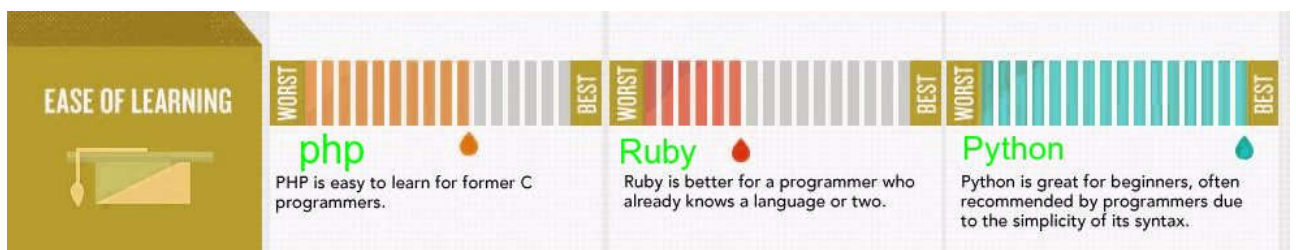


Рисунок 2.3 – Крива навченості PHP, Ruby, Python

Найбільш обговорювана мова програмування може бути корисна у багатьох відношеннях. Якщо який-небудь мову популярний або його активно обговорюють, він збирає велика спільнота користувачів, яке в найкоротший час може допомогти в рішенні вашого питання. Разом з тим, найбільш обговорюваний мова має хорошу підтримку. На зображенні знизу (рис. 2.4) наводиться статистика популярності мов програмування.

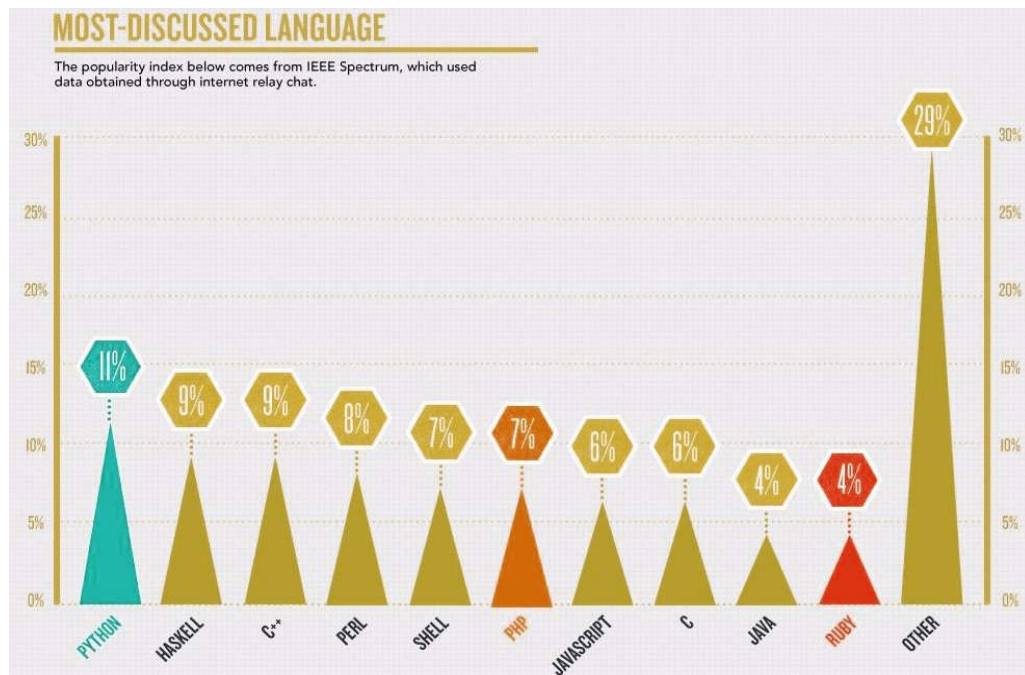


Рисунок 2.4 – Статистика популярності мов програмування

Виходячи з вищенаведеного порівняльного аналізу, можна виділити, що мова Python вважається найпопулярнішою, досить простою у вивченні, характеризується високим рівнем юзабіліті. Дана мова, хоч і має початок у 90х роках минулого століття, у сьогоденні є досить сучасною та універсальною, характеризується простотою створення програмного забезпечення та ефективному створенні веб-додатків, підтримкою принципів ООП[8]. Для виконання поставленої задачі прийнято рішення використати мову програмування Python.

2.2 Обраний метод програмної реалізації

Python - це універсальний сучасний ЯП високого рівня, до переваг якого відносять високу продуктивність програмних рішень і структурований, добре читабельний код. Синтаксис Python максимально полегшений, що дозволяє вивчити його за порівняно короткий час. Ядро має дуже зручну структуру, а широкий перелік вбудованих бібліотек дозволяє застосовувати

значний набір корисних функцій і можливостей. ЯП може використовуватися для написання прикладних програм, а також розробки WEB-сервісів.

Python[10] може підтримувати широкий перелік стилів розробки додатків, в тому числі, дуже зручний для роботи з ООП і функціонального програмування.

Один з найпопулярніших інтерпретаторів мови - CPython, написаний на Сі. Поширюється це середовище розробки безкоштовно по вільній ліцензії. Інтерпретатор підтримує більшість популярних платформ. Пітон активно розвивається.

Приблизно раз в 2 роки виходять оновлення. Важливою особливістю мови є відсутність таких стандартів кодування як ANSI, ISO і деяких інших, вони працюють завдяки інтерпретатора.

Мова почала розроблятися в другій половині 80-х років минулого століття. Автором Python став програміст з Нідерландів на ім'я Гвідо ван Россум. Спочатку мова мала стати об'єктно-орієнтованим. Фактично, це була мова сценаріїв, тобто скриптова мова. У лютому 1991 року ван Россум опублікував вихідний код мови в одній з груп.

Основними факторами успіху Python стали вдалий вибір місця презентації в популярному і масовому професійному співтоваристві в поєднанні з дійсно простим кодом і широкими можливостями.

Згодом Гвідо створив спеціалізований портал PEP, де йде регулярне обговорення щодо розвитку та покращення продукту. У 2008 року з'явилися велике оновлення мови - Python 3.0.

Версія продукту відома так само під назвою Py3k. У цій версії були усунені багато ключових недоробки в архітектурі ядра. Що було важливо - нова версія продукту зберегла повну сумісність з більш старими варіантами. Сьогодні розробниками підтримується дві лінії - Python 3.x і 2.x/

Пітон - не самий «молодий» мову програмування, але і не занадто старий. До моменту його створення вже існували такі «монстри», як Паскаль або Сі. А тому при створенні ЯП автори намагалися взяти найкраще з різних платформ для розробників. Фактично Python вдає із себе своєрідний «джем»

вдалих рішень більш ніж з 8 різних мов. Наприклад, байт компіляція з'явилася ще до створення Пітона, але була дуже вдала в нього інтегрована.

Пітон підтримує практично всі поширені операційні системи. Він може прекрасно працювати на кишенькових комп'ютерах, так і на великих серверах. У разі, якщо платформа значно застаріває, вона виключається з підтримки ядра. Наприклад, версії мови, починаючи від 2.6, вже не працюють з платформами Windows 95, 98 і ME. У разі необхідності можна скористатися більш старими версіями, відмовившись від застосування сучасних інструментів мови. І тоді додаток буде працювати в тому числі з цими ОС. Для старих версій періодично виходять патчі. Мова також може підтримувати роботу з віртуальною машиною Java.

ЯП має чітко структуроване семантичне ядро і досить простий синтаксис. Все, що пишеться на цій мові, завжди легко читаємо. У разі необхідності передати аргументи мову використовує функцію call-by-sharing.

Набір операторів в мові цілком стандартний. Зручна особливість синтаксису - це форматування тексту коду за допомогою розбивки їх на блоки за допомогою відступів, які створюють натисканням клавіш «Space» і «Tab». У синтаксисі відсутні фігурні або операторні дужки, що позначають початок і кінець блоку. Таке рішення помітно скорочує кількість рядків тіла програми і привчає програміста дотримуватися хороший стиль і акуратність при написанні коду.

Синтаксис Python має на увазі обов'язкове визначення типу даних для змінних, констант, масивів, списків і т.д. Основні типи нічим не відрізняються від інших мов з жорстко заданою типізацією.

Найважливіші типи:

- числові: цілі, дробові, речові з плаваючою точкою, комплексні;
- логічні: тип для зберігання значень алгебри логіки - «істина» або «брехня»;
- строкові: містять символи Юнікоду, в тому числі, html-код;
- списки - впорядковані масиви змінних;

- кортежі - масив упорядкованих констант, тобто значень, які не можуть змінюватися в процесі роботи;
- безлічі - масиви неупорядкованих даних;
- словники - спеціалізований масив, що складається з пари - «ключ» - «значення»;
- байти, масиви байтів - зазначені області пам'яті для зберігання зображень (jpg, gif і т.д.), pdf-документів та інших файлів;

Тут об'єктами вважаються практично всі конструкції. А тому власні типи мають, в тому числі, класи, файли або методи.

Всі бібліотеки на цій платформі прописуються як модулі. Перевагою такої концепції є можливість зібрати кілька модулів в пакет. Модуль може перебувати в архіві або безпосередньо в каталозі. Мова підтримує два види таких модулів - створення засобами Python або вже перетворені в машинний код з будь-якої мови. Модулі оформляються як окремі файли і завантажуються пакетами кожен в свій каталог.

Короткий перелік можливостей:

- будь описаний клас одноразово вдає із себе і об'єкт;
- функція множинного спадкоємства;
- підтримка віртуальних функцій;
- можливість легко керувати іменами приховувати їх особливими мітками;
- можливість життям об'єкта і розподіл пам'яті;
- управління роботи операторів як символьних, так і логічних;

Можливість імітувати поле

- управління полями - як прямий, так і частковий доступ;
- контроль над найпоширенішими операціями. Від глибокого до ітерації по об'єкту;
- можливо створювати тригери і класи;

Python відноситься до найбільш затребуваних і популярних ЯП, про що свідчать численні рейтинги і аналіз пропозицій на ринку розробки

програмних продуктів. Він досить простий, а тому вивчення мови не займе занадто багато часу. Так що, настійно радимо ознайомитися з цим прекрасним інструментом для створення ваших програм.

Django web-framework[11].

Django створили розробники видання Lawrence-Journal World. Цією газеті знадобився сайт, щоб публікувати новини в інтернеті. Програмісти Едріан Головатий і Саймон Віллісон створили веб-додаток і зробили його публічним. Навколо Django швидко сформувалося активна спільнота. Фреймворк став стрімко розвиватися зусиллями волонтерів. Значну роль в успіху Django зіграли кілька відомих сайтів, які використовували цей фреймворк. В їх число входять Pinterest, Dropbox, Spotify, сайт The Washington Post. В даний час співтовариство Django включає більше 11 тис. Розробників з 166 країн світу.

В Django реалізований принцип DRY (do not repeat yourself). Завдяки цьому скорочується час створення сайтів. Тобто при використанні Django вам не потрібно кілька разів переписувати один і той же код. Фреймворк дозволяє створювати сайт з компонентів. Це можна порівняти з будівництвом фортеці за допомогою Lego.

Django підходить для розробки високонавантажених веб-додатків. Це можливо завдяки архітектурі фреймворка, про яку піде мова нижче.

Фреймворк Django написаний на мові програмування Python, тому його структура відповідає особливостям мови. Творці реалізували в Django патерн MVC, і він застосовується в поточній версії фреймворка. Архітектура MVC дозволяє розробнику працювати з візуальним представленням і бізнес-логікою додатка окремо. До речі, при роботі з Django фахівці частіше використовують термін MVT - Model-View-Template або модель-уявлення-шаблон. Компоненти MVT можна використовувати незалежно один від одного(див. рис. 2.5).

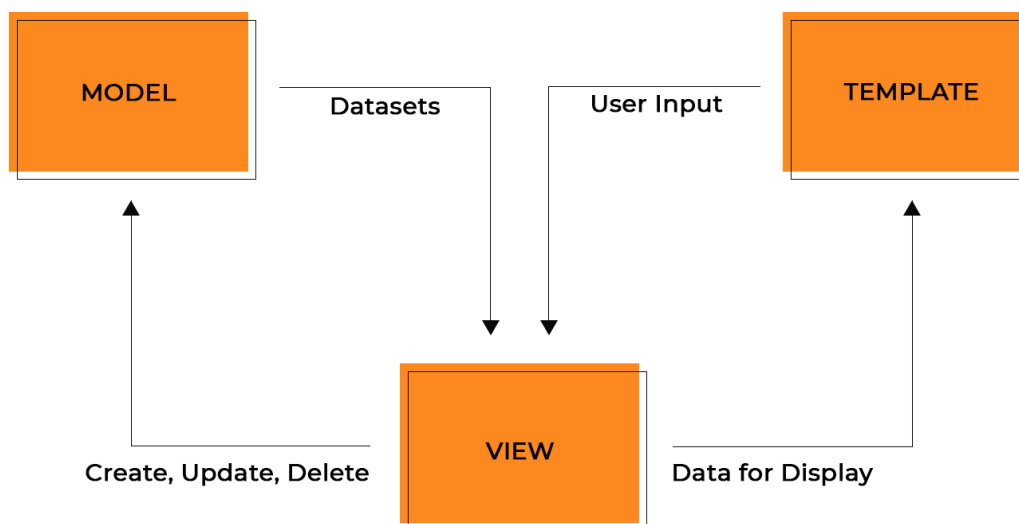


Рисунок 2.5 – Архітектура MVT

Документація Django визначає модель (model) як «джерело інформації про дані, в яких містяться ключові поля і поведінку даних». Зазвичай одна модель вказує на одну таблицю в базі даних. Django підтримує СУБД PostgreSQL, MySQL[6], SQLite.

Система управління базами даних (СУБД) - це комплекс програмно-мовних засобів, що дозволяють створити бази даних і управляти даними. Іншими словами, СУБД - це набір програм, що дозволяє організовувати, контролювати і адмініструвати бази даних. Більшість сайтів не можуть функціонувати без бази даних, тому СУБД використовується практично повсюдно.

Основні функції СУБД:

- керування даними в зовнішній пам'яті (на дисках);
- керування даними в оперативній пам'яті з використанням дискового кешу;
- журналізація змін (збереження історії), резервне копіювання і відновлення бази даних після збоїв;
- підтримка мов БД (мова визначення даних, мова маніпулювання даними);

Кожна СУБД ґрунтується на будь-якій моделі даних, це є одним з ознак класифікації. За моделлю даних СУБД бувають:

- ієрархічні. У цій моделі даних використовується уявлення БД у вигляді дерева, що складається з даних різних рівнів;
- мережеві. Дана модель є розширенням ієрархічного підходу. Ієрархічна модель має на увазі, що запис-нащадок може мати строго одного нащадка, в той час як в мережевій структурі нащадок може мати будь-яку кількість предків;
- реляційні. СУБД, орієнтовані на організацію даних як набір пов'язаних записів і атрибутів в двовимірній таблиці;
- об'єктно-орієнтовані. Для управління БД, заснованими на об'єктній моделі даних. Як правило ґрунтуються на об'єктно-орієнтованих мовах програмування;
- об'єктно-реляційні. Об'єднує в собі концепції реляційної моделі з додатковими об'єктно-орієнтованими можливостями;

Сьогодні, як і раніше найбільш популярними при створенні веб-додатків і сервісів залишаються реляційні бази даних. Для керування базами даних використовується мова SQL (Structured Query Language - структурована мова запитів)[16]. Спочатку SQL був інструментом роботи користувача з базою даних, однак з часом мова ускладнився і став швидше інструментом розробника, ніж кінцевого користувача.

В основі реляційних систем лежить реляційна модель даних. Принципи реляційної моделі були закладені в 1969-1970 рр. американським вченим Е.Ф. Коддом (E.F. Codd), в той час працювали в корпорації IBM. Будучи математиком за освітою, він привніс в область управління базами даних суворі математичні принципи і точність, яких не вистачало раннім системам. Хоча реляційний підхід утвердився не відразу, можна відзначити, що майже всі створені з кінця 70-х рр. продукти баз даних засновані саме на реляційному підході. Переважна більшість наукових досліджень в області баз даних протягом останніх 35 років також проводилося саме в цьому напрямку.

Різні рейтинги найпопулярніших СУБД очолюють MySQL, Microsoft SQL Server, PostgreSQL, SQLite.

MySQL.

Вважається однією з найпоширеніших СУБД. MySQL - реляційна СУБД з відкритим вихідним кодом, головними плюсами якої є її швидкість і гнучкість, яка забезпечена підтримкою великої кількості різних типів таблиць. Крім того, це надійна безкоштовна система з простим інтерфейсом і можливістю синхронізації з іншими базами даних. У сукупності ці фактори дозволяють використовувати MySQL як великим корпораціям, так і невеликим компаніям.

PostgreSQL

СУБД PostgreSQL - ще одна популярна і безкоштовна система. Найбільше застосування знайшла для управління БД веб-сайтів і різних сервісів. Вона універсальна, тобто підійде для роботи з більшістю популярних платформ. При цьому PostgreSQL - об'єктно-реляційна СУБД, що дає їй деякі переваги над іншими безкоштовними СУБД, в більшості є реляційними.

SQLite

SQLite - це програмна бібліотека, яка реалізує автономний, безсерверний, транзакційний механізм бази даних SQL з нульовою конфігурацією. SQLite є одним з найбільш швидкозростаючих механізмів баз даних, але він росте з точки зору популярності, а не його розміру. Вихідний код для SQLite знаходиться у відкритому доступі.

Переваги та недоліки кожної СУБД представлено у таблиці 2.2.

Таблиця 2.2 – Переваги та недоліки СУБД

СУБД	Переваги	Недоліки
Postgre	Високо функціональна і безкоштовна СУБД з відкритим кодом, хороша підтримка з боку співтовариства розробників	Місцями низька продуктивність при обробці великих обсягів інформації, мала популярність продукту
MS SQL	Високо функціональна, зручна при використанні середовища Windows	Платна, складна в адмініструванні, не сумісна зі всіма платформами
MySQL	Безкоштовна СУБД, має відкритий код, займає мало місця, легке адміністрування mysql з набором всього необхідного функціоналу, велика спільнота розробників, працює на безлічі платформ	незахищеність від втрати даних, підтримка тільки малих баз даних
SQLite	Підтримка мови запитів версії SQL92 (без особливостей типу процедур) База даних зберігається в одиночному файлі на диску Файл бази даних можна вільно розділити між процесами Підтримує бази даних розміром до 2-х терабайт (241 байт) Розміри строкових даних і BLOB даних обмежені тільки пам'яттю Мінімальний код на C / C ++ Проста і зручна бібліотека на C / C ++ Немає ніякої зовнішньої залежності від інших бібліотек Підтримка операційних систем WINDOWS (практично всі версії починаючи з WIN95), * unix	Важко масштабується Не підходить для великих баз даних(медіатека)

MySQL - це ефективна бібліотека, вбудована в додаток, який її використовує. Для максимально ефективної та швидкої розробки веб-сайту обліку продукції підприємства, яке виготовляє канцелярські вироби прийнято рішення використовувати MySQL.

Моделі містять інформацію про дані. Ці дані представлені атрибутами або полями. Оскільки модель являє собою простий клас, вона нічого не знає про інших рівнях Django. Взаємодія між рівнями відбувається через API. Модель відповідає за бізнес-логіку, методи, властивості і інші елементи, пов'язані з маніпуляцією даними. Також моделі дозволяють розробникам створювати, читати, оновлювати та видаляти об'єкти в базі даних.

Подання (view) вирішує три завдання: приймає HTTP-запити, реалізує бізнес-логіку, певну методами і властивостями, відправляє HTTP-відповідь у відповідь на запити. Тобто уявлення отримує дані від моделі і надає шаблонами (templates) доступ до цих даних або попередньо обробляє дані і потім надає до них доступ шаблонами. В Django реалізований потужний движок шаблонів і власну мову розмітки. Шаблони являють собою файли з HTML-кодом, за допомогою якого відображаються дані. Вміст файлів може бути статичним або динамічним. Шаблони не містять бізнес-логіки. Тому вони тільки відображають дані.

Також Django підходить для створення алгоритмічних генераторів, платформ для електронних розсилок, систем верифікації, систем фільтрації з динамічними правилами і складними параметрами, платформ для аналізу даних і складних обчислень, машинного навчання.

Адміністративна панель Django автоматично генерується при створенні програми. Це позбавляє розробника від необхідності створювати адміністративну панель вручну. За допомогою сторонніх додатків звичайну консоль управління Django можна вдосконалити і адаптувати під потреби

свого проєкту. Крім того, фреймворк дозволяє налаштовувати інтерфейс звичайної адміністративної панелі.

Написаний на Python код виходить читабельним і зрозумілим навіть непідготовленим людям. Це один з факторів, завдяки яким веб-додатки на Python вважаються SEO-дружніми. Django генерує семантичні URL. Їх також називають людино-зрозумілими URL або ЧПУ. У додатках на Django легко реалізуються інші функції, необхідні для пошукової оптимізації.

У популярних мовах програмування є бібліотеки, за допомогою яких зручно вирішувати спеціальні завдання. В бібліотеках можна знайти готові рішення: функції, класи, конфігурації і так далі. Завдяки таким рішенням розширюються можливості мови, а також спрощується створення додатків. Django підтримує використання бібліотек при розробці веб-додатків.

В Django реалізовано об'єктно-реляційне відображення (ORM), яке забезпечує взаємодію додатки з базами даних (БД). ORM автоматично передає дані з БД, наприклад, PostgreSQL або MySQL[7], в об'єкти, які використовуються в коді програми (див. рис. 2.6).

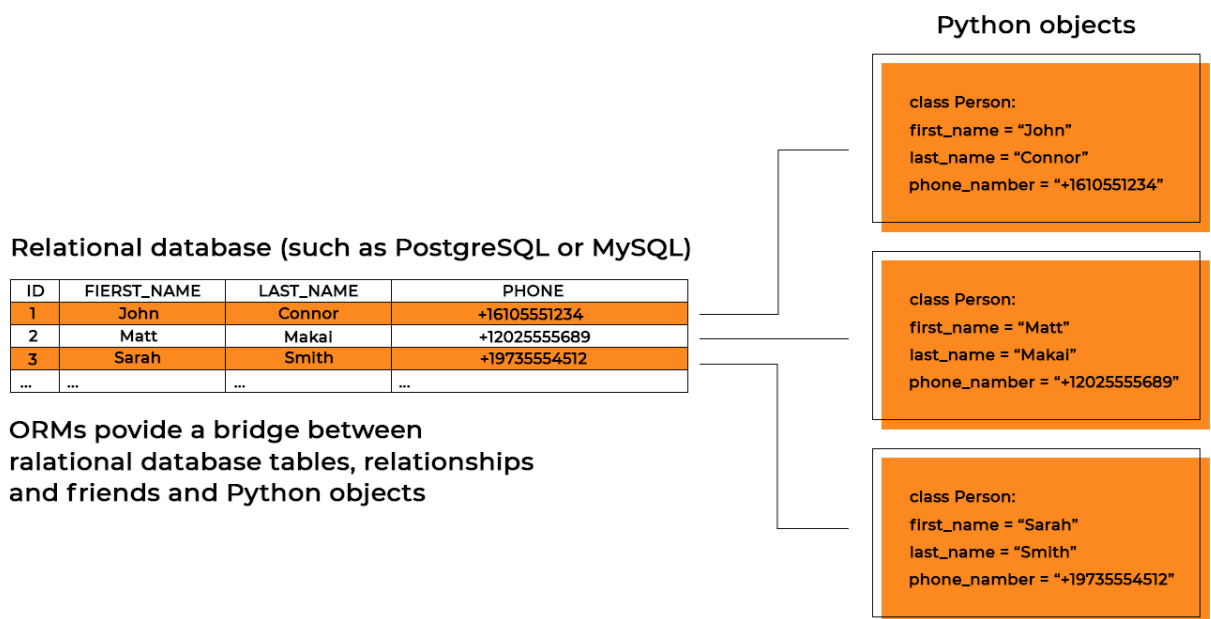


Рисунок 2.6 – Схема роботи ORM

ORM прискорює розробку прототипів і готових веб-додатків на Django. Розробнику навіть не потрібно знати мову, який використовується для взаємодії з базами даних.

Розробники вибирають Django завдяки таким характеристикам:

- поділ бізнес-логіки і візуальної частини на рівні архітектури;
- SEO-дружність;
- можливість розширення;
- розвинена інфраструктура: велика кількість бібліотек і плагінів;
- численне і дружнє співтовариство, завдяки якому легко шукати відповіді на складні питання;

Обраний метод програмної реалізації є ефективним, простим у використанні, підтримує принцип ООП[9].

Парадигма ОПП лежить в основі багатьох популярних мов. Го-мови: C ++, C #, Java, JavaScript, Objective-C, Python, PHP, Perl, Ruby, Scala, Swift та ін. Тобто топові мови, ряд мов загального призначення, мови для Android, веб і iOS розробки. Популярні сучасні мови – використовують мульти-парадигмальність, по принципам ООП можна писати і на мовах з іншими парадигмами.

ООП як стиль написання програм передбачає побудову структури, що складається з взаємодіючих об'єктів. В рамках ООП потрібно мислити об'єктами. Об'єкти розташовані в ієрархії, самостійні і якимось взаємодіють. Програма складається з модулів - блоків, які вирішують якісь завдання. Зміни в цих ділянках можуть не відбиватися на інших ділянках.

Об'єкти можуть бути абстрактними сутностями в програмі, а можуть уособлювати об'єкти фізичного середовища. Бізнес-процеси теж можна уявити як взаємодію і стану об'єктів. Можна зустріти такі аналогії, як виробництво, пристрій і водіння автомобіля, пристрій міста або будинку і ін. В світі ми всюди бачимо класи об'єктів, і об'єкти якимось взаємодіють в ієрархії. Очевидними перевагами ООП є:

- можливість легкої модифікації (при грамотному аналізі і проєктуванні);
- можливість відкату при наявності версій;
- більш легка розширюваність «Більш природна» декомпозиція програмного забезпечення, яка істотно полегшує його розробку;
- скорочення кількості міжмодульних викликів і зменшення обсягів інформації, що передається між модулями;
- збільшується показник повторного використання коду;

Розробка за допомогою мови Python вважається комфортною та легкою. Веб-розробка за допомогою Django поліпшує взаємодію з базою даних, спрощує створення бази даних, її сутностей(таблиць) та полегшує вивід даних у шаблони.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ РОЗРОБЛЕНИХ МОДЕЛЕЙ ТА АЛГОРИТМІВ

3.1 Реалізація проєктованих моделей

Моделі у Django створюються у файлах `models.py`. Оскільки Django використовує власну ORM, моделі - це класи, а об'єкти, що створюються – це об'єкти цих класів. У Django існує велика кількість типів полів[12] (атрибутів) для моделей, наприклад:

- `BinaryField ()`: зберігає бінарні дані;
- `BooleanField ()`: зберігає значення `True` або `False` (0 або 1);
- `NullBooleanField ()`: зберігає значення `True` або `False` або `Null`;
- `DateField ()`: зберігає дату;
- `TimeField ()`: зберігає час;
- `DateTimeField ()`: зберігає дату і час;
- `DurationField ()`: зберігає період часу;
- `AutoField ()`: зберігає цілочисельне значення, яке автоматично інкрементується, зазвичай застосовується для первинних ключів;
- `BigIntegerField ()`: представляє число - значення типу `Number`, яке вкладається в діапазон від -9223372036854775808 до 9223372036854775807. Залежно від обраної СУБД діапазон може трохи відрізнятись;
- `DecimalField (decimal_places = X, max_digits = Y)`: представляє значення типу `Number`, яке має максимум `X` розрядів і `Y` знаків після коми;
- `FloatField ()`: зберігає, значення типу `Number`, яке представляє число з плаваючою точкою;
- `IntegerField ()`: зберігає значення типу `Number`, яке представляє цілочисельне значення;
- `PositiveIntegerField ()`: зберігає значення типу `Number`, яке представляє позитивне цілочисельне значення (від 0 до 2147483647);

- `PositiveSmallIntegerField ()`: зберігає значення типу `Number`, яке представляє невелике позитивне цілочисельне значення (від 0 до 32767);
- `SmallIntegerField ()`: зберігає значення типу `Number`, яке представляє невелике цілочисельне значення (від -32768 до 32767);
- `CharField (max_length = N)`: зберігає рядок довжиною не більше `N` символів `TextField ()`: зберігає рядок невизначеної довжини;

Використовуючи вищенаведену інформацію, прийнято рішення реалізувати моделі наступним чином.

Продукт

```
class Product(models.Model):
    prod_type = models.ForeignKey(ProductType, on_delete=models.CASCADE,
    verbose_name="Тип продукту")

    def __str__(self):
        return f'{self.id} {self.prod_type.title}'

class Meta:
    verbose_name = 'Продукт'
    verbose_name_plural = 'Продукти'
```

Тип продукту

```
class ProductType(models.Model):
    title = models.CharField('Назва', max_length=100)
    chars = models.TextField('Характеристики', max_length=1000)
    price = models.IntegerField('Вартість')

    def __str__(self):
        return f'{self.title}, {self.price}, {self.chars}'
```

```
class Meta:
    verbose_name = 'Тип продукту'
    verbose_name_plural = 'Типи продуктів'
```

Звіт виготовлення продукції

```
class ProductionReport(models.Model):
    date = models.DateTimeField('Дата створення', auto_created=True,
editable=True, auto_now_add=True)
    products = models.ManyToManyField(Product, verbose_name="Продукти")

    def __str__(self):
        return f"Звіт виготовлення {self.date}"

class Meta:
    verbose_name = 'Звіт виготовленої продукції'
    verbose_name_plural = 'Звіти виготовленої продукції'
```

Звіт проданої продукції

```
class SellReport(models.Model):
    date = models.DateTimeField('Дата створення', auto_created=True,
editable=True, auto_now_add=True)
    products = models.ManyToManyField(Product, verbose_name="Продукти")
    customer = models.ForeignKey(Customer, on_delete=models.CASCADE)
    summary = models.IntegerField('Сума прибутку', editable=True, default=0)

    def __str__(self):
        return f"Звіт продажу {self.date}"

    def save(self, *args, **kwargs):
```

```

if self.id is None:
    super(SellReport, self).save(*args, **kwargs)

for a in self.products.all():
    self.summary += a.prod_type.price

super(SellReport, self).save(*args, **kwargs)

class Meta:
    verbose_name = 'Звіт проданої продукції'
    verbose_name_plural = 'Звіти проданої продукції'

```

ЗАМОВНИК

```

class Customer(models.Model):
    company = models.CharField('Компанія', max_length=100)
    address = models.CharField('Адреса', max_length=100)
    contacts = models.CharField('Контакти', max_length=200)

    def __str__(self):
        return self.company

class Meta:
    verbose_name = 'Замовник'
    verbose_name_plural = 'Замовники'

```

Всі зв'язки, що представлені на рис. 2 враховані та реалізовані за допомогою полів `ForeignKey()`[13] та `ManyToManyField()`.

ManyToManyField() дозволяє створювати відношення між продуктами та звітами без створення додаткової таблиці(див. рис. 3.1).

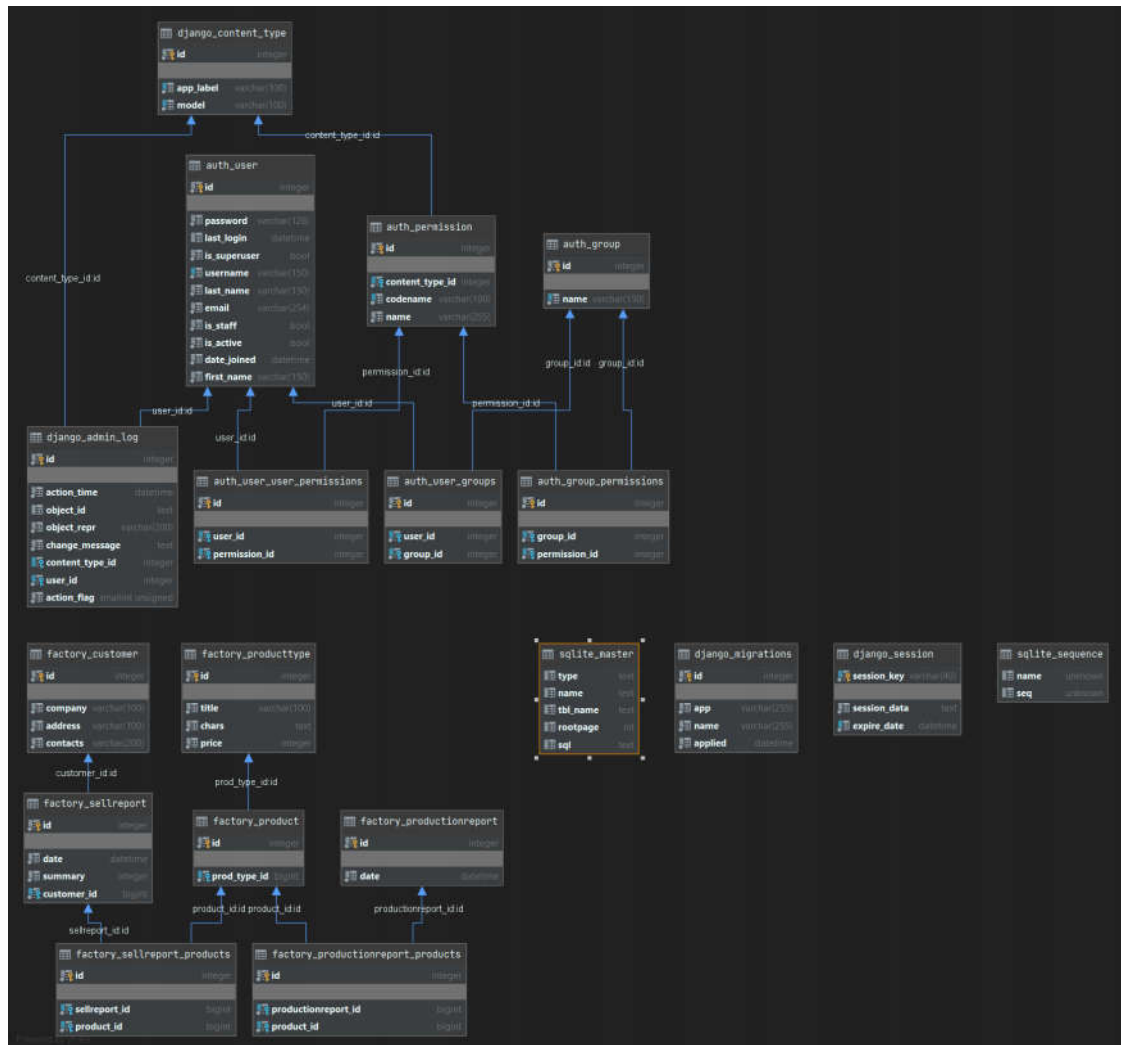


Рисунок 3.1 – Схема БД, що створена Django

Class Meta[14] – допоміжний клас, що встановлює метадані, у даному випадку – відображення даної моделі у адміністративній панелі як в одиночному випадку, так і в багатьох випадках.

Def `__str__()` – метод відображення об'єктів даної моделі, тобто які саме дані будуть відображатись у адміністративній панелі при перегляді об'єктів(див. рис. 3.2).

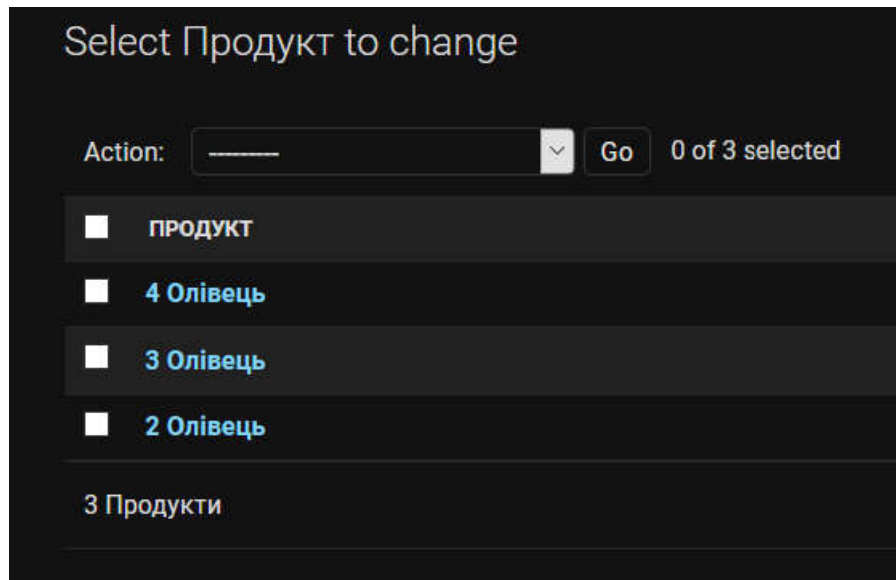


Рисунок 3.2 – Відображення об'єктів моделі за допомогою `__str__`

У звіті проданої продукції прийнято рішення створити поле `summary`, що відображає загальну суму прибутку. У момент створення звіту, дане поле можна заповнити власноруч, або, після збереження створеного звіту, сума прорахується автоматично, оскільки метод `save()` перетворений таким чином.

3.2 Реалізація користувацького інтерфейсу

Додаток Django admin може використовувати моделі для автоматичного створення частини сайту, призначеної для створення, перегляду, оновлення та видалення записів. Це може заощадити багато часу в процесі розробки, спрощуючи тестування моделей на предмет правильності даних. Воно також може бути корисним для управління даними на стадії публікації, в залежності від типу веб-сайту. Проект Django рекомендує це додаток тільки для управління внутрішніми даними (тобто для використання

адміністраторами, або людьми всередині організації), так як модельно-орієнтований підхід не обов'язково є найкращим інтерфейсом для всіх користувачів і розкриває багато зайвих подробиць про моделі.

Всі необхідні налаштування, які необхідно включити в admin додаток вашого веб-сайту, були зроблені автоматично, коли створено каркас проєкту. В результаті все, що необхідно зробити для того, щоб додати моделі в додаток admin, це зареєструвати їх.

Налаштування адміністративної панелі відбувається у файлі admin.py, де і потрібно зареєструвати всі створені моделі. Вигляд файлу admin.py зображено нижче:

```
from django.contrib import admin
from .models import *

@admin.register(Product)
class ProductAdmin(admin.ModelAdmin):
    pass

@admin.register(ProductType)
class ProductTypeAdmin(admin.ModelAdmin):
    pass

@admin.register(ProductionReport)
class ProductReportAdmin(admin.ModelAdmin):
    pass
```

```
@admin.register(SellReport)
class SellReportAdmin(admin.ModelAdmin):
    pass
```

```
@admin.register(Customer)
class CustomerAdmin(admin.ModelAdmin):
    pass
```

Якщо потрібно вивести всі поля кожної моделі у адміністративну панель, використовується синтаксис `pass`, що і даному випадку. Також, адміністративна панель може мати фільтри, пошук об'єктів моделі, наприклад:

```
@admin.register(Customer)
class CustomerAdmin(admin.ModelAdmin):
    pass

    list_filter = ('status', 'due_back')

    fieldsets = (
        (None, {
            'fields': ('book', 'imprint', 'id')
        }),
        ('Availability', {
            'fields': ('status', 'due_back')
        }),
    )
```

Загальний вигляд адміністративної панелі зображено на рис. 3.3.

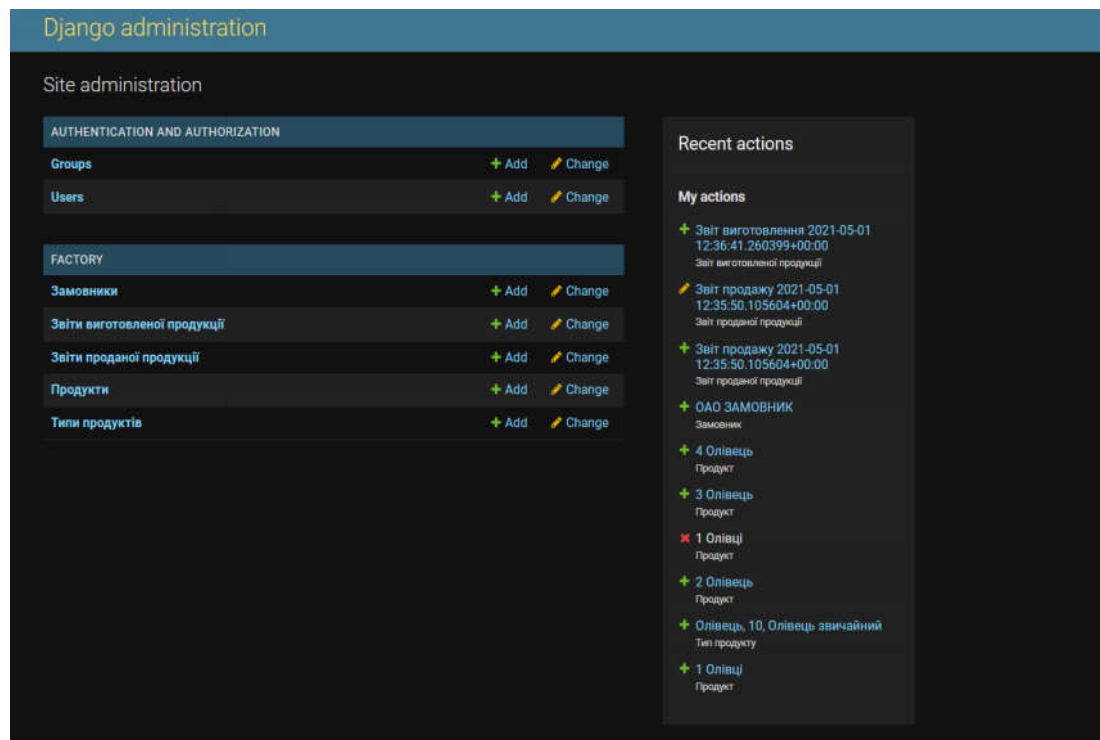


Рисунок 3.3 – Загальний вигляд адміністративної панелі Django

Однак, окрім звичайної адміністративної панелі, існує певна кількість додатків, що модифікують дану сторінку. Прикладом можуть слугувати Grappelli, DjangoSuit. Розглянемо детальніше кожен:

Grappelli[15] (див. рис. 3.4)

Інтерфейс адміністратора Django - це потужна програма для управління даними веб-сайтів - деякі можуть назвати це CMS. Grappelli додає послідовний вигляд та відчуття на основі сітки та деякі чудові функції.

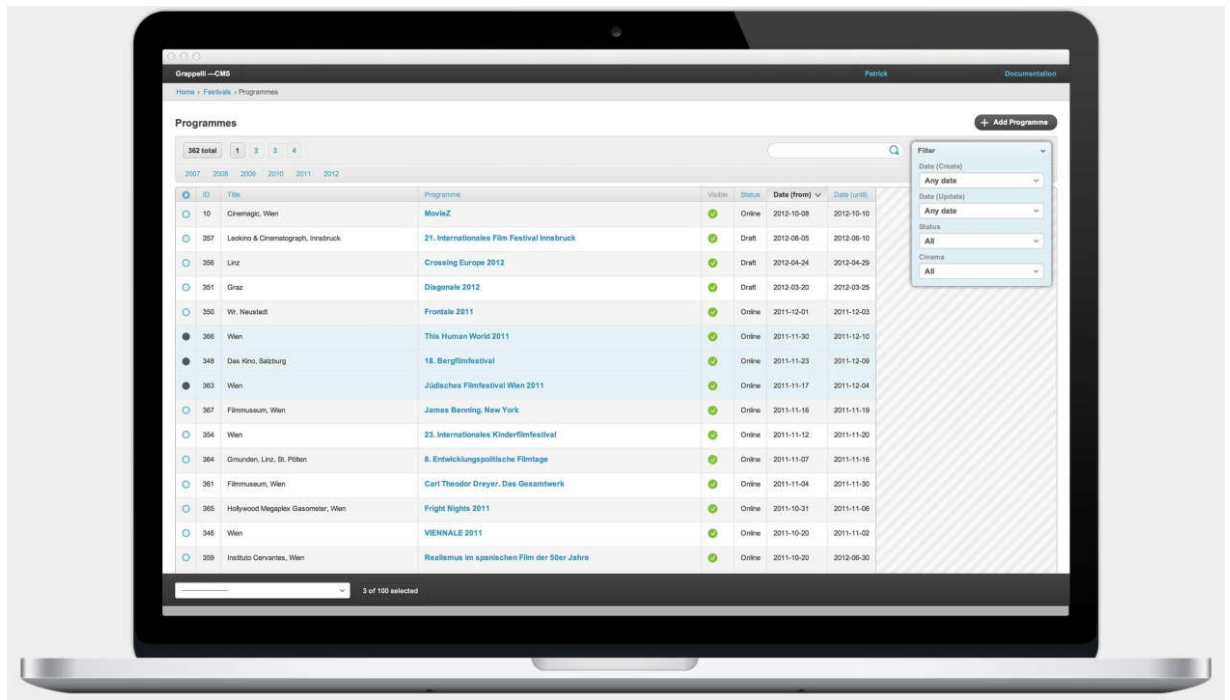


Рисунок 3.4 – Django-Grappelli

Grappelli - це альтернатива / розширення інтерфейсу адміністрування Django на основі сітки.

Django-suit (див. рис. 3.5)

Модифікована адміністративна панель для Django.

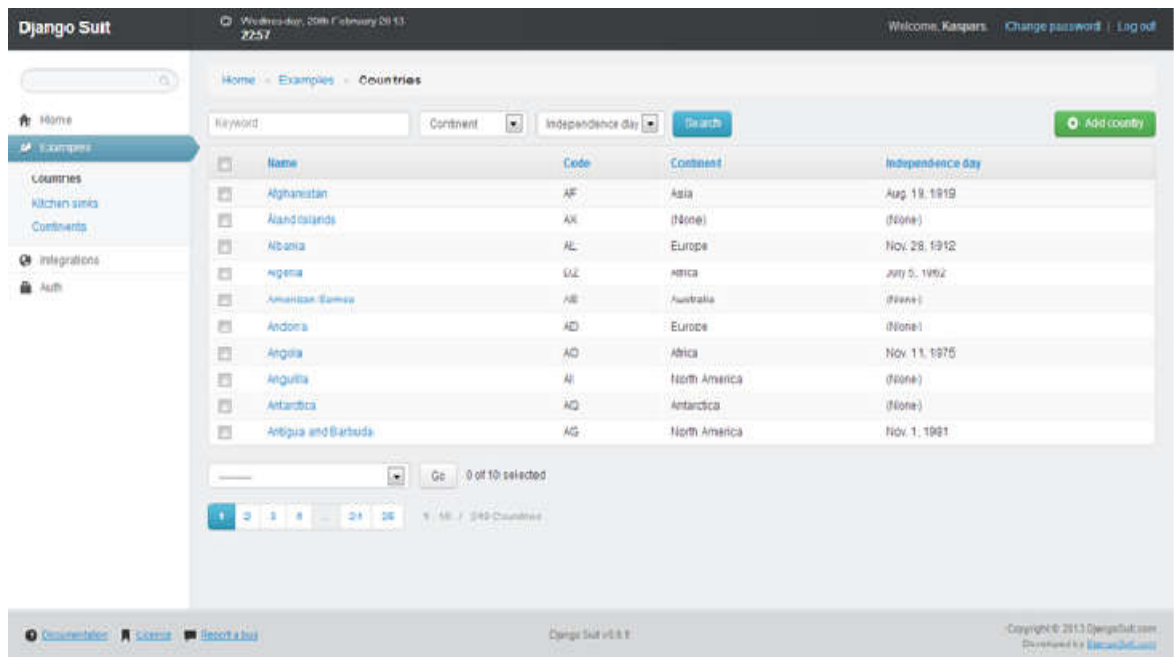


Рисунок 3.5 – Django-suit

- сучасний та професійний дизайн;
- покращені фільтри списків;
- сортування за списком та рядками;
- налаштування меню та пошук;
- корисні доповнення CSS / JS;
- на основі Twitter Bootstrap[17];
- проста установка;
- ніяких вимог до сторонніх розробників;
- підтримка сторонніх додатків: django-cms, django-filer та інші;

Функціонально нічого нового дані панелі не привносять, тільки зовнішній вигляд. Оскільки це додаткова опція, яка не критично необхідна, прийнято рішення не встановлювати модифіковані адміністративні панелі.

3.3 Інструкція використання реалізованої системи обліку

Перед початком роботи з Django у першу чергу слід встановити інтерпретатор Python версії 3.8 або вище. Для його встановлення потрібно перейти на сторінку <https://www.python.org/downloads/> та знайти посилання на завантаження останніх версій мови.

Після натискання на кнопку буде завантажено відповідну поточну ОС майстра установки Python. На ОС Windows при запуску інсталятора запускається вікно майстра установки(рис. 3.6).

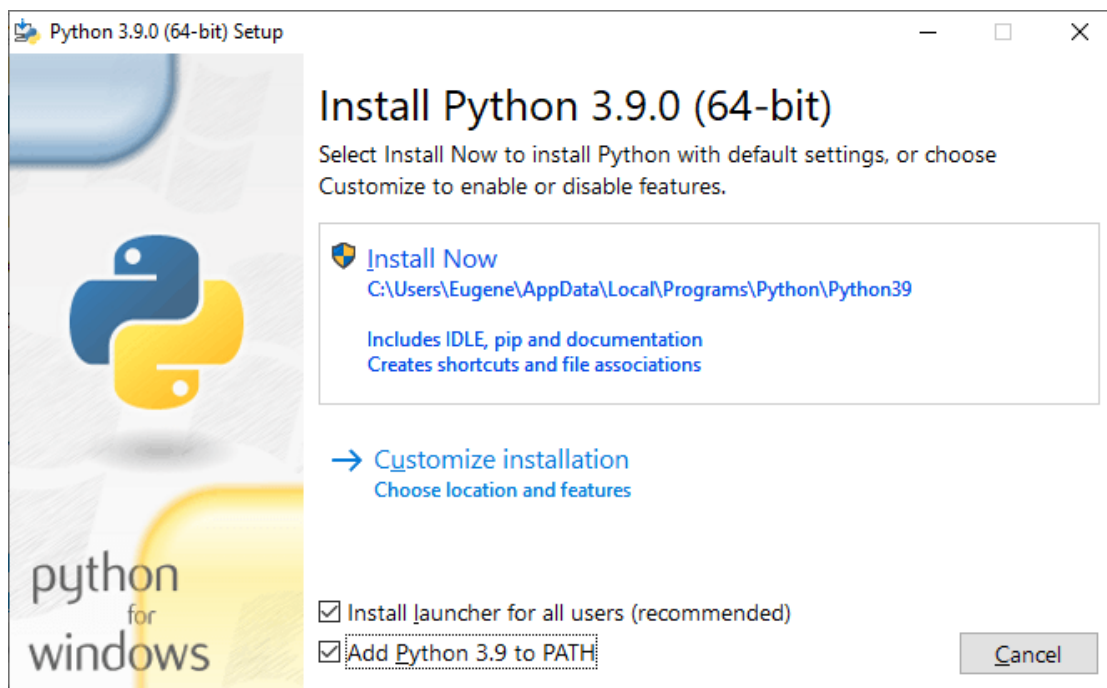


Рисунок 3.6 – Майстер установки Python(Windows)

Тут можна задати шлях, по котрому буде встановлений інтерпретатор. Можна залишити за замовчуванням `C: \ Users \ [ім'я_користувача] \ AppData \ Local \ Programs \ Python \ Python36 \`. Крім того, у самому низу потрібно відзначити флажок "Додати Python 3.9 до PATH", щоб додати шлях до інтерпретатора в змінних середовища.

Існують різні способи установки Django. Розглянемо рекомендований спосіб. Для установки буде потрібен пакетний менеджер `pip`. Менеджер `pip` дозволяє завантажувати пакети і управляти ними. Нерідко, при установці `python` також встановлюється і менеджер `pip`. У цьому випадку можна перевірити версію менеджера, виконавши в командному рядку / терміналі наступну команду:

`Pip -v`

Але якщо `pip` не встановлено, то побачимо помилку типу "`pip`" не є внутрішньою або зовнішньою командою, що виконується програмою або пакетним файлом. У цьому випадку треба встановити `pip`.

Віртуальне середовище або `virtualenv`[18] не є невід'ємною частиною розробки на Django. Однак її рекомендується використовувати, так як вона

дозволяє створити безліч віртуальних середовищ Python на одній операційній системі. Завдяки віртуальному середовищі додаток може запускатися незалежно від інших додатків на Python.

В принципі можна запускати додатки на Django і без віртуального середовища. В цьому випадку всі пакети Django встановлюються глобально.

Для роботи з віртуальним середовищем спочатку необхідно встановити пакет `virtualenv` за допомогою наступної команди

```
Pip install virtualenv
```

Тепер створимо віртуальну середу. Спочатку визначимо каталог для віртуальних середовищ, де будуть розташовуватися всі пов'язані файли і папки. Наприклад, нехай це буде каталог `C: \ virtualenv`. Перш за все перейдемо в командному рядку/ терміналі в цей каталог за допомогою команди `cd`. Потім для створення віртуального середовища виконаємо наступну команду:

```
Virtualenv Назва
```

Для використання віртуальне середовище треба активувати. І кожен раз, коли ми будемо працювати з проектом Django, пов'язану з ним віртуальне середовище треба активувати. Наприклад, активуємо вище створене середовище, яка розташовується в поточному каталозі в папці `hello`. Якщо наша ОС - Windows, то в папці `hello / Scripts /` ми можемо знайти файл `activate.bat`, який активує віртуальне середовище. Тому для Windows активація віртуального середовища буде виглядати таким чином:

```
hello\Scripts\activate.bat
```

Для Linux і MacOS активація буде проводитися за допомогою наступної команди:

```
source ~/.hello/bin/activate
```

Після закінчення роботи з віртуальним середовищем ми можемо її деактивувати. Для цього в тій же папці `hello / Scripts /` ми можемо знайти файл `deactivate.bat` і таким же чином запустити його.

Після активації віртуального середовища для установки Django виконаємо в консолі наступну команду

Pip install Django

Дана команда є фінальним кроком для встановлення Django.

Після цього потрібно перейти в директорію даного проєкту, та виконати команду

Python manage.py runserver

Дана команда запустить локальний сервер і дасть змогу використовувати даний проєкт за адресою 127.0.0.1:8000/.

Для доступу потрібна авторизація яка буде виглядати наступним чином(рис. 3.7).

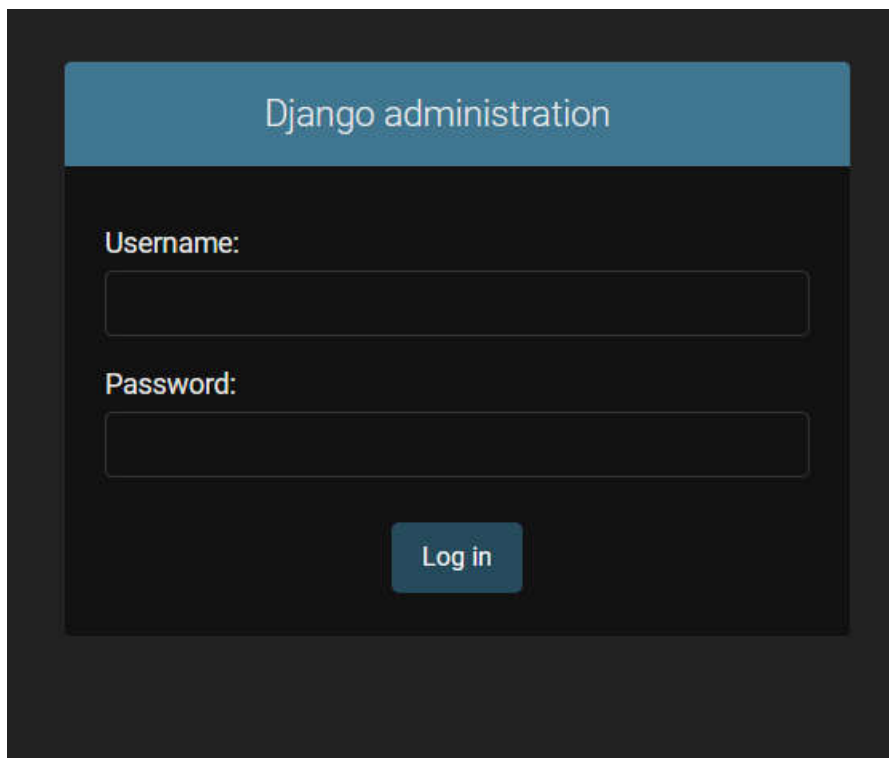


Рисунок 3.7 – Сторінка авторизації

Оскільки жодного користувача ще не створено, потрібно виконати команду

Python manage.py createsuperuser

Дана команда викликає скрипт створення супер-користувача(адміністратора), що виглядає наступним чином(рис. 3.8).

```

Tracking file by folder pattern: migrations
Username: admin
Email address:
Warning: Password input may be echoed.
Password: 123123
Warning: Password input may be echoed.
Password (again): 123123
This password is too short. It must contain at least 8 characters.
Bypass password validation and create user anyway? [y/N]: This password is too common.
This password is entirely numeric.
y
Superuser created successfully.

Process finished with exit code 0

manage.py@Project > |

```

Рисунок 3.8 – Створення адміністратора

Після даного кроку потрібно авторизуватись. Після авторизації сторінка проєкту буде виглядати як на рис. 15.

Для створення/редагування об'єктів певних моделей потрібно натиснути на назву моделі(див. рис. 3.9).

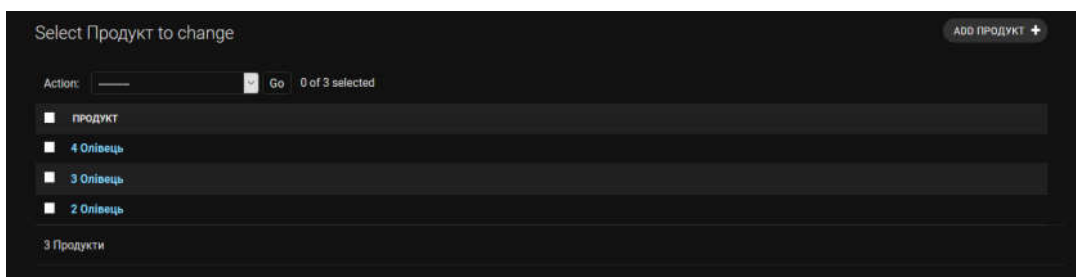


Рисунок 3.9 – Перехід до меню моделі «Продукти»

Доступні функції:

- редагування та видалення наявних об'єктів;
- створення нових об'єктів;
- фільтрація об'єктів за певними ознаками(якщо вказано в admin.py);

Створення нового об'єкту зображено на рис. 3.10.

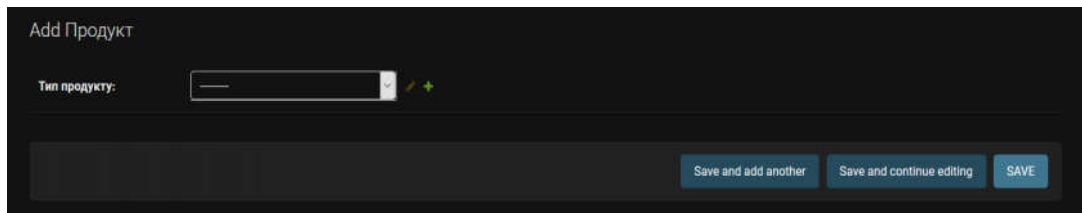


Рисунок 3.10 – Створення об'єкту «Продукт»

ID продукту встановлюється автоматично, а зв'язок «Продукт» - «Тип продукту» дозволяє одразу створювати залежні об'єкти (рис. 3.11).

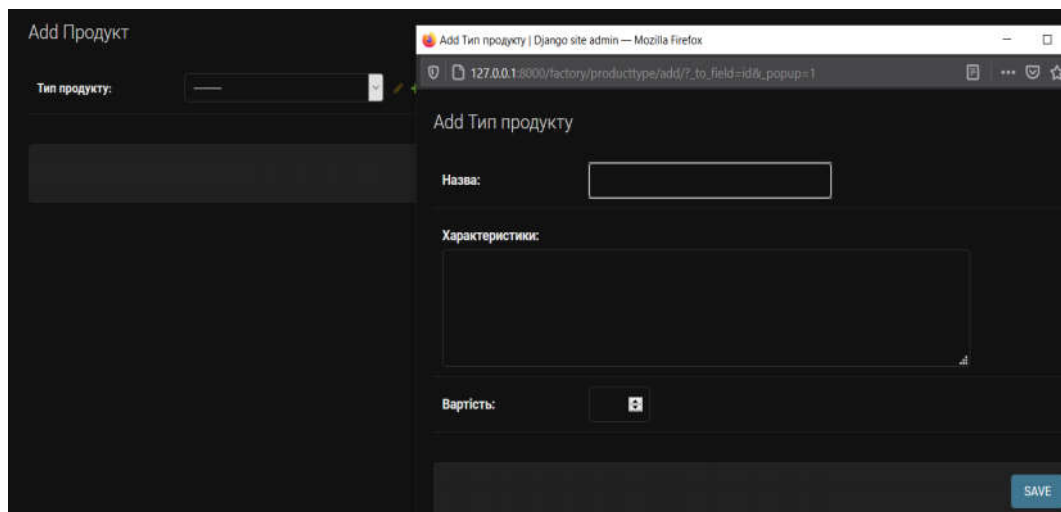


Рисунок 3.11 – Створення залежних об'єктів

Для видалення та редагування існуючих об'єктів потрібно перейти за посиланням на сторінку редагування об'єкту, що зображена його назвою, як на рис. 3.10. Процес редагування зображено на рис. 3.12.

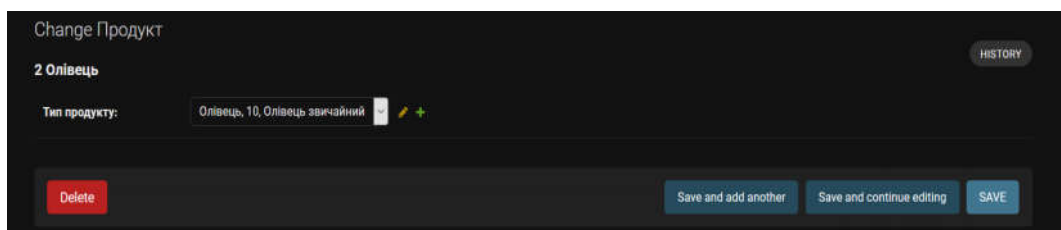


Рисунок 3.12 – Редагування об'єкту «2 Олівець»

Наведена вище інструкція дозволяє в повній мірі використовувати створений функціонал даного проєкту.

ВИСНОВКИ

В процесі роботи над випускною кваліфікаційною роботою була вивчена така предметна область як веб-сайт для ведення обліку продукції підприємства яке виготовляє канцелярські вироби. Були поставлені завдання, що визначають зміст інформаційної системи.

Наступним етапом стала розробка інформаційного забезпечення задачі, яка полягала в інформаційному аналізі всіх даних, а також визначення логічної структури бази даних.

При виборі системи управління базами даних були досліджені, проаналізовані та виявлені переваги і недоліки таких СУБД як MySQL, MS SQL Server, PostgreSQL, SQLite. Як використовуваної СУБД вибір зроблений на користь MySQL, а в якості мови програмування для розробки програми доступу до даних обраний Python, фреймворк Django. Це пов'язано з тим, що MySQL дозволяє ефективно обробляти великий обсяг даних.

При розробці інформаційної системи, був використаний для користувача інтерфейс за стандартом GUI. Інформаційна система створена для багатьох користувачів. Її користувачами бухгалтери, які спочатку здійснюють введення обліку продукції підприємства, яке виготовляє канцелярські вироби.

Інформаційна система є відкритою і має великий потенціал свого розвитку. Вона може змінюватися при зміні вимог, що пред'являються до предметної області.

Розроблена інформаційна система не претендує на комерційний повноцінний додаток, а створена у якості наукового дослідження. Розроблена система підлягає модифікації, видозміні та дозволяється до використання у подальших розробках та дослідженнях.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Гинзбург, В.М. Проектирование информационных систем в строительстве. Информационное обеспечение / В.М. Гинзбург. - М.: АСВ, 2008. - 368 с.
2. Головицына, М.В. Проектирование радиоэлектронных средств на основе современных информационных технологий: Учебное пособие / М.В. Головицына. - М.: Бином, 2016. - 503 с.
3. Емельянова, Н.З. Проектирование информационных систем: Учебное пособие / Н.З. Емельянова, Т.Л. Партыка, И.И. Попов. - М.: Форум, 2013. - 432 с.
4. Заботина, Н.Н. Проектирование информационных систем: Учебное пособие / Н.Н. Заботина. - М.: НИЦ ИНФРА-М, 2013. - 331 с.
5. Mikhail Vrubel / Михаил Врубель. - М.: Аврора, 2013. - 260 с.
6. MySQL руководство администратора. - М.: Вильямс, 2005. - 621 с.
7. Аткинсон MySQL. Библиотека профессионала/Аткинсон, Леон. - М.: Вильямс, 2014. - 624 с.
8. Г.С. Иванова Объектно-ориентированное программирование / Г.С. Иванова, Т.Н. Ничушкина, Е.К. Пугачев. - М.: МГТУ им. Н. Э. Баумана, 2014. - 368 с.
9. Г.С. Иванова Объектно-ориентированное программирование / Г.С. Иванова. - М.: Московский Государственный Технический Университет (МГТУ) имени Н.Э. Баумана, 2014. - 149 с.
10. Лутц М. Изучаем Python, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 1280 с.
11. Django documentation [Electronic resource] – Access mode: <https://docs.djangoproject.com/en/3.0/>
12. Аллен Дж. Тейлор. SQL для чайников, 7-е издание, 2012. – 416 с.

13. Томас Коннолли, Каролин Бегг. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. 2016. – 1440 с.
14. Tom Boellstorff, Bill Maurer. Data: Now bigger and better! 2015. – 339 с.
15. Grappelli [Electronic resource] – Access mode: <https://grappelliproject.com/>
16. Grappelli [Electronic resource] – Access mode: <https://djangosuit.com/>
17. Twitter Bootstrap [Electronic resource] – Access mode: <https://getbootstrap.com/2.0.2/>
18. Python. Урок 17. Виртуальное окружение. [Электронный ресурс] – Режим доступа: <https://devpractice.ru/python-lesson-17-virtual-envs/>

ДОДАТОК А
Текст програми

```

from pathlib import Path
import os

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-
1t%hw%3fuhuq4n!ci_ojrmtn97!^j44j0+)m+#3e)62u)kf7c'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'factory'
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',

```

```

'django.contrib.sessions.middleware.SessionMiddleware',
'django.middleware.common.CommonMiddleware',
'django.middleware.csrf.CsrfViewMiddleware',
'django.contrib.auth.middleware.AuthenticationMiddleware',
'django.contrib.messages.middleware.MessageMiddleware',
'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

```

```

ROOT_URLCONF = 'Project.urls'

```

```

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')]
    },
    {
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

```

```

WSGI_APPLICATION = 'Project.wsgi.application'

```

```
# Database
```

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

```
# Password validation
```

```
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
```

```
    },  
]
```

```
# Internationalization
```

```
LANGUAGE_CODE = 'en-us'
```

```
TIME_ZONE = 'UTC'
```

```
USE_I18N = True
```

```
USE_L10N = True
```

```
USE_TZ = True
```

```
# Static files (CSS, JavaScript, Images)
```

```
STATIC_URL = '/static/'
```

```
# Default primary key field type
```

```
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

ДОДАТОК Б**Текст файлу налаштувань маршрутизації**

""" Project URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:

Examples:

Function views

1. Add an import: `from my_app import views`
2. Add a URL to urlpatterns: `path("", views.home, name='home')`

Class-based views

1. Add an import: `from other_app.views import Home`
2. Add a URL to urlpatterns: `path("", Home.as_view(), name='home')`

Including another URLconf

1. Import the `include()` function: `from django.urls import include, path`
2. Add a URL to urlpatterns: `path('blog/', include('blog.urls'))`

"""

`from django.contrib import admin`

`from django.urls import path`

`urlpatterns = [
 path("", admin.site.urls),`

`]`

ДОДАТОК В
Ілюстрація презентації



Рисунок В.1 – Слайд презентації №1

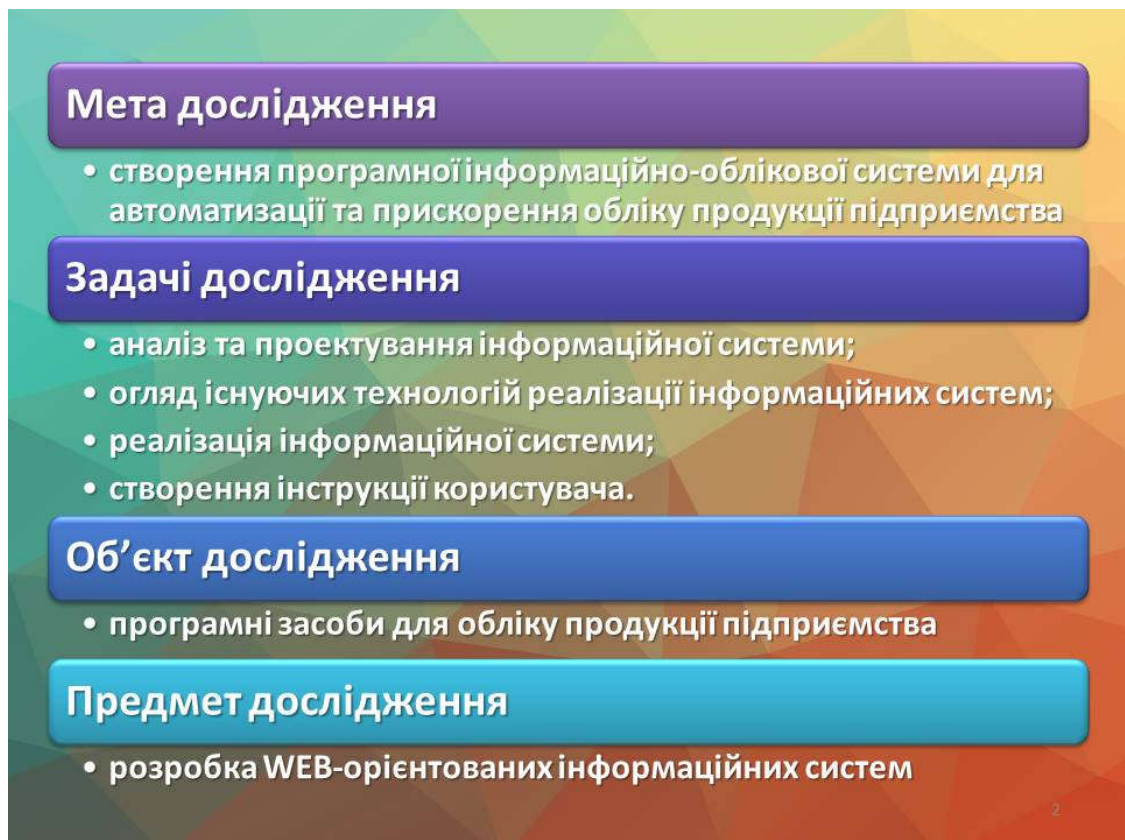


Рисунок В.2 – Слайд презентації №2



Рисунок В.3 – Слайд презентації №3



Рисунок В.4 – Слайд презентації №4



Рисунок В.5 – Слайд презентації №5



Рисунок В.6 – Слайд презентації №6

РЕАЛІЗАЦІЯ ПРОЕКТОВАНИХ МОДЕЛЕЙ

Тип продукту

```
class ProductType(models.Model):
    title = models.CharField('Назва', max_length=100)
    chars = models.TextField('Характеристики', max_length=1000)
    price = models.IntegerField('Вартість')

    def __str__(self):
        return f"{self.title}, {self.price}, {self.chars}"

    class Meta:
        verbose_name = 'Тип продукту'
        verbose_name_plural = 'Типи продуктів'
```

Продукт

```
class Product(models.Model):
    prod_type = models.ForeignKey(ProductType, on_delete=models.CASCADE,
        verbose_name="Тип продукту")

    def __str__(self):
        return f"{self.id} {self.prod_type.title}"

    class Meta:
        verbose_name = 'Продукт'
        verbose_name_plural = 'Продукти'
```

```
class ProductionReport(models.Model):
    date = models.DateTimeField('Дата створення', auto_created=True,
        editable=True, auto_now_add=True)
    products = models.ManyToManyField(Product, verbose_name="Продукти")

    def __str__(self):
        return f"Звіт виготовлення {self.date}"

    class Meta:
        verbose_name = 'Звіт виготовленої продукції'
        verbose_name_plural = 'Звіти виготовленої продукції'
```

Звіт виготовлення продукції

7

Рисунок В.7 – Слайд презентації №7

РЕАЛІЗАЦІЯ ПРОЕКТОВАНИХ МОДЕЛЕЙ

Замовник

```
class Customer(models.Model):
    company = models.CharField('Компанія', max_length=100)
    address = models.CharField('Адреса', max_length=100)
    contacts = models.CharField('Контакти', max_length=200)

    def __str__(self):
        return self.company

    class Meta:
        verbose_name = 'Замовник'
        verbose_name_plural = 'Замовники'
```

Звіт проданої продукції

```
class SellReport(models.Model):
    date = models.DateTimeField('Дата створення', auto_created=True,
        editable=True, auto_now_add=True)
    products = models.ManyToManyField(Product, verbose_name="Продукти")
    customer = models.ForeignKey(Customer, on_delete=models.CASCADE)
    summary = models.IntegerField('Сума прибутку', editable=True, default=0)

    def __str__(self):
        return f"Звіт продажу {self.date}"

    def save(self, *args, **kwargs):
        if self.id is None:
            super(SellReport, self).save(*args, **kwargs)
        for a in self.products.all():
            self.summary += a.prod_type.price
        super(SellReport, self).save(*args, **kwargs)

    class Meta:
        verbose_name = 'Звіт проданої продукції'
        verbose_name_plural = 'Звіти проданої продукції'
```

8

Рисунок В.8 – Слайд презентації №8

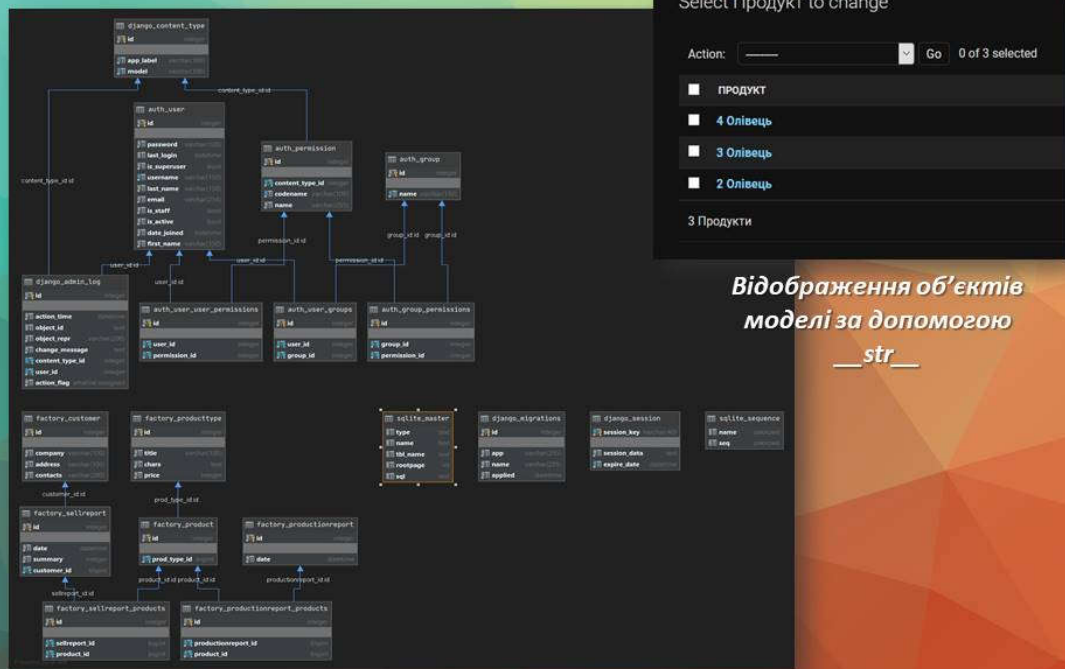


Рисунок В.9 – Слайд презентації №9

РЕАЛІЗАЦІЯ КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ

```
from django.contrib import admin
from .models import *

@admin.register(Product)
class ProductAdmin(admin.ModelAdmin):
    pass

@admin.register(ProductType)
class ProductTypeAdmin(admin.ModelAdmin):
    pass

@admin.register(ProductionReport)
class ProductReportAdmin(admin.ModelAdmin):
    pass

@admin.register(SellReport)
class SellReportAdmin(admin.ModelAdmin):
    pass

@admin.register(Customer)
class CustomerAdmin(admin.ModelAdmin):
    pass
```

```
@admin.register(Customer)
class CustomerAdmin(admin.ModelAdmin):
    pass

    {list_filter =
('status', 'due_back')}
    fieldsets = (
        (None, {
            'fields': ('book', 'imprint', 'id')    }
        ),
        ('Availability', {
            'fields': ('status', 'due_back')
        }),
    )
```

Вигляд файлу admin.ru

Загальний вигляд адміністративної панелі

Рисунок В.10 – Слайд презентації №10

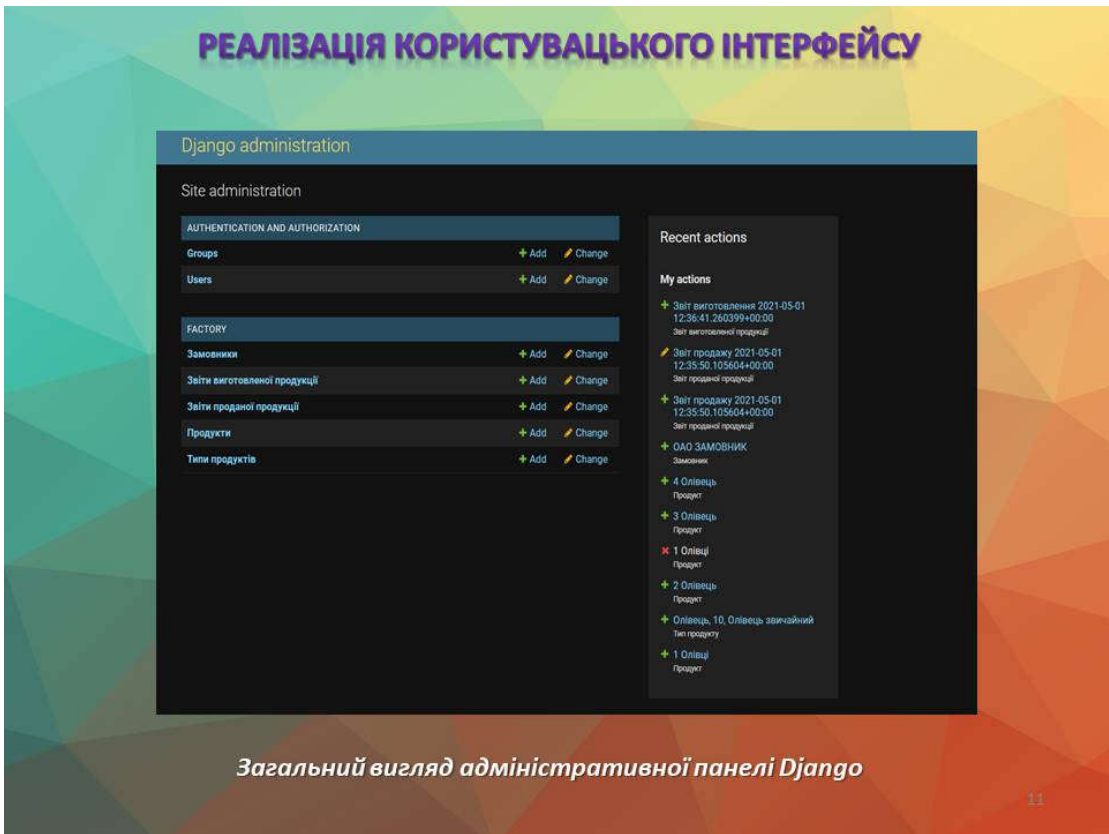


Рисунок В.11 – Слайд презентації №11

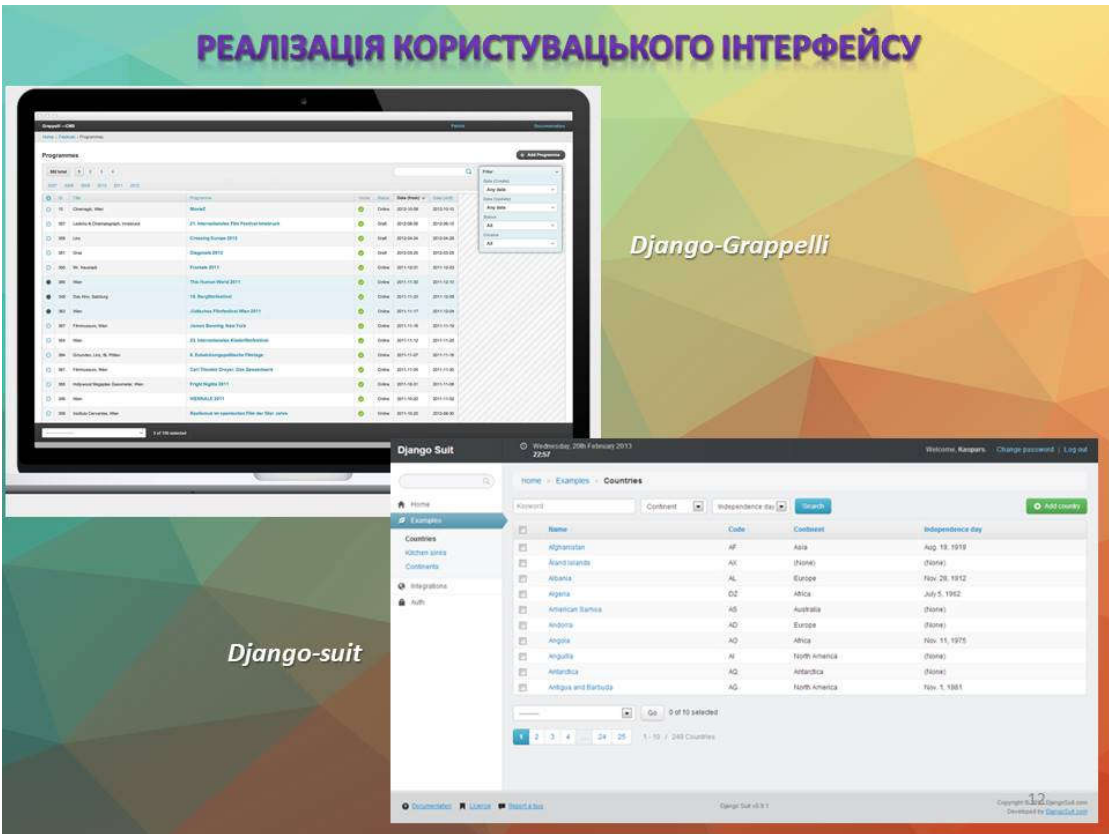


Рисунок В.12 – Слайд презентації №12

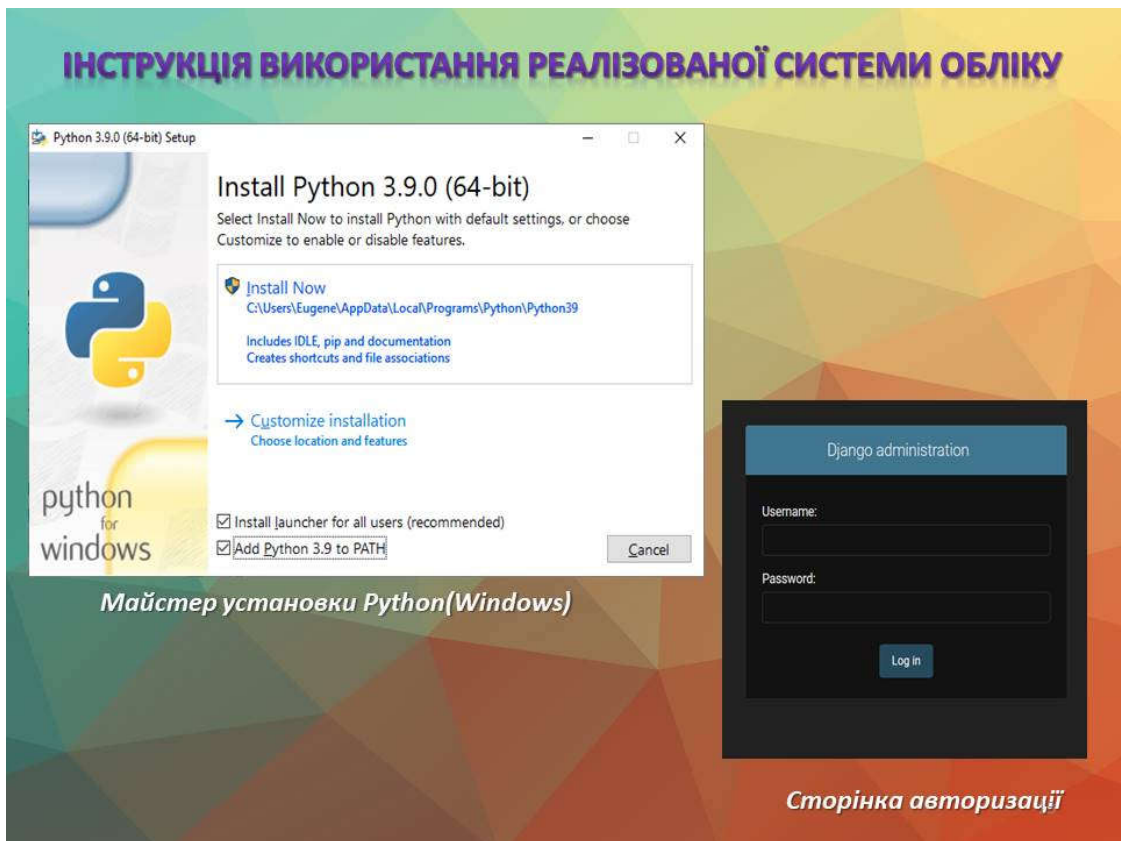


Рисунок В.13 – Слайд презентації №13

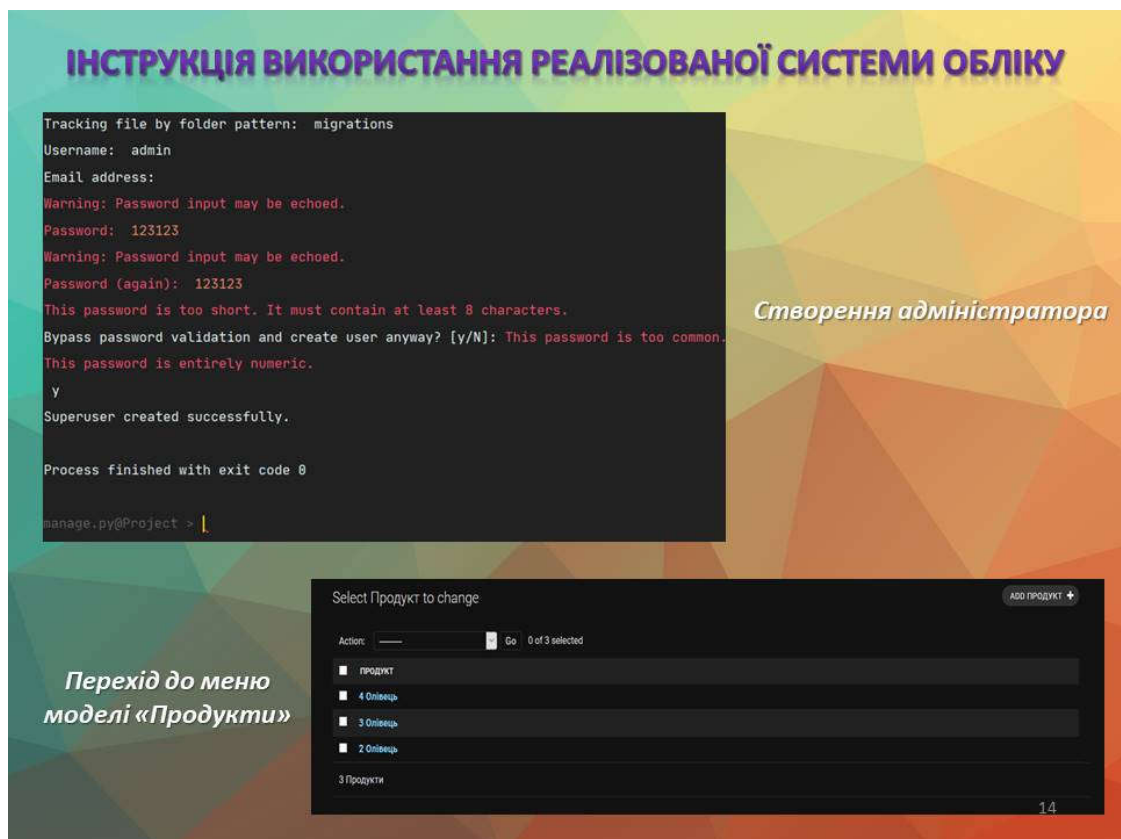


Рисунок В.14 – Слайд презентації №14

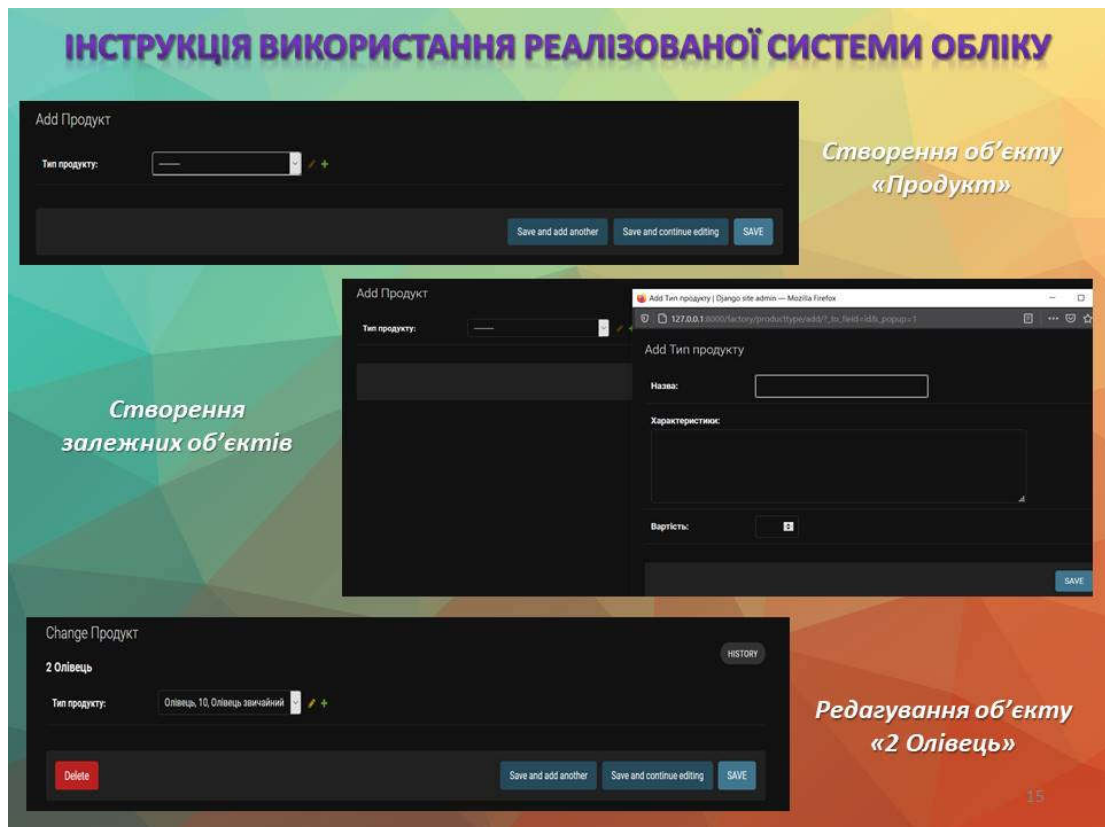


Рисунок В.15 – Слайд презентації №15

ВИСНОВОК

- Таким чином, при виборі системи управління базами даних були досліджені, проаналізовані та виявлені переваги і недоліки таких СУБД як MySQL, MS SQL Server, PostgreSQL, SQLite. Як використовуваної СУБД вибір зроблений на користь MySQL, а в якості мови програмування для розробки програми доступу до даних обраний Python, фреймворк Django. Це пов'язано з тим, що MySQL дозволяє ефективно обробляти великий обсяг даних.
- При розробці інформаційної системи, був використаний для користувача інтерфейс за стандартом GUI. Інформаційна система створена для багатьох користувачів. Її користувачами бухгалтери, які спочатку здійснюють введення обліку продукції підприємства, яке виготовляє канцелярські вироби.
- Інформаційна система є відкритою і має великий потенціал свого розвитку. Вона може змінюватися при зміні вимог, що пред'являються до предметної області.
- Розроблена інформаційна система не претендує на комерційний повноцінний додаток, а створена у якості наукового дослідження. Розроблена система підлягає модифікації, видозміні та дозволяється до використання у подальших розробках та дослідженнях.

Рисунок В.16 – Слайд презентації №16