

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Львівський національний університет імені Івана Франка

Факультет електроніки та комп'ютерних технологій

Кафедра радіоелектронних і комп'ютерних систем

КУРСОВА РОБОТА

з курсу “Бази даних та знань”

на тему

**Створення веб додатку для управління складом магазину одягу
використовуючи базу даних MySQL**

Виконав:

студент групи ФеС-22

спеціальності:

126 - Інформаційні системи і технології

_____ **Данило ЦЕБРО**

Науковий керівник:

_____ **доц. Ярослав БОЙКО**

«___» _____ 2023 р.

Львів 2023

ЗМІСТ	
АНОТАЦІЯ	2
ВСТУП	3
1. ЗАГАЛЬНІ ВІДОМОСТІ Й ТЕРМІНИ	4
1.1 Веб-додатки та їх особливості.....	4
1.2 Різниця між веб-додатком та веб-сайтом	4
1.3 Кваліфікація за технічною ознакою та призначенням	5
2. ВИКОРИСТАНІ ТЕХНОЛОГІЇ ТА ЗАСОБИ ПРИ РОЗРОБЦІ ПРОЄКТУ	7
2.1 Мова програмування Python	7
2.2 Фреймворк Django.....	9
2.3 Bootstrap.....	12
2.4 Система контролю версій Git.....	13
2.5 Середовище розробки Visual Studio Code	13
2.6 Система управлінням БД MySQL	15
3. ПРОЦЕС РОЗРОБКИ ВЕБ-ДОДАТКУ	18
3.1 Створення проєкту у Django.....	18
3.2 Створення візуальної частини веб-додатку.....	19
3.3 Програмування логіки веб-додатку	22
3.4 Впровадження базового захисту у веб-додаток	24
ВИСНОВОК	26
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	27
ДОДАТКИ	28

АНОТАЦІЯ

Дана курсова робота є прикладом створення веб-додатку, який надає користувачу можливість керувати складом магазину одягу через веб-додаток. Робота включає в себе розробку бази даних та розробку веб-додатку на мові програмування Python та використання фреймворку Django, що інтегрується у систему MySQL для зберігання, редагування, видалення та перегляду товарів і замовлень, зберігання інформації про працівників. Інтерфейс веб-додатку включає в себе зручний доступ до таких категорій як: «Працівники», «Товар» та «Замовлення», також інтерфейс містить профіль користувача та вікно реєстрації. Розроблений інтерфейс дозволяє не лише переглядати, а й редагувати інформації за умови наявності такого рівня доступу у користувача.

Висновок включає в себе оцінку ефективності веб-додатку та окреслює можливості щодо поліпшення продукту. Анотація в свою чергу містить короткий обрис головних аспектів роботи, визначаючи її внесок у галузь розробки та управління базами даних.

Abstract

This course work is an example of creating web application that allows users to manage the inventory of clothing store through web interface. The work includes development of database and a web interface for web application using the Python programming language and Django framework, which integrates with MySQL system for storing, editing, deleting, reviewing products, orders, as well as storing information about employees.

The web application interface provides convenient access to categories such as: “Employees”, “Products”, “Orders”. It also includes a user profile and registration form. The developed interface allows not only viewing but also editing information, provided that the user has appropriate level of access. The conclusion evaluates the effectiveness of the web application and outlines suggestions and opportunities for improving the product. The annotation provides a overview of the main aspects of the work, defining its contribution to the field of development of database.

ВСТУП

Сучасне життя у світі високих технологій важко уявити без різноманітних веб-додатків, які покращують та поліпшують найрізноманітніші аспекти нашого життя. Звісно, використання веб-додатків не могло не торкнутись й полегшення бізнес процесів.

Одними з таких галузей є торгівля та управління складами. Веб-додатки дуже сильно збільшили ефективність й простоту роботи працівників різноманітних складів, зокрема й магазинів одягу, з-за допомогою веб-додатків можна максимально просто вести облік товарів і замовлень з будь-якої точки планети.

Ця курсова робота спрямована на створення веб-додатку для керування складом магазину одягу, який має на меті полегшити процеси управління товарними запасами та продажами в магазинах одягу. Застосування сучасних технологій дозволяє покращити швидкість обробки замовлень, вести облік наявних товарів та надає зручний інтерфейс для користувачів.

В даній курсовій роботі ми розглянемо важливі моменти планування й створення веб-додатку для керування складом магазину одягу такі як: необхідні функції веб-додатку, набір технологій, який буде використовуватись, дизайн та тестування.

Також я приділю необхідну увагу питанням безпеки, а саме: обмеження прав на читання даних й перегляду певних сторінок веб-застосунку, яка підвищить захист , обов'язкове вікно реєстрації для нових користувачів та обов'язковий вхід після виходу з сайту, також необхідна увага приділяється правам на редагування, видалення та додавання нових записів даних.

Мета курсової роботи полягає у вирішенні наявних проблем у логістиці, надаючи необхідні інструменти для оптимізації управління складом та покращення взаємодії з клієнтами.

1. ЗАГАЛЬНІ ВІДОМОСТІ Й ТЕРМІНИ

1.1 Веб-додатки та їх особливості

Веб-додаток – це програмне забезпечення або програма, яка може бути відкрита у будь-якому браузері, складається з двох частин: фронт-енд та бек-енд.

Особливості веб-додатку:

- **Кросплатформеність** – на відміну від програмного забезпечення, яке розробляється для комп'ютерів веб-додаток працює на будь-якій операційній системі (MacOS, GNU/Linux, Windows), основною вимогою є підтримка сучасних браузерів, найчастіше це Google Chrome.
- **Велика кількість технологій** – з огляду на те що веб-додатки є одними з найпопулярніших видів програмного забезпечення – для них існує велика кількість різноманітних бібліотек, фреймворків, веб-додатки можна писати майже на будь-якій мові програмування, наприклад JavaScript, Python, Ruby, Golang, тощо.
- **Універсальність** – веб-додатки створюються для вирішення великої кількості проблем бізнесу і простих користувачів,

1.2 Різниця між веб-додатком та веб-сайтом

- **Основне призначення** – основне призначення веб-додатку це розробка програмного забезпечення з яким користувач може взаємодіяти, в свою чергу основне призначення веб-сайту це перегляд вмісту без можливості впливу на даний вміст користувачем.
- **Взаємодія з користувачем** – користувач веб-додатку може проводити маніпуляції з даними такі як читання, редагування, видалення і створення зважаючи на права доступу цього користувача. Користувач веб-сайту може лише переглядати дані, які розміщені на даному веб-сайті без можливості маніпуляції над цими даними.

- **Аутентифікація** – більшість веб-додатків вимагають від користувача аутентифікації, через те що різні користувачі мають різний рівень доступу до веб-додатку, і на основі цього рівня доступу мають різні можливості для роботи з даними. В свою чергу більшість веб-сайтів вимагає тільки пароль, що може надаватись адміністратором веб-сайту
- **Складність** – веб-додатки розробляються з-за допомогою використання комплексу різних технологій для вирішування певних проблем, а веб-сайт це статична сторінка з текстовою інформацією на ній. Відповідно вносити й додавати новий функціонал у веб-додаток набагато складніше, ніж у веб-сайт.

1.3 Кваліфікація за технічною ознакою та призначенням

Веб-додатки поділяються на різні види за своїми технічними ознаками та призначенням.

Кваліфікація за технічною ознакою:

- **SPA(Single Page Application)** - одно сторінковий веб-додаток. Користувачі користуючись додатком перемикаються між вкладками залишаючись на одній сторінці. При такій роботі веб-додатку оновлюються та завантажуються лише ті частини веб-додатку, якими на даний момент використовує користувач, це дуже сильно впливає на швидкість роботи веб-додатку в позитивному плані.

Перевагами такого виду веб-додатку є велика швидкість його роботи, яка досягається завдяки тому, що вміст, який не змінюється не потрібно перезавантажувати. Найпопулярнішим прикладом SPA-додатку є Gmail.

Недоліками такого формату виступає вразливість до XSS(міжсайтовий скрипт) атак зловмисників, які полягають у додаванні шкідливих скриптів на одну з сторінок веб-додатку, які виконуються коли користувач заходить на сторінку з таким скриптом.

- **MPA(Multi Page Application)** – найбільш поширений вид веб-додатків, відрізняється тим, що користувач під час користування веб-додатком переходить на різні HTTP-сторінки, через що обмін даними відбувається повільніше, ніж у SPA-додатках. Найбільш популярними прикладами таких додатків є інтернет-магазини, такі як: Rozetka, Amazon.

Кваліфікація за призначенням:

- **E-commerce системи** – дані системи розробляються для того, щоб користувачі могли замовляти і продавати товар без посередників, тобто у ланцюжку продажу лише дві особи: клієнт і продавець. Найбільш популярні e-commerce системи це різноманітні інтернет-магазини, маркет-плейси, онлайн-каталоги.
- **CRM системи** – такі веб-додатки розробляються для автоматизації продажу, обробки заявок або замовлень, за рахунок такої системи можна відслідковувати кількість замовлень, товару, історію продажів, тощо.
- **ERP системи** – такі системи автоматизують не лише систему продажу товару а і роботу всієї компанії, за допомогою такої системи можна надавати завдання різним підрозділам компанії, бачити різноманітну статистику.
- **Портали** – це веб-додатки, які виступають площадками для комунікації між людьми. Портали існують як на вільні теми так і специфічні, або професійні портали де люди спілкуються на професійну тему, шукають роботу і так далі. Прикладом порталів є Reddit, LinkedIn.

2. ВИКОРИСТАНІ ТЕХНОЛОГІЇ ТА ЗАСОБИ ПРИ РОЗРОБЦІ ПРОЄКТУ

2.1 Мова програмування Python

Для створення веб-додатків використовується велика кількість різноманітних мов програмування, найпопулярнішими мовами для розробки веб-додатків є – JavaScript, Python, PHP, усі ці мови мають велику кількість бібліотек та фреймворків для веб-розробки, серед всіх цих мов я хотів би виділити мову програмування Python.

Python був створений у 1990 році талановитим програмістом Гвідо ван Россум з Голландії, як інтерпретована об'єктно-орієнтована мова програмування високого рівня із суворою динамічною типізацією. Великими плюсами мови програмування Python є підтримка кількох парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна, аспектно-орієнтована. Та універсальність цієї мови, Python використовують не тільки у веб-розробці, а у машинному навчанні, програмуванні нативного комп'ютерного програмного забезпечення, розв'язанні математичних задач та аналізі даних.

Мова програмування Python була обрана мною через певний ряд причин, а саме:

1. **Велика спільнота** – довга історія існування та популярність мови програмування Python збрала навколо себе велику кількість розробників, що призвело до створення великої кількості матеріалів для вивчення цієї мови, створення великої кількості бібліотек та фреймворків, які полегшують розробку рішень для найрізноманітніших проблем, зокрема й у веб-розробці.
2. **Інтерпритованість** – код у мові програмування Python виконується рядок за рядком, що полегшує розробку та тестування розроблених рішень.
3. **Об'єктно-орієнтованість** – мова програмування Python підтримує об'єктно-орієнтований підхід, що надає можливість формувати код у

класи та їх об'єкти, це буде використано у подальшому для проектування і організації бази даних у моєму веб-додатку.

4. **Підтримка пакетів і модулів** – мова програмування Python підтримує пакети й модулі, як вбудовані так і розроблені іншими людьми, що значно пришвидшує розробку та допомагає організувати код по різним модулям, що значно покращує розробку та робить архітектуру проекту більш організованою.
5. **Синтаксис** – синтаксис у мові програмування Python максимально простий і привітливий для розробника, що підвищує швидкість розробки продукту.
6. **Швидкість розробки** – один з найбільших плюсів мови програмування Python є швидкість розробки, простий синтаксис, велика кількість модулів і бібліотек, інтерпритованість цієї мови надає дуже швидкий і приємний досвід розробки.





Рис 2.1.2 Логотип мови програмування Python

2.2 Фреймворк Django

Django – високорівневий відкритий фреймворк для мови програмування Python, який використовується для розробки веб-додатків. Веб-додаток на Django складається з однієї декількох частин, які для коректної роботи фреймворку рекомендується робити модульними.

Особливості фреймворку Django:

1. **Вбудована панель для адміністрування сайту** – Django має вбудований модуль, який містить в собі веб-сторінку для адміністрування веб-додатком, де можна створювати користувачів, міняти їм права доступу, об'єднувати їх у групи, також додавати дані, переглядати останні дії, які були виконані на сайті.

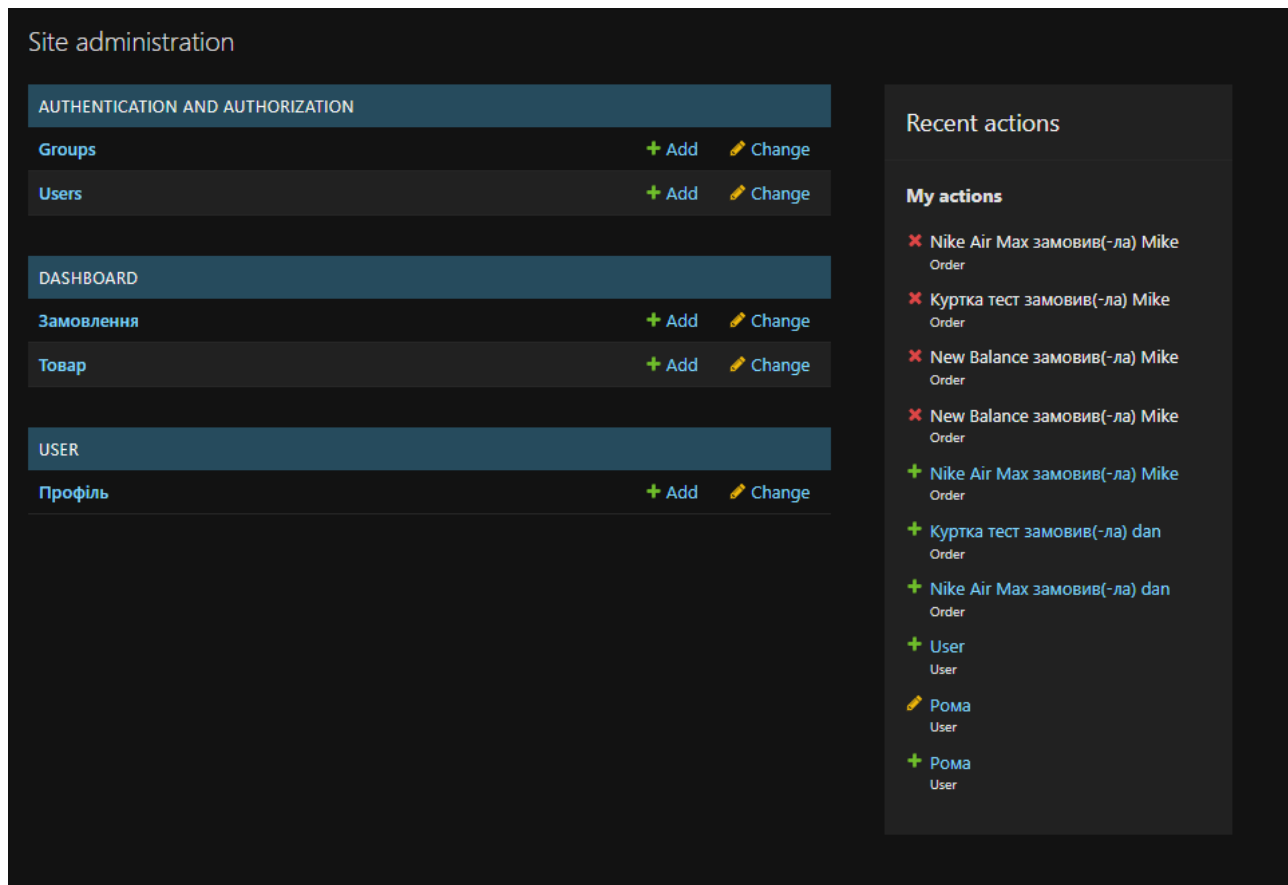


Рис 2.2.1 Вбудована у Django веб-сторінка для адміністрування веб-додатком.

2. **Модульність** – незважаючи на те що модульність проекту є лише рекомендацією для коректної роботи, в більшості всіх проектів написаних на Django є модульними, що значно полегшує орієнтацію в коді проекту, та пришвидшує час розробки.
3. **Об'єктно-орієнтованість бази даних** – об'єкти бази даних в Django називаються «моделями» і прописуються за принципом об'єктно-орієнтованого програмування, що органічно організовує дані у «моделі».

```
class Product(models.Model):
    name = models.CharField(max_length=100, null=True)
    category = models.CharField(max_length=50, null=True, choices=CATEGORY)
    quantity = models.PositiveIntegerField(null=True)

    class Meta:
        verbose_name_plural = 'Товар'

    def __str__(self):
        return f'{self.name}'
```

Рис 2.2.2 Приклад «моделі» написаною на мові програмування Python використовуючи фреймворк Django.

Архітектура проекту на фреймворку Django:

Архітектура проекту на Django складається з декількох модулів, а саме:

1. **Модуль “Project”** – модуль проекту містить в собі файли кешу, файл що містить в собі налаштування, такі як: база даних, що використовується, список зареєстрованих «додатків», та файл з усіма URL-посиланнями на веб-сторінки у моєму веб-додатку.

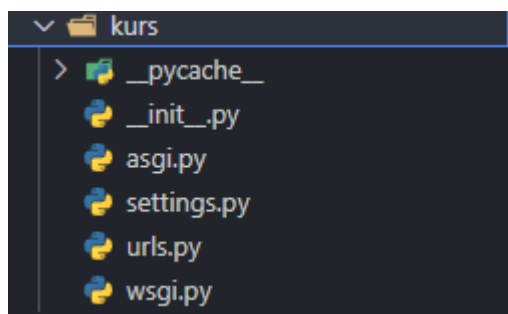


Рис 2.2.3 Модуль «Project» у моєму проєкті.

2. **Модуль «application»** - модуль проекту з якого складається веб-додаток, у одному веб-додатку може міститись декілька таких модулів, в собі містять файли, що визначають логіку сторінки, «моделі» що використовуються для роботи з даними, файл де прописуються тести, та усі посилання, що використовуються цим модулем.

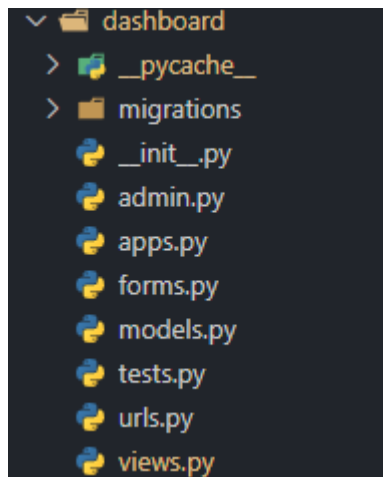


Рис 2.2.4 Модуль “application” у моєму проєкті.

3. **Модуль «template»** - модуль, що містить в собі шаблони та html-код сторінок у веб-додатку.
4. **Файл manage.py** – файл через, який здійснюється керування проєктом, а саме: створення супер-користувачів(користувачі, що мають усі права доступу), звичайних користувачів, команда запуску локального серверу, робить міграцію даних з «моделей» у базу даних, що вказана в налаштуваннях, за стандартом це SQLite.
5. **Файл db.sqlite3** – файл з базою даних проєкту, що створюється по замовчуванню, якщо змінити в налаштуваннях проєкту систему бази даних, її буде потрібно прив’язати до вже існуючої на сервері, в такому випадку новий файл не створюється.

2.3 Bootstrap

Bootstrap – це безкоштовний набір інструментів з відкритим кодом. Основним призначенням якого є допомога у створенні веб-сайтів та веб-додатків, Bootstrap містить набір шаблонів для CSS і HTML для кнопок, форм, навігації, спливаючих меню та інших компонентів інтерфейсу веб-сайту. Bootstrap – клієнтський фреймворк, тобто користувач напряму взаємодіє з його елементами, на відміну від фреймворків серверної частини, про існування яких користувач навіть може не здогадуватись. Репозиторій Bootstrap є одним з найпопулярніших на GitHub,

окрім звичайних веб-розробників, його також використовують у NASA та MSNBC.

2.4 Система контролю версій Git

Git – це розподілена система контролю версіями файлів, також використовується для спільної роботи. Лінус Торвальдс створив Git для керування розробкою ядра Linux, а сьогодні цей проект підтримує Джуніо Хамано.

Git є найефективнішою та надійною системою керування версіями, що надає засоби нелінійної розробки, що базуються на методі створення і злиття гілок. У системі Git є можливість прив'язки цифрових підписів розробників, тегів та комітів. Віддалений доступ до репозиторіїв Git здійснюється з-за допомогою git-демонів, SSH або HTTP сервером.

Git-демон та SSH сервер є одними з найпопулярніших та найнадійніших методів доступу, незважаючи на різноманітні обмеження HTTP також є популярним методом отримання віддаленого доступу до репозиторіїв Git.

Я обрав Git як систему контролю версій, через те що вона надає можливість створення віддалених репозиторіїв у GitHub, що як зрозуміло виходячи з назви базується на системі Git, це надало мені можливість зберігати різні версії в моєму віддаленому репозиторії без створення великої кількості непотрібних копій на моєму комп'ютері. Це надало мені можливість тестувати й переробляти різні версії мого веб-додатку без втручання й внесення змін у основну, робочу версію застосунку.

2.5 Середовище розробки Visual Studio Code

Visual Studio Code – одне з найпопулярніших інтегрованих середовищ розробки для різних мов програмування таких як: JavaScript, Python, C++, Fortran, C#, Go, Java, Rust. Visual Studio Code або як його ще називають VS Code розроблений

компанією Microsoft спільно з Electron Framework, для редагування й написання коду на всіх популярних операційних системах.

Особливості VS Code:

- 1. Відкритий код** – код Visual Studio Code знаходиться у відкритому доступі у репозиторії GitHub, що надає можливість великій базі користувачів переглядати код інтегрованого середовища розробки та редагувати його під свої потреби, наприклад змінюючи інтерфейс, зміну комбінацій клавіш, теми, зміну параметрів та надає можливість розробляти розширення, які значно покращують досвід розробки.
- 2. Розширення** – через відкритий код VS Code у спільноти розробників, що використовують це середовище розробки є можливість створювати різноманітні розширення, що поліпшують ефективність розробки, роблять її швидшою та приємнішою, наприклад під час виконання даної курсової розробки я використовував розширення Live Server, що надавало мені можливість переглядати як виглядає мій веб-застосунок в реальному часі. Також я використав розширення Pylance, що додало інтелектуальне завершення коду та підсвічування синтаксису для мови програмування Python.
- 3. Вбудована система контролю версій Git** – VS Code має завчасно вбудовану систему контролю версій Git та надає графічний інтерфейс для використання цієї системи, що значно полегшує роботу з різними версіями проєкту, що дозволяє максимально швидко керувати версіями веб-додатку.
- 4. Інтелектуальний редактор коду** – Visual Studio Code надає ефективний редактор коду, який підсвічує синтаксис, що вказує на помилки у коді, пояснює як вони виникли та пропонує шляхи їх виправлення, також підсвітка синтаксису допомагає у орієнтації в коді. Також редактор коду має функції інтелектуального завершення коду, тобто під час написання коду VS Code показує можливі варіанти того, що ви хотіли б написати, таке

авто завершення коду значно підвищує процес написання коду й розробки веб-додатку.

- 5. Можливість працювати з веб-додатку** – VS Code надає можливість розробникам працювати з їхнього браузера, тобто задля початку роботи з цим середовищем розробки не обов'язково навіть встановлювати програму на ваш комп'ютер, а потрібно лише стабільне підключення до мережі «Інтернет» та браузер, що підтримує роботу у Visual Studio Code, наприклад Google Chrome або Firefox.
- 6. Вбудований термінал** – У Visual Studio Code є можливість використання вбудованого терміналу, ви можете вибрати термінал на основі тих, які є встановленими у вашій операційній системі, в моєму випадку це термінали: PowerShell, bash та звичайний командний рядок Windows.

2.6 Система управління БД MySQL

MySQL це система управління базами даних великих та малих масштабів, що є важливою складовою інформаційних технологій, що дозволяє керувати й обробляти різні бази даних. MySQL має графічний інтерфейс MySQL Workbench, що дозволяє зручно й ефективно створювати бази даних, керувати ними та обробляти дані, що утримуються в цих базах даних.

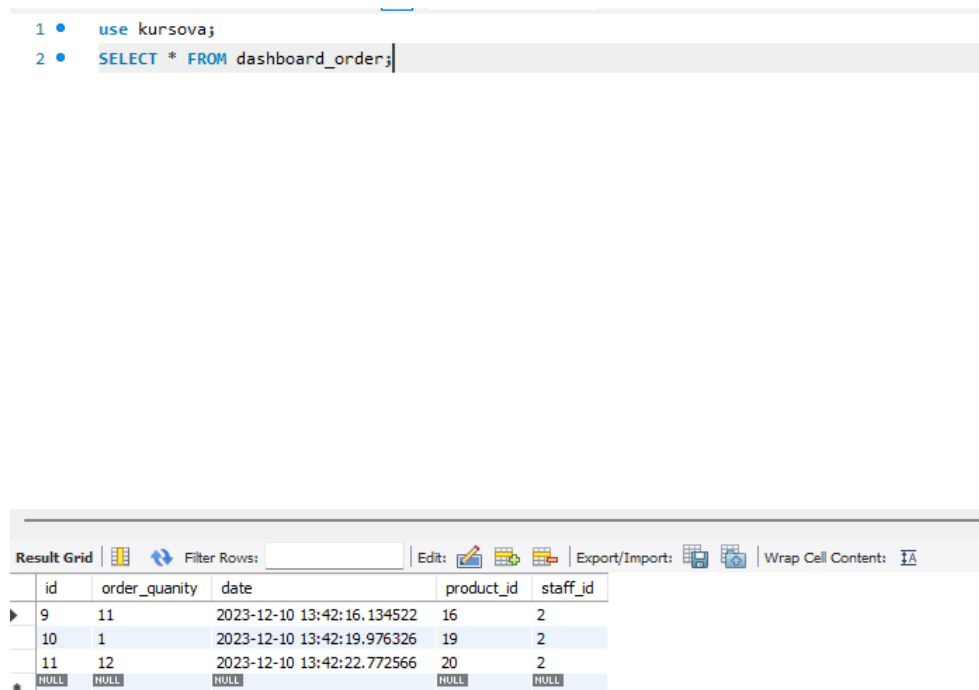


Рис 2.6.1 Інтерфейс робочого вікна у MySQL Workbench

Також MySQL Workbench надає можливість створювати діаграми для баз даних.

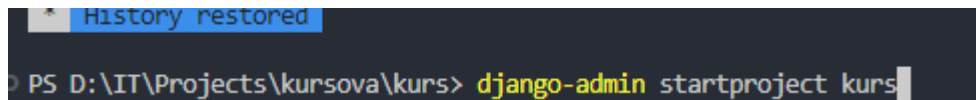
Ці діаграми відображають існуючі таблиці у вибраній базі даних, зв'язки між цими таблицями та типи цього зв'язку, також на діаграмі видно поля у таблицях, типи даних цих полів та кількість допустимих символів, що може набувати значення цього поля, ще ми можемо побачити чи є поле ключем та тип ключа: Primary або ж Foreign Key.

3. ПРОЦЕС РОЗРОБКИ ВЕБ-ДОДАТКУ

3.1 Створення проєкту у Django

Керування проєктом в Django здійснюється з-за допомогою файлу `manage.py`, що містить в собі команди для створення проєктів та застосунків в середині нього, створення користувачів та налаштування цих користувачів так і самого проєкту.

Проєкт в Django створюється з-за допомогою простої команди в терміналі: `django-admin startproject «назва проєкту»`.

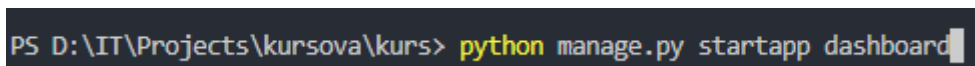


```
PS D:\IT\Projects\kursova\kurs> django-admin startproject kurs
```

Рис 3.1.1 Створення проєкту курсової в Django

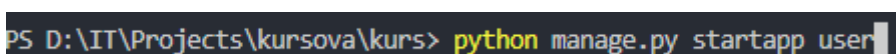
Після цього у нас утвориться папка з проєктом, яка містить файли з основними налаштуваннями всього проєкту, а саме такі файли як налаштування, що містить інформації про базу даних що використовується, зареєстровані застосунки у цьому проєкті, посилання, та інше.

Також я створив необхідні застосунки з яких складається мій проєкт, а таких було саме два, сам застосунок `dashboard` та проєкт, що містить логіку й сторінки, що необхідні для процесу реєстрації та входу користувача у систему, ці проєкти створюються такими командами: `python manage.py startapp dashboard` та `python manage.py startapp user`



```
PS D:\IT\Projects\kursova\kurs> python manage.py startapp dashboard
```

Рис 3.1.2 Команда створення застосунку `dashboard`



```
PS D:\IT\Projects\kursova\kurs> python manage.py startapp user
```

Рис 3.1.3 Команда створення застосунку `user`

Після створення архітектури проєкту необхідно налаштувати його та додати базу даних MySQL, з огляду на те що по замовчуванню використовується система керування базами даних SQLite3, це робиться в файлі settings.py у папці проєкту

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'kursova',
        'USER': 'root',
        'PASSWORD': 'admin',
        'HOST': 'localhost',
        'PORT': '3306',
    }
}
```

Рис 3.1.4 Зміна системи управління базами даних на MySQL у файлі settings.py

3.2 Створення візуальної частини веб-додатку

Після створення самого проєкту та його налаштування наступним кроком є створення візуальної клієнтської частини проєкту, а саме написання HTML-коду з використанням шаблонів Bootstrap. Для того, щоб Django розпізнавав ці сторінки, як ті що потрібно використати, необхідно створити папку templates:

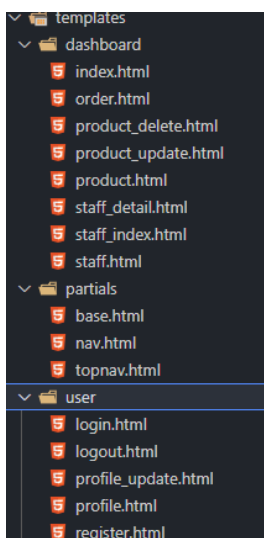


Рис 3.2.1 Папка templates, що містить всі html сторінки проєкту.

Та зареєструвати їхні посилання у папці проєкту та папках відповідних застосунків

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('dashboard.urls')),
    path('register/', user_view.register, name='user-register'),
    path('profile/', user_view.profile, name='user-profile'),
    path('profile/update/', user_view.profile_update, name='user-profile-update'),
    path('', auth_views.LoginView.as_view(template_name='user/login.html'), name='user-login'),
    path('logout/', auth_views.LogoutView.as_view(template_name='user/logout.html'), name='user-logout'),
]
```

Рис 3.2.2 Реєстрація URL-посилань у налаштуваннях проєкту

```
urlpatterns = [
    path('dashboard/', views.index, name='dashboard-index'),
    path('staff/', views.staff, name='dashboard-staff'),
    path('staff/detail/<int:pk>', views.staff_detail, name='dashboard-staff-detail'),
    path('product/', views.product, name='dashboard-product'),
    path('product/update/<int:pk>', views.product_update, name='dashboard-product-update'),
    path('product/delete/<int:pk>', views.product_delete, name='dashboard-product-delete'),
    path('order/', views.order, name='dashboard-order'),
]
```

Рис 3.2.3 Реєстрація URL-посилань у налаштуваннях застосунку dashboard

Після того як ми створили візуальну частину та зареєстрували її посилання в налаштуваннях ми можемо переглянути результат з-за допомогою команди `python manage.py runserver`, що дає можливість запустити локальний сервер з-за допомогою фреймворку Django, який дає можливість переглядати всі зміни у вашому проєкті у реальному часі.

```
PS D:\IT\Projects\kursova\kurs>
December 16, 2023 - 13:56:06
Django version 4.2.7, using settings 'kurs.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Рис 3.2.4 Успішний запуск локального сервера

Після того як ми перейдемо по вказаному у терміналі посиланні ми побачимо індекс-сторінку, або ж сторінку на яку попадає користувач, який використовує мій веб-додаток.

Індекс-сторінкою може бути будь-яка сторінка, яка встановлена такою, для збільшення рівня захисту даних у даній курсовій роботі я вирішив встановити сторінку авторизації як індекс-сторінку.

Реєстрація Вхід

Вхід

Username:

Password:

Вхід

Рис 3.2.5 Індекс-сторінка веб-додатку.

Також мій веб-додаток містить різноманітні сторінки для перегляду різних категорій та даних, таких як «Товар», «Працівники» та «Замовлення»

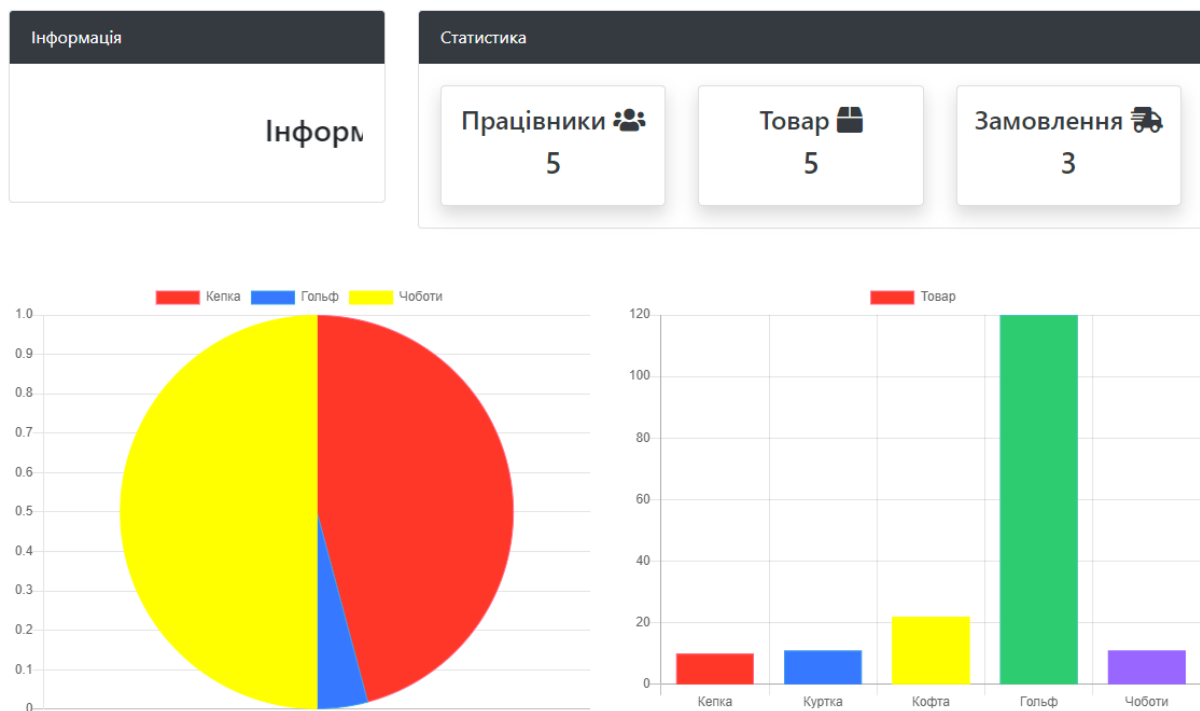


Рис 3.2.6 Сторінка користувача з права адміністратора.

Якщо авторизуватись як користувач з правами адміністратора ми зможемо переглядати повну статистику товарів і замовлень на графіках та перейти на сторінки з працівниками, товаром та замовленнями.

Статистика

Працівники 👤
5

Товар 📦
5

Замовлення 🚚
3

#	Ім'я	Ел. Пошта	Телефон
<div>Переглянути</div>	dan	dan1@gmail.com	1231321
<div>Переглянути</div>	Mike	mikestaff@gmail.com	593816
<div>Переглянути</div>	Рома	roma@gmail.com	None
<div>Переглянути</div>	User		None
<div>Переглянути</div>	Bob	bobloh@gmail.com	None

Статистика

Працівники 👤
5

Товар 📦
5

Замовлення 🚚
3

ID	Назва	Категорія	Кількість	Дії
16	Кепка	Головний убір	10	<div>РедагуватиВидалити</div>
17	Куртка	Верхній одяг	11	<div>РедагуватиВидалити</div>
18	Кофта	Верхній одяг	22	<div>РедагуватиВидалити</div>
19	Гольф	Верхній одяг	120	<div>РедагуватиВидалити</div>
20	Чоботи	Взуття	11	<div>РедагуватиВидалити</div>

Рис 3.2.7 Сторінки з працівниками та товаром

Статистика				
Працівники 👤 5		Товар 📦 5		Замовлення 🚚 3
Товар	Категорія	Кількість	Оформив Замовлення	Дата Замовлення
Кепка	Головний убір	11	Mike	Dec. 10, 2023, 1:42 p.m.
Гольф	Верхній одяг	1	Mike	Dec. 10, 2023, 1:42 p.m.
Чоботи	Взуття	12	Mike	Dec. 10, 2023, 1:42 p.m.

Рис 3.2.8 Сторінка з замовленнями

3.3 Програмування логіки веб-додатку

Після створення візуальної частини, що складалась лише з HTML-коду необхідно імплементувати логіку веб-додатку, а саме написання «моделей» для заповнення й організації баз даних та написання функцій, які використовуються у курсовій роботі.

Усі моделі й логіка прописуються у папках відповідних застосунків, зокрема в папці dashboard, я прописав моделі для замовлень та товарів і створив категорії цих товарів

```

CATEGORY = (
    ('Верхній одяг', 'Верхній одяг'),
    ('Взуття', 'Взуття'),
    ('Штани', 'Штани'),
    ('Куртка', 'Куртка'),
    ('Головний убір', 'Головний убір'),
)

class Product(models.Model):
    name = models.CharField(max_length=100, null=True)
    category = models.CharField(max_length=50, null=True, choices=CATEGORY)
    quantity = models.PositiveIntegerField(null=True)

    class Meta:
        verbose_name_plural = 'Товар'

    def __str__(self):
        return f'{self.name}'

```

Рис 3.3.1 Код для моделі товару й категорій.

```

class Order(models.Model):
    product = models.ForeignKey(Product, on_delete=models.CASCADE, null=True)
    staff = models.ForeignKey(User, models.CASCADE, null=True)
    order_quantity = models.PositiveIntegerField(null=True)
    date = models.DateTimeField(auto_now_add=True)

    class Meta:
        verbose_name_plural = 'Замовлення'

    def __str__(self):
        return f'{self.product} замовив(-ла) {self.staff.username}'

```

Рис 3.3.2 Код для створення моделі замовлень.

Також я виконав функції для роботи з цими моделями та даними, що вони зберігають, ці функції містять в собі додавати, видаляти й редагувати дані в категоріях товару, працівників та замовлень, також відображають зміни у кількості товару й замовлень в реальному часі.

```

@login_required(login_url='user-login')
def staff(request):
    workers = User.objects.all()
    workers_count = workers.count()
    order_count = Order.objects.all().count()
    items_count = Product.objects.all().count()
    context = {
        'workers': workers,
        'workers_count': workers_count,
        'order_count': order_count,
        'items_count': items_count
    }
    return render(request, 'dashboard/staff.html', context)

@login_required(login_url='user-login')
def staff_detail(request, pk):
    workers = User.objects.get(id = pk)
    context = {
        'workers': workers,
    }
    return render(request, 'dashboard/staff_detail.html', context)

@login_required(login_url='user-login')
def index(request):
    orders = Order.objects.all()
    products = Product.objects.all()
    order_count = Order.objects.all().count()
    items_count = Product.objects.all().count()
    workers_count = User.objects.all().count()
    if request.method == 'POST':
        form = OrderForm(request.POST)
        if form.is_valid():
            instance = form.save(commit=False)
            instance.staff = request.user
            instance.save()
            return redirect('dashboard-index')
    else:
        form = OrderForm()
    context = {
        'orders': orders,
        'form': form,
        'products': products,
        'order_count': order_count,
        'items_count': items_count,
        'workers_count': workers_count,
    }
    return render(request, 'dashboard/index.html', context)

```

Рис 3.3.3 Частина коду імплементації логіки

3.4 Впровадження базового захисту у веб-додаток

Для впровадження захисту у цій курсовій роботі я впровадив систему авторизації користувачів та на основі рівня користувача після авторизації він може переглядати лише певний зміст та дані. Також є супер-користувачі це користувачі у яких є всіх права доступу й вони можуть заходити та користуватись адмін-панеллю та веб-додатком в цілому без будь-яких обмежень на відміну простих користувачів. Наприклад створений користувач Mike без будь-яких прав має лише можливість створювати замовлення з існуючих товарів та переглядати лише дані з категорії замовлення, які створив він, також Mike може переглядати власний профіль у веб-додатку та редагувати його

Створити запит

Товар:

Кількість:

Створити запит

Історія замовлень

Товар	Категорія	Кількість	Дата
Кепка	Головний убір	11	Dec. 10, 2023, 1:42 p.m.
Гольф	Верхній одяг	1	Dec. 10, 2023, 1:42 p.m.
Чоботи	Взуття	12	Dec. 10, 2023, 1:42 p.m.

Рис 3.4.1 Вигляд категорії замовлень для користувача Mike

Профіль

Інформація про профіль

Редагувати

Ім'я

Mike

Пошта

mikestaff@gmail.com

Номер

593816

Адреса

Київ




Рис 3.4.2 Профіль користувача Mike, який він може редагувати

Також під час тестування застосунку, я помітив, що користувачі, які не є авторизованими у систему можуть спокійно нею користуватись, якщо знають

URL-адресу певної сторінки у веб-додатку й це була велика проблема для мене, тому що це ставило під загрозу безпеку всієї системи та даних, тому що будь-який зломисник міг видалити, редагувати, пошкодити дані знаючи URL-адресу, яка веде на сторінки, що їх зберігають. Для захисту даних від такої атаки потрібно було додати в код спеціальний метод, який по замовчуванню встановлений у Django: `@login_required(login_url = 'user-login')`, цей метод, що встановлюється перед функціями перенаправляє користувачів, що не пройшли авторизацію на сторінку з авторизацією, це захистило мої дані у веб-додатку від атаки, що була описана вище.

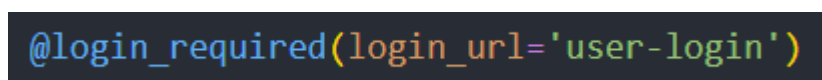
A screenshot of a code editor showing a line of Python code: `@login_required(login_url='user-login')`. The code is written in a dark-themed editor with syntax highlighting: '@login_required' is in blue, 'login_url=' is in green, and 'user-login' is in yellow.

Рис 3.4.3 Метод `@login_required` в коді курсової роботи.

ВИСНОВОК

В ході даної курсової роботи було розглянуто та проаналізовано теоретичну інформацію про розробку веб-додатків з-за допомогою фреймворку Django розробленого для мови програмування Python та системи керування базами даних. Розглянуті основні аспекти захисту даних від атак.

Для реалізації даного веб-додатку було використано наступні технології: мова програмування Python, фреймворк Django, система керування базами даних MySQL, набір інструментів та шаблонів Bootstrap та система контролю версій Git.

Як результат веб-додаток надає користувачам можливості для додавання, редагування, читання й видалення даних щодо товарів, працівників та замовлень, що відносяться до складу магазину одягу та його керуванням.

Проєкт має великий простір для вдосконалення й подальшого розвитку, також він може бути швидко перероблений під інші сфери продажу товарів, такі як склад автозапчастин, продуктів, тощо. В цілому проєкт є досить актуальним в сучасному світі з огляду на вирішення проблем логістики й бізнесу, що пов'язаний з продажом товарів і використання сучасних технологій розробки так і системи керування базами даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Документація Python[електронний ресурс]/режим доступу:
<https://docs.python.org/3/>
2. Документація Django[електронний ресурс]/режим доступу:
<https://docs.djangoproject.com/en/5.0/>
3. Документація Bootstrap[електронний ресурс]/режим доступу:
<https://getbootstrap.com/docs/5.3/getting-started/introduction/>
4. Документація Git[електронний ресурс]/режим доступу:
<https://git-scm.com/doc>
5. Документація GitHub[електронний ресурс]/режим доступу:
<https://docs.github.com/en>
6. Ерік Маттес. Пришвидшений курс Python: Практичний, проєктно-орієнтований вступ до програмування – Україна: Видавництво Старого Лева, 2022. – 600с.
7. Вікі-стаття Гвідо ван Россум [електронний ресурс]/режим доступу:
https://uk.wikipedia.org/wiki/Гвідо_ван_Россум
8. Документація MySQL[електронний ресурс]/режим доступу:
<https://dev.mysql.com/doc/>

ДОДАТКИ

Репозиторій проєкту на GitHub[електронний ресурс]/режим доступу:

<https://github.com/AstroMate/kursova>

Код файлу kurs/setting.py

```
"""
```

Django settings for kurs project.

Generated by 'django-admin startproject' using Django 4.2.7.

For more information on this file, see

<https://docs.djangoproject.com/en/4.2/topics/settings/>

For the full list of settings and their values, see

<https://docs.djangoproject.com/en/4.2/ref/settings/>

```
"""
```

```
from pathlib import Path
```

```
# Build paths inside the project like this: BASE_DIR / 'subdir'.
```

```
BASE_DIR = Path(__file__).resolve().parent.parent
```

Quick-start development settings - unsuitable for production

See <https://docs.djangoproject.com/en/4.2/howto/deployment/checklist/>

SECURITY WARNING: keep the secret key used in production secret!

```
SECRET_KEY = 'django-insecure-x50-c&cxcd1m6!lm8fzk1f2y^lgbp^px1)6%n6uv+7nz2%$kvu'
```

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',  
    'dashboard.apps.DashboardConfig',  
    'user.apps.UserConfig',
```

```
'crispy_forms',  
  
]  
  
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 = 'kurs.urls'
```

```
TEMPLATES = [  
  
    {  
  
        'BACKEND': 'django.template.backends.django.DjangoTemplates',  
  
        'DIRS': [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 = 'kurs.wsgi.application'
```

```
# Database
```

```
# https://docs.djangoproject.com/en/4.2/ref/settings/#databases
```

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'kursova',
        'USER': 'root',
        'PASSWORD': 'admin',
        'HOST': 'localhost',
        'PORT': '3306',
    }
}
```



```
# Password validation
```

```
# https://docs.djangoproject.com/en/4.2/ref/settings/#auth-password-validators
```

```
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
```

```
# https://docs.djangoproject.com/en/4.2/topics/i18n/
```

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

```
TIME_ZONE = 'UTC'
```

```
USE_I18N = True
```

```
USE_TZ = True
```

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

```
# https://docs.djangoproject.com/en/4.2/howto/static-files/
```

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

```
STATICFILES_DIRS = [
```

```
    BASE_DIR / 'static'
```

```
]
```

```
STATIC_ROOT = (BASE_DIR / 'asert/')
```

```
MEDIA_ROOT = (BASE_DIR / 'media')
```

```
MEDIA_URL = '/media/'
```

```
LOGIN_REDIRECT_URL = 'dashboard-index'
```

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

```
# https://docs.djangoproject.com/en/4.2/ref/settings/#default-auto-field
```

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

```
Код dashboard/views.py
```

```
from django.shortcuts import render, redirect
```

```
from django.http import HttpResponseRedirect
```

```
from django.contrib.auth.decorators import login_required
```

```
from .models import Product, Order
```

```
from .forms import ProductForm, OrderForm
```

```
from django.contrib.auth.models import User
```

```
# Create your views here.
```

```
@login_required(login_url='user-login')
```

```
def index(request):
```

```
    orders = Order.objects.all()
```

```
    products = Product.objects.all()
```

```

order_count = Order.objects.all().count()

items_count = Product.objects.all().count()

workers_count = User.objects.all().count()

if request.method == 'POST':

    form = OrderForm(request.POST)

    if form.is_valid():

        instance = form.save(commit=False)

        instance.staff = request.user

        instance.save()

        return redirect('dashboard-index')

else:

    form = OrderForm()

context = {

    'orders':orders,

    'form':form,

    'products':products,

    'order_count':order_count,

    'items_count':items_count,

    'workers_count':workers_count,

}

return render(request, 'dashboard/index.html', context)

@login_required(login_url='user-login')

```

```
def staff(request):

    workers = User.objects.all()

    workers_count = workers.count()

    order_count = Order.objects.all().count()

    items_count = Product.objects.all().count()

    context = {

        'workers' : workers,

        'workers_count': workers_count,

        'order_count':order_count,

        'items_count':items_count

    }

    return render(request, 'dashboard/staff.html', context)
```

```
@login_required(login_url='user-login')
```

```
def staff_detail(request, pk):

    workers = User.objects.get(id = pk)

    context = {

        'workers' : workers,

    }

    return render(request, 'dashboard/staff_detail.html', context)
```

```
@login_required(login_url='user-login')
```

```
def product(request):
```

```

workers = User.objects.all()

workers_count = workers.count()

order_count = Order.objects.all().count()

items = Product.objects.raw("SELECT * FROM dashboard_product")

items_count = Product.objects.all().count()

if request.method == "POST":

    form = ProductForm(request.POST)

    if form.is_valid():

        form.save()

        return redirect('dashboard-product')

else:

    form = ProductForm()

context = {

    'items' : items,

    'form' : form,

    'workers_count':workers_count,

    'order_count':order_count,

    'items_count':items_count,

}

return render(request, 'dashboard/product.html', context)

@login_required(login_url='user-login')

def product_delete(request, pk):

```

```

item = Product.objects.get(id=pk)

if request.method == 'POST':

    item.delete()

    return redirect('dashboard-product')

return render(request, 'dashboard/product_delete.html')

```

```

@login_required(login_url='user-login')

def product_update(request, pk):

    item = Product.objects.get(id=pk)

    if request.method == 'POST':

        form = ProductForm(request.POST, instance=item)

        if form.is_valid():

            form.save()

            return redirect('dashboard-product')

    else:

        form = ProductForm(instance=item)

    context = {

        'form' : form,

    }

    return render(request, 'dashboard/product_update.html', context)

@login_required(login_url='user-login')

```

```

def order(request):

    orders = Order.objects.all()

    order_count = Order.objects.all().count()

    workers = User.objects.all()

    workers_count = workers.count()

    items_count = Product.objects.all().count()

    context = {

        'orders':orders,

        'workers_count':workers_count,

        'order_count':order_count,

        'items_count':items_count,

    }

    return render(request, 'dashboard/order.html', context)

```

Код dashboard/model.py

```

from django.db import models

from django.contrib.auth.models import User

# Create your models here.

```

```

CATEGORY = (

    ('Верхній одяг', 'Верхній одяг'),

    ('Взуття', 'Взуття'),

    ('Штани', 'Штани'),

```



```
(('Куртка', 'Куртка'),  
 ('Головний убір', 'Головний убір'),  
)
```

```
class Product(models.Model):  
  
    name = models.CharField(max_length=100, null=True)  
  
    category = models.CharField(max_length=50, null=True, choices=CATEGORY)  
  
    quantity = models.PositiveIntegerField(null=True)
```

```
class Meta:  
  
    verbose_name_plural = 'Товар'
```

```
def __str__(self):  
  
    return f'{self.name}'
```

```
class Order(models.Model):  
  
    product = models.ForeignKey(Product, on_delete=models.CASCADE, null=True)  
  
    staff = models.ForeignKey(User, models.CASCADE, null=True)  
  
    order_quantity = models.PositiveIntegerField(null=True)  
  
    date = models.DateTimeField(auto_now_add=True)
```

```
class Meta:
```

```
    verbose_name_plural = 'Замовлення'
```

```
def __str__(self):
```

```
    return f'{self.product} замовив(-ла) {self.staff.username}'
```