**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ**
**ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА ТЕХНОЛОГІЙ**
**КАФЕДРА КОМП'ЮТЕРИЗОВАНИХ СИСТЕМ УПРАВЛІННЯ**

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____ Олександр ЛИТВИНЕНКО
"_____"_____ 2023 р.

# КВАЛІФІКАЦІЙНА РОБОТА

## (ПОЯСНЮВАЛЬНА ЗАПИСКА)
## ЗДОБУВАЧА ВИЩОЇ ОСВІТИ
## ОСВІТНЬОГО СТУПЕНЯ «МАГІСТР»

**Тема:** Програмний комплекс обліку роботи працівників логістичної авіакомпанії

**Виконавець:** Яна АХЛАМОВА

**Керівник:** Надія МАРЧЕНКО

**Нормоконтролер:** Євгеній ТУПОТА

**Київ 2023**

**НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ**

Факультет комп’ютерних наук та технологій

Кафедра комп’ютеризованих систем управління

Спеціальність 123 «Комп'ютерна інженерія

Освітньо-професійна програма «Системне програмування»

Форма навчання денна

ЗАТВЕРДЖУЮ

Завідувач кафедри

Олександр ЛИТВИНЕНКО

«_____»_____2023 р.

**ЗАВДАННЯ**

**на виконання кваліфікаційної роботи**

Ахламової Яни Олександрівни

1. Тема кваліфікаційної роботи Програмний комплекс обліку роботи працівників логістичної авіакомпанії

затверджена наказом ректора від «28» 08 2023 р. № 1494/ст

2. Термін виконання роботи (проєкту): з 02.10.2023 р. по 31.12.2023 р.

3. Вихідні дані до роботи (проєкту): середовище розробки *PyCharm*, мова програмування *Python*, фреймворки *Django*, *Bootstrap*, база даних *PosgresSQL*

4. Зміст пояснювальної записки: вступ, аналіз існуючих методів обліку працівників, вибір засобів розробки, програмна реалізація комплексу, приклад застосування висновки.

5. Перелік обов'язкового графічного (ілюстративного) матеріалу:

1) структура зв’язків бази даних;

2) архітектурний шаблон веб-додатку;

3) організація програмного коду;

4) інтерфейс програмного комплексу;

6. Календарний план

| № п/п | Етапи виконання кваліфікаційної роботи | Термін виконання етапів | Примітка |
|---|---|---|---|
| 1 | Провести пошук та аналіз джурел за темою кваліфікаційної роботи | 2.10.2023–2 .11.2023 | |
| 2 | Розробка плану кваліфікаційної роботи | 03.11.2023– 13.11.2023 | |
| 3 | Провести аналіз існуючих розробок та написати перший розділ | 14.11.2023– 24.11.2023 | |
| 4 | Розробити вимоги та написати другий розділ | 25.11.2023– 30.11.2023 | |
| 5 | Розробити програмний комплес та написати третій розділ | 31.11.2023– 10.12.2023 | |
| 6 | Оформити пояснювальну записки | 11.12.2023– 18.12.2023 | |
| 7 | Підготувати презентацію | 19.12.2023– 24.12.2023 | |
| 8 | Підговка до захисту | 25.12.2023– 31.12.2023 | |

7. Дата отримання завдання: «02» 10. 2023 р.

Керівник кваліфікаційної роботи _____ Надія МАРЧЕНКО

Завдання прийняв до виконання _____ Яна АХЛАМОВА

# РЕФЕРАТ

Кваліфікаційна робота на тему «Програмний комплекс обліку роботи працівників логістичної авіакомпанії ». Записка до кваліфікаційної роботи містить: 82 с., 30 рис., 40 літературних джерел, 1 додаток.

Ключові слова: ОБЛІК РОБОТИ,ВЕБ-ЗАСТОСУНОК, *DJANGO*.

Об’єктом дослідження є процес обліку роботи працівників логістичної авіакомпанії..

Предметом дослідження є програмний комплекс обліку роботи працівників логістичної авіакомпанії..

Метою кваліфікаційної роботи є  розробка програмного комплексу для обліку роботи працівників логістичної авіакомпанії.

Технічними та програмними засобами, що використовувалися при розробці кваліфікаційної роботи є персональний комп’ютер з операційною системою *Windows* 10, мова програмування *Python*, фреймворк *Django* та *Bootstrap*

Методи проектування: аналіз існуючих підходів, програмна реалізація програмного комплексу обліку працівників логістичної авіакомпанії у форматі веб-застосунку.

Наукова новизна полягає у комплексному підході до обліку роботи працівників.

Результати кваліфікаційної роботи рекомендується використовувати при обліку роботи працівників.

**ЗМІСТ**

# ВСТУП

Сучасний рівень технологій і їх швидкий розвиток роблять вплив на усі сфери життя. Бізнес і управління не виключені, і вимагають постійної оптимізації процесів. Стежити за робочим часом та продуктивністю працівників стає все важливішою задачею в умовах зростаючої конкуренції.

Історія обліку робочого процесу працівників почалася з ручних методів та простих інструментів, а з часом еволюціонувала до використання механічних обчислювальних машин та комп'ютеризації. У сучасному світі роль обліку робочого часу відіграють веб, десктоп та мобільні додатки, які дозволяють контролювати робочий час з будь-якого місця. Облік роботи працівників важливий для контролю робочого часу працівників, оцінки їхньої продуктивності, розрахунку заробітної плати та планування ресурсів.

**Актуальність теми.** Логістика в авіаційній індустрії є ключовим компонентом успішного функціонування та вирішення різних завдань в сучасному світі. Недостатня ефективність та автоматизація обліку роботи працівників у логістичній авіакомпанії призводить до затримок, неправильного розподілу завдань та підвищення витрат, що ускладнює оптимальне управління ресурсами та впливає на загальну продуктивність компанії.

Зараз, у змінному та динамічному світі, де вимоги до швидкості, ефективності постійно зростають, авіакомпанії стикаються з потребою у вдосконаленні систем управління персоналом. Оптимізація робочих процесів, точний облік робочого часу та контроль за виконанням завдань — це ключові аспекти успішного функціонування логістичних авіакомпаній. В умовах конкурентної боротьби і підвищених вимог пасажирів, важливо мати інструменти, які спрямовані на ефективне управління персоналом, забезпечуючи якість послуг та оптимальну роботу працівників.

**Метою** даної кваліфікаційної роботи є розробка програмного комплексу для обліку роботи працівників логістичної авіакомпанії, забезпечуючи облік робочого часу, моніторинг виконання завдань.

Для досягення поставленох мети необхідно:

1. Проаналізувати наявні програмні рішення та підходи до організіції та обліку роботи працівників логістичної авіакомпанії.
2. Проаналізувати засоби розробки для реалізації поставлених завдань.
3. Визначити вимоги до програмного комплексу.
4. Розробити структуру програмного комплесу.
5. Програмно реалізувати комплекс обліку роботи працівників логістичної авіакомпанії.

**Об’єкт дослідження**. Процес обліку роботи працівників логістичної авіакомпанії.

**Предмет дослідження**. Програмний комплекс обліку роботи працівників логістичної авіакомпанії.

**Наукова новизна отриманих результатів** полягає у комплексному підході до обліку роботи працівників.

У порівняні з існуєчими програмними розробками, запропонований підхід пропонує не лише окремі модулі для обліку робочого часу та моніторингу виконання завдань, але й інтегровану систему, що об'єднує ці функції та дозволяє вести комплексний облік всієї робочої діяльності працівників.

**Практичне значення отриманих результатів**. Результати даного проєкту можуть бути практично застосовані для підвищення ефективності управління персоналом логістичних авіакомпаній, що сприятиме покращенню якості послуг та оптимізації бізнес-процесів.

**Особистий внесок випускника**. Всі результати, представлені у кваліфікаційній роботі, отримані випускником особисто.

Кваліфікаційна робота складається з наступних основних розділів:

1) аналіз існуючих методів обліку роботи працівників:

− аналіз обліку роботи;

− аналіз засобів розробки;

− аналіз існуючих розробок щодо обліку роботи працівників;

2) вибір засобів розробки:

− порівняння мов програмування;

− порівняння баз даних;

− вибір фреймворку;

− вибір середовища розробки;

3) програмна реалізація комплексу:

− визначення вимог до програмного комплексу;

− опис структури програмного комплексу;

− розробка програмного комплексу;

− тестування роботи програмного комплексу;

4) робота програмного комплексу:

− можливості роботи працівників;

− можливості роботи адміністраторів.

Кожен розділ завершується висновками, що підбивають основні результати та визначені напрямки для подальших досліджень.

Розроблений веб-додаток для обліку роботи працівників відрізняється простотою впровадження та користування. Для його належної роботи потрібно лише налаштувати підключення до вже наявних джерел даних, таких як база даних з інформацією про працівників або системи контролю робочого часу. Всі інші етапи, такі як збір, обробка та зберігання даних про працівників, автоматизовані та виконуються програмним забезпеченням автоматично.

Цей простий у використанні веб-додаток спрощує впровадження системи обліку роботи, що дозволяє зосередитися на підключенні до вже існуючих джерел інформації та оптимізації його під потреби конкретної компанії. Такий підхід забезпечує високий рівень зручності та ефективності в управлінні робочим процесом, дозволяючи легко використовувати отриману інформацію для прийняття управлінських рішень.

# РОЗДІЛ 1
# АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ОБЛІКУ РОБОТИ

## 1.1. Аналіз обліку роботи працівників

Історія обліку, зокрема обліку роботи працівників, виникає ще в давнину і пов'язана з розвитком організацій і трудових відносин. Розглянемо історію обліку в цілому:

Ручний облік: Найперший етап в історії обліку пов'язаний із застосуванням ручних методів. Люди використовували прості інструменти та журнали для відслідковування різних видів інформації, включаючи витрати, доходи і працю працівників.

Механічні обчислювальні машини: З розвитком технологій з'явились механічні обчислювальні машини, такі як калькулятори та облікові машини, які полегшили ручний облік і розрахунки.

Ера комп'ютерів: У 20-му столітті настав період переходу до комп'ютеризації. Поява комп'ютерів і розвиток програмного забезпечення сприяли автоматизації обліку в різних галузях, включаючи бухгалтерію та управління персоналом.

Електронні таблиці: З поширенням персональних комп'ютерів у 1980-90-х роках з'явились електронні таблиці, такі як *Microsoft Excel*, що значно полегшили ведення обліку.

Веб-застосунки та мобільні додатки: У сучасному світі зростає популярність веб та мобільних додатків для обліку роботи. Вони дозволяють вести облік та контролювати робочий час з будь-якого місця, що особливо важливо для компаній з розподіленою робочою силою.

Облік роботи працівників важливий з таких причин:

− Контроль за робочим часом: Облік дозволяє визначити, скільки часу працівники витрачають на виконання своїх обов'язків. Це може бути

корисним для ефективного управління ресурсами та планування робочого часу.

- Оцінка продуктивності: Облік допомагає в оцінці продуктивності працівників. Збирання даних про час, витрачений на роботу, дозволяє управлінню визначити, як ефективно використовуються робочі години.
- Розрахунок заробітної плати: Облік робочого часу є основою для розрахунку заробітної плати працівників. Це важливо для визначення вартості праці та вчасної виплати заробітної плати.
- Планування ресурсів: Збір і аналіз даних про робочий процес допомагає компанії краще розуміти свої потреби в робочій силі. Це сприяє ефективному плануванню ресурсів та розподілу завдань.
- Визначення ефективність процесу: Розрахунок робочого часу може свідчити про ефективність робочого процесу. Це дає змогу визначити можливості для оптимізації та підвищення продуктивності.
- Дотримуйтеся правил і стандартів: авіаційна галузь вимагає дотримання правил щодо робочого часу. Час може допомогти компаніям дотримуватися цих вимог і уникнути можливих порушень.
- Управління витратами: відстеження робочого часу може виявити області, де можна зменшити витрати або оптимізувати використання ресурсів.

Зважаючи на складність та різноманітність завдань в авіаційній індустрії, облік роботи працівників логістичної авіакомпанії має багатоаспектний характер. Зупинимося на більш конкретних аспектах використання обліку праці в логістичних авіакомпаніях.

Технічне обслуговування обладнання: логістичні авіакомпанії використовують відстеження часу, щоб визначити оптимальні графіки технічного обслуговування літаків і обладнання, щоб уникнути незапланованих витрат на технічне обслуговування та максимізувати експлуатаційні ресурси.

Вантажні операції: Нарахування заробітної плати авіакомпанії логістики відіграє важливу роль у плануванні та виконанні операцій із завантаження та

розвантаження вантажів. Ефективне управління часом співробітників у цих процесах скорочує час очікування та збільшує швидкість обробки вантажу.

Обслуговування клієнтів і відстеження. Групи обслуговування клієнтів і відстеження використовують відстеження часу, щоб постійно контролювати стан відправлення, запити клієнтів і своєчасні відповіді на проблеми.

Розвантаження та завантаження літаків. Співробітники, відповідальні за розвантаження та завантаження літаків, використовують відстеження часу, щоб спланувати оптимальні стратегії завантаження, щоб максимізувати місткість вантажу та забезпечити ефективну конфігурацію вантажного простору.

Надзвичайні ситуації та реагування: бізнес-документація важлива для організації планування робочої сили на випадок надзвичайної ситуації, наприклад: технічні збої, аварійні посадки або інші непередбачені події, що вимагають швидкого та ефективного реагування.

Оптимізація робочого часу персоналу: управління працею допомагає визначити можливості для оптимізації робочого часу, таким чином уникаючи звільнення персоналу в періоди низької потреби в персоналі.

### 1.2. Аналіз середовища використання

Десктопні програми з'явилися на ринку програмного забезпечення задовго до веб-рішень. Проте, коли широкопоширене покриття Інтернету стало нормою, веб-застосунки почали здобувати популярність і в кінцевому підсумку здобули набагато більшу частку ринку. Це, звичайно, не означає, що ера десктопних програм завершилася. Як десктопні, так і веб-застосунки мають свої переваги та недоліки.

Веб-додаток – це програмне забезпечення, що знаходиться на віддаленому сервері і працює у веб-браузері. Простіше кажучи, веб-додатки запитують контент з сервера і негайно генерують веб-документи для користувачів. У цьому контексті важлива роль відводиться веб-браузеру, оскільки він інтерпретує скрипти та відображає додаток користувачам у вигляді веб-сторінок, а не рядків коду.

Також важливо розуміти, що веб-додатки та веб-сайти – це різні типи веб-рішень. У той час як стандартні веб-сайти зазвичай використовують статичні макети, веб-додатки є динамічними та інтерактивними. Вони дозволяють користувачам не тільки переглядати вміст, а й вносити зміни до інтерфейсу та взаємодіяти з його елементами. Наприклад, веб-додатки включають до себе дошки оголошень, книги для гостей, електронні поштові програми, соціальні медіа, бронювальні платформи та багато іншого.

З іншого боку, десктопний додаток – це програмне забезпечення, яке встановлюється на комп'ютері (нотбуці або ПК). Користувач може завантажити його з Інтернету або встановити через службу зберігання даних. Десктопні додатки працюють як самостійне програмне забезпечення, що означає, що їх можна використовувати в автономному режимі і не потребують доступу до Інтернету або веб-браузера для роботи. Деякі класичні приклади десктопних додатків включають графічні редактори, електронні таблиці, медіаплеєри, текстові редактори.

Загалом, десктопні та веб-додатки можуть виконувати схожі функції, але їх природа відрізняється. Тому детальніше розглянемо переваги та недоліки веб- та десктопних додатків.

Переваги веб-додатків:

- Не потрібна установка – для доступу як до веб-додатків, так і до десктопних необхідно комп'ютер. Однак веб-додатки не потребують установки. Через це вони не займають місця на вашому жорсткому диску. Достатньо *URL*-адреси, імені користувача та пароля для використання цих додатків.
- Немає проблем з оновленнями – веб-додатки не вимагають від користувача жодних дій для оновлення. Оскільки оновлення застосовуються безпосередньо до сервера, користувачі завжди отримують останню версію програми при кожному запуску. Це означає відсутність надокучливих сповіщень, перезавантажень та повільної роботи старої версії.
- Незалежність від платформи – оскільки веб-додатки є кросплатформовими, їх можна запускати на різних платформах. По суті, їх можна

використовувати на будь-якому пристрої з доступом до Інтернету та веб-браузером. Хоча можуть виникати проблеми з сумісністю браузерів, не потрібно розробляти додаток під певну платформу, що розширює можливості додатку і зменшує витрати.

- Легкий доступ – рівень доступності є ще однією важливою відмінністю між десктопними та веб-додатками. Не потрібно мати свій комп'ютер під рукою, щоб використовувати веб-додаток. Користувачі можуть отримувати доступ до веб-додатків з будь-якого ПК (або іншого пристрою) з підключенням до Інтернету. Тому місце знаходження та пристрій, яким користуєтеся, зовсім не мають значення.
- Доступ з мобільного пристрою – більшість веб-додатків мають адаптивний дизайн і легко доступні з мобільного пристрою. Оскільки сучасні люди використовують смартфони дуже активно, це велика перевага.

Недоліки веб-додатків:

- Безпека – веб-додатки піддаються більшій кількості загроз безпеки, ніж десктопні програми. Є кілька причин для цього, такі як доступ до Інтернету, різноманітні користувачі, які не завжди ідентифіковані, кібератаки тощо.
- Залежність від Інтернету – інтернет широко розповсюджений, але він ще не скрізь. Якщо зникне покриття Інтернетом, веб-додатки будуть непридатні до використання.
- Повільна продуктивність – веб-додатки також вважаються повільнішими, ніж десктопні. Хоча це не завжди так, все залежить від швидкості Інтернету та багатьох інших факторів, передача великої кількості даних кожного разу, коли запитуєте *HTML*-документ, може призвести до сповільнення роботи додатку.
- Залежність від браузера – існує багато веб-браузерів. Тому веб-додатки повинні бути сумісними з усіма ними, щоб надати користувачам можливість отримати доступ до рішення з будь-якого пристрою. Це також робить вас залежними від того, як браузер підтримує певні функції та можливості.

Якщо це не враховано, користувачі можуть зіткнутися з деякими труднощами під час роботи з додатком.

Переваги десктопних додатків :

- Не залежить від Інтернету – З десктопними додатками не потрібно турбуватися про повільне підключення до Інтернету. По суті, можна отримати доступ до свого додатка, доки зможете отримувати доступ до свого комп'ютера. Тим часом в Інтернет необхідний під час використання веб-додатків.
- Конфіденційність – у порівнянні з веб-додатками, десктопні додатки пропонують набагато сильнішу захист персональних даних. Коли йдеться про конфіденційність, користувачі зазвичай відчувають себе більш комфортно, зберігаючи свою інформацію на своїх комп'ютерах, а не в Інтернеті. Звісно, дані на комп'ютері також не є повністю безпечними, але вони точно більш безпечні, ніж у мережі.
- Краще використання ресурсів комп'ютера – в цілому, десктопні додатки швидше за веб-додатки, оскільки вони працюють незалежно на вашому комп'ютері і не вимагають підключення до Інтернету. В цьому випадку незалежність від Інтернету призводить до позитивного результату.
- Нижчі витрати на хостинг – розробка десктопного додатка потребує мінімальних витрат на хостинг. Все, що потрібно тут – це обслуговувати кілька сторінок та завантажувати файл на кожен пристрій. Тим часом витрати на хостинг веб-додатків набагато вищі.

Недоліки десктопних додатків:

- Стаціонарність – на відміну від веб-додатків, десктопні додатки можна використовувати лише на комп'ютерах, на яких вони встановлені. Так, наприклад, якщо змінити комп'ютер, це не вплине на спілкування з веб-додатком. Але якщо використовувати десктопний додаток, доведеться встановлювати його на кожному новому пристрої і починати все спочатку.

- Потрібно встановлювати додаток - на відміну від веб-додатків, десктопні додатки вимагають установки, що означає, що вам потрібно мати вільне місце на жорсткому диску. Процес установки також займає певний час.
- Оновлення – як вже зазначалося, десктопні додатки мають бути встановлені на кожному комп'ютері окремо. Те ж саме стосується оновлень. Потрібно переконатися, що всі користувачі оновили додаток, незалежно від кількості – чи це кілька людей, чи тисячі.
- Додаткове потребується місце для зберігання – на відміну від веб-додатків, десктопні додатки потребують певного місця на комп'ютері для встановлення та безперешкодної роботи. Якщо на жорсткому диску недостатньо вільного місця, доведеться видаляти деякі інші рішення або файли.

Розробка веб-застосунку для обліку роботи працівників логістичної авіакомпанії виявиться надійним вибором, оскільки дозволить забезпечити доступ до облікових даних з будь-якого місця за наявності Інтернету, уникне потреби встановлення програм на кожному пристрої та забезпечить автоматичні оновлення, що спростить управління системою і полегшить роботу працівників, які працюють у різних місцях.

### 1.3. Огляд існуючих розробок

На сьогоднішній день, логістичні авіакомпанії активно використовують програмні комплекси для обліку роботи своїх працівників, оскільки це важливий інструмент для оптимізації операцій та забезпечення ефективного функціонування авіакомпаній у конкурентному середовищі. Огляд існуючих розробок у даній області допомагає краще розуміти, які функції та можливості вже існують, а також визначити прогалини та напрямки для подальшого дослідження та розробки.

Деякі існуючі розробки та рішення, пов'язані з обліком роботи працівників в логістичних авіакомпаніях, включають в себе:

1. *SAP ERP*: (*Enterprise Resource Planning*) – це інтегрована система управління ресурсами підприємства, яка може використовуватися для обліку роботи та управління персоналом в авіаційній і логістичній галузях (рис.1.1).

Основні функції:

- Управління активами: *SAP* гарантує, що кожен елемент обладнання у системі потребує належного технічного обслуговування. Це забезпечує безперервне координацію планування, розкладу та виконання обслуговування, максимізуючи час експлуатації.
- Фінанси: Оптимізує фінансові процеси – від транзакцій та аналітики до планування, а також ризиків та відповідності підприємства. *SAP* допомагає виявляти продажі, заборгованості, списання поганих боргів та витрати.
- Виробництво: Пов'язано з процесом проектування продукту та управлінням життєвим циклом продукту. *SAP* підтримує функції з виробничого інжинірингу, операцій з продукцією, управлінням якістю та узагальненим аналізом виробництва.
- Продажі: Охоплює функції, пов'язані з управлінням контрактами, замовленнями на продаж та підтримкою послідовності даних про продажі.
- Дослідження та інжиніринг продукту: *SAP* надає повний огляд життєвого циклу продукту від початкового етапу до виробництва. Це також стосується завдань, пов'язаних з підприємницьким портфелем, управлінням проектами, відповідністю продукту та управлінням життєвим циклом продукту.
- Угода про майстер обслуговування та управління даними: Ці функції забезпечують прибуткові результати послуг. У *SAP* є функції для управління майстер-даними обслуговування та угодами, управлінням запасами обслуговування, управлінням підписками на замовлення, управлінням фінансовими службами та управлінням операціями обслуговування.
- Джерела та закупівля: Стосується функцій обробки закупівлі та оформлення замовлень. У *SAP* є функції, що підтримують операційну закупівлю, управління рахунками-фактурами, централізовану закупівлю, джерела постачання, управління контрактами та управління постачальниками.

- Управління ланцюжком постачання: *SAP* може надати повний огляд всіх процесів, пов'язаних з ланцюжком постачання. У них є модулі, які оцінюють продажі, закупівлі, замовлення на перевезення товарів, управління складом та управління запасами.

Плюси:

- Інтегрована система, яка охоплює різні аспекти бізнесу, включаючи фінанси, облік ресурсів, управління персоналом та інше.
- Забезпечує широкий спектр можливостей для управління операціями, включаючи логістику та авіацію.
- Можливість індивідуалізації і розширення функціональності.

Мінуси:

- Високі витрати на впровадження та підтримку.
- Складна система, яка може вимагати додаткового навчання працівників.


Рис. 1.1. *SAP ERP*

2. *Oracle Transportation Management*: Ця програма від *Oracle* спеціалізується на управлінні транспортними операціями, включаючи планування, відстежування та облік роботи працівників, пов'язаних з транспортом (рис. 1.2).

Основні функції:

- Моделювання логістичної мережі: Ця функція надає логістичній компанії простий спосіб стратегії. Можна використовувати обмеження та дані

реального світу для створення ідеальних сценаріїв і досягнення високоточних результатів, які моделюють вплив змін на операції. Порівняння в реальному часі дозволяє "моделювати" прийняття рішень на основі даних, перш ніж запускати їх у реальному світі.

- *3D*-візуалізація: Інструмент *3D*-конфігурації навантаження надає можливість переглядати оптимальне наповнення, наприклад, вантажівки або контейнера, з різних ракурсів. Він розкладає відвантаження за географічними ознаками, дозволяючи точно бачити, які вантажні одиниці розгружаються на першому пункті доставки.
- Моделювання транспортної мережі: Можна моделювати транспортну мережу вашої компанії в системі управління транспортом *Oracle*, а потім дозволити програмі ідентифікувати оптимальний шлях для замовлень з урахуванням вартості та часу транзиту. Візуалізувати мережу на карті та бачити маршрути, якими пройдуть замовлення на шляху до кінцевих пунктів призначення.
- Реальний час даних: Надає в реальному часі дані, такі як умови дорожнього руху, випадки на дорозі та погоду за один клік кнопки.
- Від початку до кінця замовлення та відвантаження: Ця функція тримає в курсі того, що відбувається з товари у транзиті. Потужна технологія повторної відправки разом з вбудованими можливостями робочого процесу повідомляє про можливі проблеми до того, як вони виникнуть.
- Закупівля транспорту: Ця функціональність дозволяє підготувати та провести торги з постачальниками послуг. Після завершення торгівницького процесу оптимізаційний алгоритм використовує критерії для розумного призначення перевізників.
- Функція запиту тарифів: Функція запиту тарифів дозволяє перевіряти тарифи та визначати всі можливі варіанти доставки товарів до їх пунктів призначення. Аналізує всі види транспорту, перевізників та рівні обслуговування, щоб ідентифікувати найбільш підходящі варіанти та впорядковувати їх за вартістю та часом транзиту.

Плюси:

- Ефективне управління транспортними операціями та логістикою.
- Підтримка глобальних маршрутів і складського управління.
- Інтеграція з іншими системами *Oracle*.

Мінуси:

- Вимагає досвіду та ресурсів для впровадження та підтримки.
- Високі вартості ліцензій та обслуговування.


Рис. 1.2. *Oracle*

3. *Microsoft Dynamics 365*: *Microsoft* пропонує рішення для управління ресурсами підприємства, яке може бути налаштоване для використання в галузі логістики і авіації (рис.1.3).

Основні функції:

- Управління відносинами з клієнтами (*CRM*): Динамічна система *CRM* дозволяє вести облік клієнтів, управляти контактами, взаємодіяти з клієнтами та вдосконалювати комунікацію з ними.
- Управління продажами: Набір інструментів для керування процесом продажу, від створення лідів до управління угодами та кошторисами, забезпеченням прогнозування продажів та аналітики.
- Управління маркетингом: Функції для планування та виконання маркетингових кампаній, ведення рекламних заходів, створення контенту та взаємодії з аудиторією.

- Управління операціями: Включає інструменти для оптимізації бізнес-процесів, автоматизації управління запасами, фінансових операцій, управлінням ланцюгом постачання.

- Управління фінансами: Модуль для обліку фінансів, фінансового планування, бюджетування, оподаткування, обліку зобов'язань та фінансової звітності.
- Служба підтримки клієнтів (*Customer Service*): Інструменти для взаємодії з клієнтами, вирішення їхніх запитань та проблем, автоматизація процесу обробки звернень.
- Управління ресурсами людей (*Human Resources*): Функціонал для керування людськими ресурсами, управлінням персоналом, плануванням навчання та розвитку персоналу.
- Аналітика та звітність: Інструменти для аналізу даних, побудови звітів, створення графіків та прогнозування на основі даних, зібраних з усіх сфер діяльності бізнесу.

Плюси:

- Інтегрована платформа, що охоплює *ERP* та *CRM* функціональність.
- Можливість інтеграції з іншими продуктами *Microsoft*.
- Зручний інтерфейс та велика спільнота користувачів.

Мінуси:

- Вартість відносно висока для малих і середніх підприємств.
- Обмеження в функціональності порівняно з іншими *ERP*-системами.


Рис. 1.3. *Microsoft Dynamics 365*

4. *IBM Sterling Transportation Management System*: Ця платформа розроблена для управління транспортними операціями, включаючи відстежування роботи працівників і оптимізацію маршрутів.

Основні функції:

- Планування транспорту: *IBM Sterling TMS* дозволяє створювати оптимальні маршрути та графіки доставок на основі різних параметрів, таких як час, вартість, тип транспорту тощо.
- Управління вантажем: Система дозволяє відслідковувати рух вантажу та координувати його доставку з точністю до деталей.
- Оптимізація маршрутів: *IBM Sterling TMS* враховує різноманітні фактори, щоб забезпечити найбільш ефективні маршрути, зменшуючи витрати та час доставки.
- Управління витратами: Система надає інструменти для контролю витрат на транспортні послуги, що дозволяє планувати бюджети та оптимізувати витрати на логістику.
- Інтеграція зі сторонніми системами: *IBM Sterling TMS* може легко інтегруватися з іншими системами управління складом, *ERP*-системами та іншими логістичними програмами для покращення управління ланцюгами постачання.

- Аналіз та звітність: Система надає засоби для аналізу даних про транспортні потоки, що допомагає приймати виважені рішення та оптимізувати процеси доставки.
- Управління відносинами з клієнтами: *IBM Sterling TMS* надає можливість взаємодіяти з клієнтами, забезпечуючи доступ до інформації про статус доставок та спрощуючи комунікацію.
- Глобальний охоплення: Система дозволяє управляти транспортними потоками на глобальному рівні, об'єднуючи процеси доставки в різних частинах світу.

Плюси:

- Потужна система для управління транспортними операціями.
- Можливість оптимізації маршрутів та управління запасами.
- Розширюється та налаштовується для потреб конкретного підприємства.

Мінуси:

- Високі витрати на впровадження та підтримку.
- Вимагає фахівців для ефективного використання.

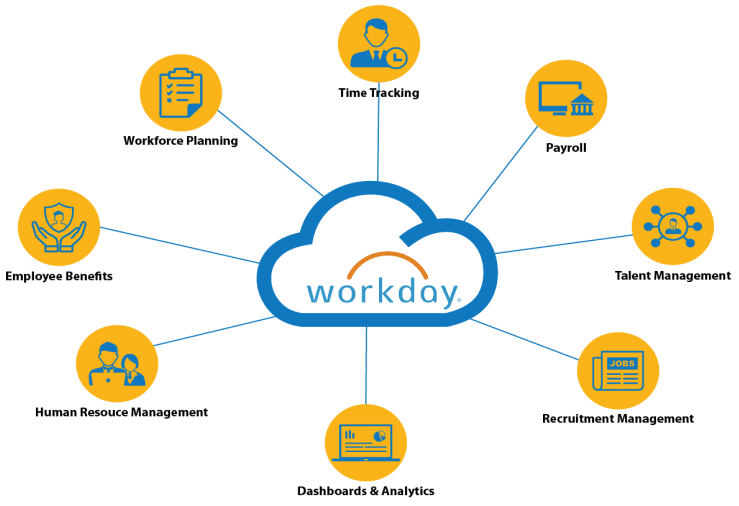


Рис.1.4. *Workday*

5. *Workday*: Ця програма використовується для управління робочим часом, відпустками та іншими аспектами роботи працівників (рис. 1.4).

Основні функції:

- Управління людськими ресурсами (*HRM*): Модуль для управління всіма аспектами життєвого циклу працівника, включаючи найм, звільнення, відділення, розвиток кар'єри, оцінку роботи, компенсації, навчання та розвиток.
- Фінансове управління: Функції для управління бюджетами, обліку фінансів, оподаткування, фінансової звітності, операцій з обліку обігу коштів, закупівель та інвестицій.
- Управління операціями: Модуль для управління процесами поставок, інвентаризацією, управлінням проектами, оптимізацією бізнес-процесів та оперативним управлінням.
- Аналітика та звітність: Інструменти для аналізу даних, побудови звітів, аналітики продуктивності та візуалізації даних для прийняття більш обгрунтованих управлінських рішень.
- Управління талантами: Модуль для виявлення, розвитку та утримання талантів в організації, включаючи планування наступництва та управлінням кар'єрним ростом.
- Управління відносинами з клієнтами (*CRM*): Функції для ведення бази клієнтів, взаємодії з клієнтами, створення маркетингових кампаній та підтримки клієнтів.
- Управління ризиками та відповідністю (*GRC*): Модуль для контролювання ризиків, дотримання внутрішніх та зовнішніх стандартів, а також для управління комплаєнсом та безпекою даних.
- Адаптивність та персоналізація: *Workday* надає можливості для адаптації системи до потреб конкретної організації, дозволяючи налаштувати функції та інтерфейс для кращої продуктивності користувачів.

Плюси:

- Простий та інтуїтивно зрозумілий інтерфейс.
- Підтримка управління робочим часом, відпустками та оплатою праці.
- Хмарне рішення, що полегшує підтримку та оновлення.

Мінуси:

- Орієнтована на управління людськими ресурсами, тому може бути менш підходящою для складних логістичних операцій.
- Може бути менше функціональності порівняно зі спеціалізованими системами.

### 1.4. Висновки до розділу

У даному розділі було коротко розглянуто історію обліку роботи та необхідність обліку роботи в логістичній авіакомпанії. Також проаналізовано готові програмні рішення та можливі засоби розробки.

Розробка веб-застосунку для обліку роботи працівників логістичної авіакомпанії виявляється кращим вибором порівняно з десктоп-додатком з огляду на кілька ключових моментів.

Почнемо з того, що логістична авіакомпанія має працівників, які працюють у різних місцях. Веб-застосунок дозволить забезпечити доступ до облікових даних з будь-якого пристрою, на якому є Інтернет, незалежно від місця розташування працівника.

Одним з ключових аспектів, який підтримує вибір веб-додатка, є відсутність необхідності установки. Це означає, що працівники авіакомпанії зможуть отримати доступ до системи обліку без необхідності завантаження та встановлення програми на кожному окремому комп'ютері. Замість цього, їм достатньо мати доступ до веб-браузера.

Оновлення веб-додатків автоматично застосовуються до сервера, тому користувачі завжди отримують доступ до останньої версії програми при запуску. Це означає відсутність проблем з управлінням оновленнями для кожного окремого пристрою працівника.

Системи обліку роботи працівників в авіалогістиці є потенційною областю для досліджень, оскільки вони охоплюють багатоаспектний характер діяльності та стають важливим інструментом оптимізації операцій у логістичних авіакомпаніях. Системи обліку праці статають невід'ємною частиною управління персоналом,

сприяючи забезпеченню ефективного використання ресурсів та оптимізації бізнес-процесів у даній галузі.

# РОЗДІЛ 2
# ВИБІР ЗАСОБІВ РОЗРОБКИ ПРОГРАМНОГО КОМПЛЕКСУ

## 2.1. Порівняння мов програмування

Різні мови програмування використовуються для створення веб-додатків з різними перевагами та особливостями. Кожна мова має свої унікальні особливості та переваги, які впливають на спосіб її використання у розробці.

### 2.1.1. *JavaScript*

*JavaScript* (рис.2.1) – найпопулярніша мова програмування для веб-розробки, і це не випадково. Вона використовується як для фронтенду, так і для бекенду, що робить її універсальним вибором. Крім того, вона підтримується усіма основними веб-браузерами, що означає, що будь-який код, написаний на *JavaScript*, буде сумісний майже з будь-яким пристроєм.

Однією з основних переваг *JavaScript* є його легкість використання. Вона має відносно простий синтаксис і легка для вивчення. Крім того, у *JavaScript* є велика бібліотека фреймворків і інструментів, таких як *React* і *Angular*, які полегшують та прискорюють розробку.

Переваги:

- Динамічність та універсальність: Дозволяють створювати різноманітні застосування.
- Виконання на клієнтському боці: Покращують швидкість відтворення сторінок та користувацький досвід.
- Широке поширення та активна спільнота: Мають велику кількість користувачів та активних розробників.
- Інтеграція з *HTML* та *CSS*: Допомагають у безперервній веб-розробці.

- Можливість створення інтерактивних елементів: Забезпечують динамічний інтерфейс.

Недоліки:

- Неспільна підтримка браузерами: Виникають проблеми сумісності з різними браузерами.
- Відсутність стандартизації: Призводить до проблем сумісності між різними реалізаціями.
- Слабка типізація: Може викликати непередбачувану поведінку, ускладнюючи відлагодження.
- Уразливості безпеки: Існує ризик вразливостей, таких як міжсайтовий скриптінг (*XSS*) та міжсайтове підроблення запитів (*CSRF*).
- Менша продуктивність: Можуть бути менш продуктивними, особливо для великих або складних додатків.


Рис. 2.1. Логотип *JavaScript*

### 2.1.2. *Java*

*Java* (рис. 2.2) – популярна мова програмування, яка широко використовується для веб-розробки. Вона відома своєю швидкістю та масштабованістю, що робить її відмінним вибором для великих проектів. *Java* також має високу переносимість, що означає, що код, написаний на *Java*, може бути запущений практично на будь-якому пристрої.

Однією з основних переваг *Java* є її безпека. Вона має вбудовані засоби безпеки, які зменшують її вразливість до атак, що робить її відмінним вибором для проектів, які потребують високого рівня захисту. У *Java* також існує велика спільнота розробників, які діляться своїм досвідом та знаннями, що робить його легким у використанні та отриманні допомоги у разі потреби.


Рис. 2.2. Логотип *Java*

Переваги:

- Незалежність від платформи, що дозволяє легко розгортати на різних операційних системах.
- Велике та визначене спільнотою співтовариство, яке надає широкий вибір бібліотек та підтримку.
- Міцні функції безпеки, що робить її популярним вибором для підприємств.
- Широкий спектр застосувань, включаючи настільні, мобільні та веб-розробки.
- Сильна перевірка типів та обробка помилок, що призводить до стабільного та безпечного коду.

Недоліки:

- Повільна продуктивність порівняно з мовами нижчого рівня, такими як *C++*.
- Висока крутизна кривої вивчення, особливо для початківців.
- Довготривала синтаксична конструкція порівняно з іншими мовами, що робить код менш конкретним.

- Може призводити до більших розмірів файлів порівняно з іншими мовами.
- Може бути менш гнучкою порівняно з динамічнотипізованими мовами, такими як *Python*.

### 2.1.3. *Python*

*Python* – ще одна популярна мова програмування для веб-розробки. Вона відома своєю простотою та легкістю використання, що робить її чудовим вибором для початківців. *Python* використовується як для розробки фронтенду, так і для бекенду, а також має велику бібліотеку фреймворків та інструментів, що прискорює та полегшує розробку.

Однією з основних переваг *Python* є його зрозумілість. Синтаксис мови розроблений таким чином, що його легко читати і розуміти, що полегшує розуміння та підтримку коду. *Python* також має великі можливості масштабування, тому є відмінним вибором для великих проектів, які потребують складного коду.


Рис. 2.3. Логотип *Python*

Переваги:

- Легкість вивчення та використання, зрозуміла синтакса
- Велика та активна спільнота, що надає значну підтримку
- Чудова для прототипування та швидкості розробки
- Широкий спектр застосувань, включаючи наукові обчислення, аналіз даних та веб-розробку
- Висока зчитуваність та збереження коду

Недоліки:

- Обмеження продуктивності порівняно з мовами нижчого рівня

- Динамічна типізація може призводити до помилок під час виконання
- Може бути не підходить для задач із високими вимогами до пам'яті
- Може працювати повільніше у деяких випадках, наприклад, з великими наборами даних
- Може бути менш ефективною для певних завдань

### 2.1.4. *Ruby*

Хоча мова *Ruby* відома своїм фреймворком *Ruby* on *Rails*, вона має свої власні переваги. Ця сценарна мова використовується як у фронтенді, так і у бекенді веб-розробки.

*Ruby* (рис. 2.4) – це загального призначення, зручна у використанні, відкрита, гнучка, інтерпретована, легко вивчається, надійна, динамічно типізована, об'єктно-орієнтована, універсальна, інтуїтивно зрозуміла мова, яка може бути застосована до численних веб-продуктів, від веб-сайтів до веб-додатків та більше. Її високорівневий синтаксис особливо легкий у використанні, оскільки він дуже схожий на англійську мову.


Рис. 2.4 Логотип *Ruby*

Переваги:

- Динамічна і виразна, що робить її відмінною для прототипування та швидкої розробки.
- Легка вивчення та використання, з чітким синтаксисом.
- Велике та відоме співтовариство, яке забезпечує широкі бібліотеки та підтримку.

- Хороша для швидкої розробки з увагою на зручність читання та підтримку коду.
- Динамічне типізування може призводити до більш гнучкого та адаптивного коду.

Недоліки:

- Повільна продуктивність порівняно з іншими мовами, такими як *C++* та *Go*.
- Обмежена підтримка одночасності та паралелізму.
- Неоднакова обробка помилок та підтримка відлагодження.
- Може бути менш продуктивною за інші мови, особливо для великих або складних застосунків.
- Динамічний типізм може бути менш безпечним за статично типізовані мови, такі як *Java* або *Go*.

Для обліку роботи працівників у логістичній авіакомпанії, мова програмування *Python* є відмінним вибором. Завдяки своїй простоті, легкості вивчення та розуміння синтаксису, *Python* відомий своєю дружністю до користувача. Це особливо корисно для створення програм, які необхідні для управління персоналом, складом, та різними логістичними аспектами авіакомпанії

### 2.2. Порівняння баз даних

Керування базами даних у веб-додатках є важливою частиною розробки програмного забезпечення та адміністрування. Бази даних гарантують, що дані додатків зберігаються та організовані належним чином, і можуть бути оброблені та отримані швидко у відповідь на запит користувача.

#### 2.2.1. *SQL*

Реляційні бази даних є найбільш поширеним та традиційним типом баз даних, що ґрунтується на концепції таблиць, рядків і стовпців. Вони використовують структуровану мову запитів (*SQL*) для визначення та маніпулювання даними, і

застосовують строгі правила та обмеження для забезпечення цілісності та однорідності даних. Реляційні бази даних ідеально підходять для веб-додатків, які потребують зберігання та опитування чітко визначених, структурованих та нормалізованих даних, наприклад, електронні торгові системи, банківські системи або системи обліку запасів. Деякі з найпопулярніших реляційних баз даних для веб-розробки це *MySQL*, *PostgreSQL*, *Oracle* та *SQL Serve*.

*MySQL* – це система управління базами даних з відкритим вихідним кодом, яка широко використовується для веб-застосунків та інших програм. Сумісна з різними операційними системами та мовами програмування, такими як *Java*, *C++, Python* тощо. Ця реляційна база даних максимізує продуктивність і утворює повний стек веб-розробки, спільно використовуючи інші програмні продукти з відкритим вихідним кодом, такі як *PHP*, *Apache*, *Linux* і т.д. В даний час *MySQL* має комерційні та спільнотні версії, в залежності від потреб користувача.

*Microsoft SQL Server* – продукт корпорації *Microsoft*, призначений для зберігання та доступу до даних на кількох серверах з врахуванням продуктивності та безпеки. Як реляційна база даних, *MS SQL Server* забезпечує безпеку інформації на основі структури таблиць та вбудованих опцій високої доступності та відновлення після аварій, таких як *Always On Availability Groups* та *Failover Clustering. Розроблений Microsoft, SQL Server* краще інтегрується з іншими продуктами *Microsoft*, такими як *.NET Framework* і *Visual Studio.*

*PostgreSQL* – це відкрита реляційна база даних для розробки веб-застосунків, відома своєю масштабованістю, надійністю та розширюваністю. *PostgreSQL* - це чудовий спосіб заощадження грошей, оскільки цілком безкоштовний на сучасних платформах, таких як *Windows*, *MacOS*, *Linux* та *UNIX*, без ліцензійних витрат або обмежень використання. Основною перевагою *PostgreSQL* є підтримка розширеного *SQL*, що допомагає розробникам працювати з складними обробками та аналізом даних.

*SQLite* – одна з найкращих баз даних для веб-застосунків, написаних на мові програмування *C*. Навіть якщо ви зберігаєте великі об'єми інформації та бінарних об'єктів розміром у кілька гігабайтів, *SQLite* легко керується та надає максимальну

продуктивність. На відміну від інших традиційних клієнтсько-серверних баз даних для веб-сайтів, *SQLite* зберігає дані в одному файлі і особливо підходить для мобільних пристроїв, вбудованих систем та невеликих застосунків. Це найкращий вибір для розробників мобільних додатків, які шукають швидку та надійну базу даних.

### 2.2.2. *NoSQL*

Нереляціні бази даних – це широка категорія баз даних, які не використовують реляційну модель, а використовують різні структури та формати даних, такі як пари ключ-значення, документи, графи або стовпці. Вони не використовують *SQL*, але мають свої власні мови запитів або *API*. *NoSQL* бази даних призначені для обробки великих обсягів даних, які є неструктурованими, напівструктурованими або динамічними, та забезпечують високу масштабованість, доступність та продуктивність. Вони підходять для веб-додатків, які потребують зберігання та обробки даних, які є складними, різнорідними або змінюються, таких як соціальні медіа, аналітика чи системи реального часу. Деякі з найпопулярніших *NoSQL* баз даних для веб-розробки це *MongoDB*, *Redis*, *Cassandra* та *Neo4j*.

*MongoDB* – це документно-орієнтована база даних типу *NoSQL*, яка є безкоштовною для всіх користувачів. Створена для зберігання неструктурованих даних у форматі, схожому на *JSON*, *MongoDB* відома своєю гнучкістю, підтримкою горизонтального масштабування для декількох серверів та автоматичним шаруванням. Цю систему управління базами даних довіряють навіть великі компанії, такі як *Facebook* та *IBM*, як найкращу базу даних для веб-сайтів. *MongoDB* підходить для всіх типів користувачів, оскільки для роботи з нею не потрібно знати що-небудь про взаємозв'язки або структури даних.

*Cassandra* – це колоночно-орієнтована система баз даних, побудована на принципах *Dynamo*, але вдосконалена у деяких своїх проблемах. *Cassandra* не пропонує розробникам сильної однорідності. Натомість ця база даних забезпечує високу доступність, дозволяючи користувачам розгортати копії даних на кількох

хостах і терпіти відмови на будь-якому хості. Тому *Cassandra* є популярним вибором для стартапів.

*Redis* – це система зберігання даних у вигляді структури даних в оперативній пам'яті, яка виступає як база даних, кеш та брокер повідомлень. Зберігаючи дані в пам'яті, вона забезпечує дуже високу продуктивність та особливо підходить для реального часу, наприклад, для банківських додатків. Дані в *Redis* зберігаються у формі пар ключ-значення, схожих на хеш-карту *Java* або словник *Python*. *Redis* підходить для використання в ситуаціях, де важливий низький рівень затримки доступу до даних.

### 2.2.3. Хмарні бази даних

Хмарні бази даних – це бази даних, що розміщені та управляються провайдерами хмарних послуг, такими як *Amazon Web Services (AWS), Google Cloud Platform* (*GCP*) або *Microsoft Azure*. Вони пропонують різноманітні переваги, такі як масштабованість, надійність, безпека та ефективність вартості, а також доступ до різноманітних функцій та інструментів. Хмарні бази даних можуть бути як реляційними, так і *NoSQL*, залежно від сервісу та моделі даних. Вони ідеально підходять для веб-застосунків, які потребують використання переваг хмарних обчислень, таких як гнучкість, рухливість та інновації. Деякі з найпопулярніших хмарних баз даних для веб-розробки - це *Amazon RDS, Google Cloud SQL, Azure SQL Database* та *Firebase*.

*Amazon RDS* – це послуга управління реляційними базами даних в хмарі, яка надає можливість запуску, управління та масштабування реляційних баз даних у середовищі *AWS*. Вона підтримує різні типи реляційних баз даних, такі як *MySQL, PostgreSQL, Oracle, SQL Server* та *Amazon Aurora*. *Amazon RDS* надає автоматизовані можливості резервного копіювання, масштабування й автоматичного підтримування баз даних. Це дозволяє розробникам зосередитися на розробці програмного забезпечення, не турбуючись про адміністрування баз даних.

*Google Cloud SQL* – це послуга керування реляційними базами даних у хмарному середовищі *Google Cloud Platform (GCP)*. Ця послуга підтримує бази даних *MySQL, PostgreSQL* та *SQL Server*. *Google Cloud SQL* надає розширені можливості для резервного копіювання, масштабування, шифрування даних, а також забезпечує високу доступність. Вона інтегрується з іншими сервісами *GCP*, що дає змогу легко поєднувати роботу з базами даних із іншими хмарними сервісами *Google*.

*Azure SQL Database* – це керована служба бази даних, яка пропонується у хмарному середовищі *Microsoft Azure*. Вона спеціалізується на роботі з *Microsoft SQL Server* і надає можливості автоматизованого управління базами даних, включаючи автоматичне резервне копіювання, масштабування, високу доступність та вбудовану безпеку. *Azure SQL Database* дозволяє швидко розгортати й масштабувати бази даних, забезпечуючи високу продуктивність та надійність.

### 2.2.4. Графові бази даних

Графові бази даних – це тип *NoSQL* баз даних, які використовують концепцію вузлів та зв'язків для представлення та зберігання даних у вигляді мережі взаємопов'язаних сутностей та відносин. Вони використовують мови запитів до графів, такі як *Cypher* або *Gremlin*, для переходу та маніпулювання даними на основі їх зв'язків та шаблонів. Графові бази даних ідеально підходять для веб-додатків, які потребують моделювання та аналізу високорівневих, складних та динамічних даних, таких як соціальні мережі, системи рекомендацій або бази знань. Деякі з найпопулярніших графових баз даних для веб-розробки це *Neo4j*, *OrientDB*, *ArangoDB* та *Dgraph*.

*Neo4j* – це популярна графова база даних, побудована на основі вузлів, зв'язків і властивостей. Вона має гнучкий та потужний інтерфейс, який дозволяє працювати з високо-пов'язаними даними. Основна перевага *Neo4j* полягає в швидкості та ефективності роботи з великими графами даних. *Neo4j* пропонує мову запитів

*Cypher*, спеціально розроблену для взаємодії з графовою моделлю, що робить роботу з даними більш зручною.

*OrientDB* – це гібридна графова база даних, яка поєднує в собі можливості графових, об'єктно-орієнтованих та документних баз даних. Вона пропонує високу продуктивність та гнучкість в роботі з даними. *OrientDB* підтримує *SQL*-подібну мову запитів, а також можливості графової моделі, що робить його відмінним вибором для проектів, які потребують об'єднання різних моделей даних.

*ArangoDB* – це багатомодельна база даних, яка підтримує графову, ключ-значення та документну моделі. Ця база даних відома своєю гнучкістю, високою продуктивністю та зручними можливостями для розробки. Вона пропонує *AQL* (*ArangoDB Query Language*) для роботи з даними, що дозволяє виконувати складні операції та запити до графів, документів і ключ-значення.

*Dgraph* – це розподілена графова база даних з відкритим вихідним кодом, яка спеціалізується на розподіленому зберіганні графів даних. Вона розроблена для обробки великих об'ємів даних та має гнучкий графовий модель та вбудований підтримку мови запитів *GraphQL*. *Dgraph* відомий своєю швидкодією та здатністю до горизонтального масштабування для роботи з великими та змінними графами даних.

### 2.2.5. *In-memory databases*

*In-memory databases* – це бази даних, які зберігають і обробляють дані виключно у основній пам'яті сервера, а не на диску. Це дозволяє досягти швидшої продуктивності, меншої затримки та вищої пропускної здатності, а також підтримувати операції та транзакції в реальному часі. Бази даних у пам'яті корисні для веб-застосунків, які потребують обробки даних високої швидкості, таких як кешування, обмін повідомленнями, стрімінг або геймінг. Деякі з найпопулярніших баз даних у пам'яті для розробки веб-застосунків це *Redis*, *Memcached*, *VoltDB* і *Tarantool*.

*Memcached* – це розподілена система кешування пам'яті, яка використовується для зберігання парами ключ-значення у пам'яті. Вона призначена для прискорення доступу до даних, шляхом зберігання у пам'яті попередньо обчислених або отриманих результатів запитів до джерел даних. *Memcached* допомагає зменшити час доступу до даних та витрати на завантаження сервера, розподіляючи завдання кешування між багатьма серверами. Він зазвичай використовується для прискорення веб-сторінок та зниження навантаження на бази даних, шляхом кешування запитів.

*VoltDB* – це інтерактивна, реляційна база даних, спроектована для реального часу та швидкості. Вона здатна обробляти великі обсяги транзакцій у реальному часі, використовуючи масштабовані архітектури. *VoltDB* має спрощену структуру для високошвидкісного виконання і обробки даних у реальному часі. Вона часто використовується для розробки високонавантажених веб-застосунків, які потребують реактивності та здатності обробляти великі об'єми транзакцій.

*Tarantool* – це високошвидкісна інтерактивна база даних, яка поєднує можливості бази даних та сервера застосунків в одному рішенні. Вона має розширені можливості управління даними, включаючи підтримку *Lua* для створення збережених процедур, використання відомих механізмів кешування та підтримку реплікації для забезпечення високої доступності. *Tarantool* добре підходить для високонавантажених веб-застосунків, де важлива швидкодія, а також для використання в системах, що вимагають швидкості, таких як ігрові платформи та додатки, що працюють з реальним часом.

Для програмного комплексу обліку роботи працівників буде використовуватися *PostgreSQL*. Він здатний працювати з великими обсягами даних, що особливо важливо для систем обліку роботи працівників. Також він надійний, масштабований та безпечний. Його різноманітність типів даних, підтримка *SQL* та розширених операцій дозволяють ефективно вести облік роботи працівників. Активна спільнота розробників забезпечує підтримку та оновлення.

### 2.3. Вибір фреймворків

*Python* – це потужна та універсальна мова програмування, яка здобула величезну популярність у світі веб-розробки. Існую безліч фреймворків для створення веб-додатків. Двома найпоширенішими веб-фреймворками *Python* є *Flask* і *Django* (рис. 2.5). Хоча *Flask* і *Django* використовуються для створення веб-додатків, вони мають різні філософії, варіанти використання та функції.

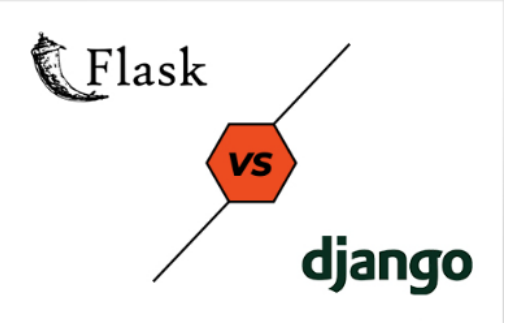


Рис. 2.5. Логотипи *Flask* і *Django*

### 2.3.1. *Flask vs Django*

*Flask* і *Django* – це обидва відкриті веб-фреймворки *Python*, які спрощують процес створення веб-додатків. Вони надають інструменти та бібліотеки, які полегшують роботу з такими загальними завданнями, як маршрутизація, обробка *HTTP*-запитів та відповідей, взаємодія з базами даних та відображення шаблонів. Однак вони відрізняються своїми філософіями проектування та підходять для різних типів проектів.

Філософія дизайну

*Flask*, створений Арміном Ронахером, є мікро-фреймворком, який слідує філософії "мікро", що означає, що він надає лише основи для створення веб-додатків без накладання якихось конкретних правил структури додатка. Це дає розробникам свободу вибору власних інструментів та бібліотек для таких завдань, як робота з шаблонами, доступ до баз даних та обробка форм. *Flask* часто називають "мікро" фреймворком через його мінімалістичність та легкість, що робить його ідеальним для невеликих та середніх проектів або для розробників, які віддають перевагу гнучкості та вільності у структурі проекту.

З іншого боку, *Django*, створений Адріаном Головатим та Саймоном Вілісоном, є повноцінним веб-фреймворком, що слідує філософії "все включено". *Django* включає все необхідне для створення складних веб-додатків, включаючи *ORM* (об'єктно-реляційний мапер) для взаємодії з базами даних, систему шаблонів, адміністративний інтерфейс та багато інших функцій "з коробки". Він часто називається "повноцінним" фреймворком через надання широкого набору інструментів та бібліотек для створення великомасштабних веб-додатків з чіткою та структурованою архітектурою. *Django* підходить для складних проектів, які потребують потужних можливостей та слідують певній структурі.

Гнучкість та свобода

Однією з основних переваг *Flask* є його гнучкість та свобода. *Flask* дозволяє розробникам повністю контролювати структуру проекту, компоненти та бібліотеки. Розробники можуть вибирати власний *ORM*, систему шаблонів та інші інструменти, що дозволяє їм використовувати бібліотеки, з якими вони знайомі або віддають перевагу. *Flask* ідеально підходить для розробників, які бажають мати можливість будувати свій додаток з нуля та мають повний контроль над кожним аспектом свого проекту.

З іншого боку, *Django* слідує попередньо визначеною структурою і включає власний набір інструментів та бібліотек, що може обмежити гнучкість та свободу для розробників, які віддають перевагу власним виборам бібліотек або мають іншу структуру проекту на увазі. Філософія "все включено" *Django* може бути корисною для розробників, які хочуть чітку та структуровану архітектуру та не хочуть витрачати час на пошук та інтеграцію різних бібліотек. Однак це може бути не підходящим для тих, хто віддає перевагу можливості вибирати власні інструменти та компоненти.

Крива навчання

Крива навчання для *Flask* зазвичай вважається відносно низькою. Мінімалістичний та простий підхід *Flask* робить його легким для розуміння та швидкого початку роботи для розробників. Його проста структура проекту та відсутність строгих конвенцій дозволяють розробникам плавно вивчити його та

легко розуміти потік застосунку. Документація *Flask* також обширна та добре документована, що робить його відмінним вибором для початківців або розробників, які нові у веб-розробці.

З іншого боку, крива навчання для *Django* може бути складнішою через його повний набір функцій та строгу структуру проекту. *Django* слідує архітектурі *MVT*, що може потребувати від розробників розуміння того, як різні компоненти, такі як моделі, види та шаблони, взаємодіють один з одним. *ORM*, система шаблонів та адміністративний інтерфейс *Django* також можуть потребувати часу для вивчення та розуміння. Однак у *Django* є відмінна документація та велике співтовариство, яке може надати підтримку розробникам, які нові у роботі з фреймворком.

Швидкість розробки

Мінімалістичний підхід *Flask* і його гнучкість дозволяють розробникам швидше створювати програми. Оскільки *Flask* не накладає жодного певного способу структурування програми і дозволяє розробникам обирати свої власні інструменти та бібліотеки, вони можуть швидко адаптуватися до свого улюбленого робочого процесу та створювати програму так, як вони цього прагнуть. Проста структура проекту *Flask* і відсутність жорстких конвенцій також полегшують ітерацію та швидке внесення змін.

З іншого боку, широкий набір функцій у *Django* та жорстка структура проекту можуть вимагати більше часу на налаштування, що може вплинути на швидкість розробки, особливо для невеликих проектів чи прототипів. Однак вбудовані можливості *Django*, такі як *ORM*, движок шаблонів та адміністративний інтерфейс, можуть заощадити час розробки в подальшому для масштабних додатків, які потребують складних функцій.

Масштабованість

Як *Flask*, так і *Django* здатні створювати масштабовані веб-додатки, але вони мають різний підхід. Мінімалістичний і легкий підхід *Flask* робить його придатним для невеликих і середніх проектів, де розробники мають повний контроль над структурою проекту та його компонентами. *Flask* дозволяє розробникам вибирати

власні інструменти та бібліотеки, що дає їм можливість оптимізувати додаток для масштабованості відповідно до конкретних вимог проекту.

З іншого боку, філософія "все включено" *Django* та попередньо визначена структура проекту роблять його придатним для масштабних додатків, які вимагають складних функцій та мають чітку та структуровану архітектуру. Вбудовані можливості *Django*, такі як *ORM*, движок шаблонів та адміністративний інтерфейс, надають міцну основу для створення масштабованих додатків. У *Django* також є потужна екосистема сторонніх бібліотек та плагінів, які можуть подальше підвищити масштабованість додатка.

Спільнота та Екосистема

Як *Flask*, так і *Django* мають великі та активні спільноти, але вони відрізняються своєю екосистемою та підтримкою спільноти. У *Flask* менша спільнота порівняно з *Django*, але він відомий своєю простотою, гнучкістю та простотою використання. Спільнота *Flask* дуже активна, з регулярними оновленнями, обширною документацією та активними форумами підтримки. У *Flask* також зростає екосистема сторонніх бібліотек та плагінів, хоча вона може бути не такою повною, як екосистема *Django*.

З іншого боку, у *Django* велика та досвідчена спільнота з розгалуженою екосистемою сторонніх бібліотек, плагінів та повторно використовуваних додатків. Спільнота *Django* відома своєю міцністю, надійністю та стабільністю. У *Django* є обширна документація, розсилки, форуми та сильна система підтримки, що робить його надійним вибором для масштабних додатків або проектів, які потребують багато ресурсів та підтримки спільноти.

Безпека

Як *Flask*, так і *Django* надають пріоритет безпеці, і обидва фреймворки мають вбудовані можливості та найкращі практики для допомоги розробникам створювати безпечні веб-додатки. *Flask* використовує мінімалістичний підхід, що означає, що за замовчуванням він не включає багато вбудованих засобів безпеки. Однак *Flask* надає міцну основу для створення безпечних додатків, і розробники мають можливість вибирати власні інструменти та бібліотеки для впровадження заходів безпеки.

З іншого боку, *Django* має багато вбудованих засобів безпеки, таких як захист від міжсайтових скриптів (*XSS*), міжсайтових запитів підробки (*CSRF*) та атак внесення *SQL*-запитів. *Django* також має вбудований адміністративний інтерфейс з міцними механізмами аутентифікації та авторизації, що полегшує розробникам створення безпечних веб-додатків без необхідності реалізації цих заходів безпеки з нуля.

Розгортання та Хостинг

Як *Flask*, так і *Django* можна розгортати та хостити на різних платформах і веб-серверах. Легкий та гнучкий характер *Flask* дозволяє легко розгортати його на різних платформах, таких як *Heroku*, *AWS*, *Google Cloud* або будь-який інший хостинговий сервіс, який підтримує *Python*. *Flask* дозволяє розробникам обирати власну стратегію розгортання та настроювати процес розгортання відповідно до їх конкретних вимог.

*Django*, з іншого боку, має вбудовану підтримку для розгортання з використанням власного серверу розробки, а також популярних веб-серверів, таких як *Apache*, *Nginx* та *Gunicorn*. *Django* також має інтеграцію з популярними хостинговими сервісами, такими як *Heroku, AWS* і *Google Cloud*, що полегшує розгортання та хостинг додатків *Django* на цих платформах.

*Flask* та *Django* – це обидва популярні веб-фреймворки *Python*, які задовольняють різні потреби та вподобання розробників. Для розробки прорамного комплексу обліку роботи працівників логістичної авіакомпанії буде використовуватись *Django*, адеже він є комплексним і функціонально насиченим фреймворком із жорсткою структурою проекту, який підходить для масштабних додатків або проектів, що потребують складних функціональностей. *Django* використовує архітектуру *MVT* та має вбудовані можливості, такі як *ORM*, движок шаблонів та адміністративний інтерфейс, що робить його надійним вибором для створення масштабованих веб-додатків. У *Django* велика та досвідчена спільнота з обширною документацією та підтримкою, що робить його популярним серед підприємств та промисловості.

### 2.3.2. *Bootstrap*

Для розробки інтерфейсу програмного комплексу буде використовуватись фреймворк *Bootstrap* (рис 2.6).

*Bootstrap* – це відкритий веб-фреймворк. Цей фреймворк містить набір інструментів, компонентів, шаблонів та стилів *CSS* і *JavaScript*, спрямованих на швидку та просту розробку адаптивних та стильних веб-інтерфейсів.

Особливості фреймоворку:

Адаптивність

Система сітки *Bootstrap* базується на концепції "*flexbox*", яка є новою моделлю макетування *CSS*, що спрощує створення адаптивних макетів. Система сітки розділяє екран на 12 колонок, які можна змінювати розмір та перегруповувати, щоб відповідати потребам будь-якого розміру екрана. Це дозволяє створювати веб-сайти, які виглядають добре на всіх пристроях, від мобільних телефонів до настільних комп'ютерів.

Рис. 2.6. Логотип *Bootstrap*

Мобільна орієнтованість

*Bootstrap* розроблений з урахуванням мобільних додатків. Система сітки оптимізована для мобільних браузерів, і *Bootstrap* включає кілька компонентів, спеціально призначених для мобільних пристроїв, таких як мобільна навігаційна панель та мобільне випадаюче меню. Це полегшує створення веб-сайтів, які є як адаптивними, так і дружніми до мобільних пристроїв.

Консистентність

Готові компоненти, адаптивна сітка та *JavaScript*-плагіни *Bootstrap* значно прискорюють процес розробки. Розробники можуть швидко створювати професійно вигляд сайтів та веб-додатків, зменшуючи час та зусилля, витрачені на розробку. *Bootstrap* забезпечує послідовність та стандартизований дизайн. Ця єдність у стилізації та макетуванні по всьому проекту допомагає зберігати єдиний та професійний вигляд, навіть коли кілька членів команди працюють над різними частинами проекту.

Налаштування

*Bootstrap* має великі можливості налаштування. Ви можете змінювати кольори, шрифти та інші аспекти фреймворку, щоб відповідати вашому власному брендингу. Крім того, *Bootstrap* включає кілька плагінів, які можна використовувати для додавання додаткової функціональності до вашого веб-сайту. Це дозволяє створювати унікальні веб-сайти, які виглядають так, як ви хочете.

Швидкість

*Bootstrap* є легким та швидким. Він не додає багато зайвого коду на ваш веб-сайт, що може допомогти покращити його швидкість завантаження. Це важливо як для користувачів, так і для оптимізації для пошукових систем.

Спільнота

У *Bootstrap* велика та активна спільнота розробників. Це означає, що є багато ресурсів, доступних для вас, щоб навчитися використовувати фреймворк, і також існує кілька готових шаблонів та компонентів, які можна використовувати для швидкого початку. Це полегшує отримання допомоги, коли вам це потрібно, і також означає, що завжди є нові та інноваційні способи використання *Bootstrap*.

### 2.4. Вибір середовища розробки

Розробка веб-застосунків у сучасному світі вимагає вдосконалених інструментів та технологій для забезпечення продуктивної та ефективної роботи. Використання сучасних фреймворків та інструментів стає ключовим для успішної

створення програмного забезпечення. У цьому контексті використання *Django, Bootstrap* та *PostgreSQL* виявляється одним з найкращих варантів, оскільки ці технології надають потужні інструменти для розробки стабільних, зручних та естетичних веб-застосунків.

Для розробки програмного комплексу використовується *PyCharm* (рис. 2.7), так як воно відповідає основним вимогам:

- доступність;
- зручний для використання та розуміння інтерфейс;
- сумісна з усіма версіями операційної системи *Windows*;
- наявність необхідних для роботи інструментів;
- надає спеціалізовані інструменти для розробки веб-застосунків на *Django*, включаючи автодоповнення для фреймворку, підказки та швидкий доступ до функцій *Django*;
- має широкий вибір плагінів, які можна легко встановити для розширення його функціональності за потребою;
- має велику спільноту користувачів, що дозволяє легко знаходити відповіді на питання та отримувати підтримку в разі потреби.

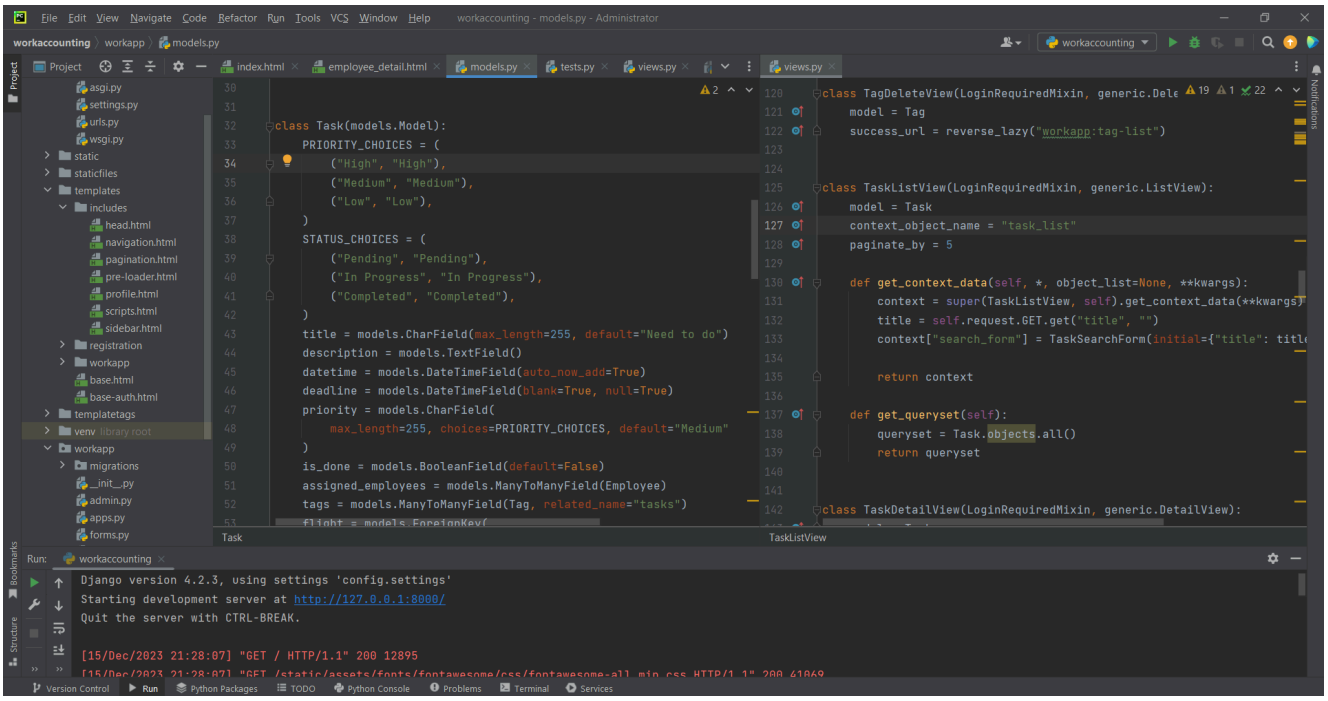


Рис. 2.7. Зовнішній вигляд *IDE PyCharm*

Налаштування та тестування веб-застосунків вимагає використання веб-браузера, який відповідає певним вимогам для забезпечення ефективності розробки та перевірки функціоналу.

*Chrome*, як один з основних веб-браузерів, має ряд характеристик, які роблять його відмінним вибором для розробки веб-застосунків:

- активно оновлюється, щоб підтримувати останні версії стандартів *HTML, CSS* та *JavaScript*, що дозволяє переконатися, що веб-застосунки відображаються коректно та працюють на різних пристроях;
- *Chrome Developer Tools* надає широкий спектр інструментів, таких як консоль, інспектор елементів, аналізатор продуктивності, телефоний формат (рис. 2. 8) та інші, що допомагають відлагоджувати, вирішувати проблеми та перевіряти функціонал веб-застосунків;
- доступний на різних операційних системах (*Windows, macOS, Linux*), що дозволяє перевірити, як веб-застосунок відображається на різних платформах;
- надає підтримку захищеного з'єднання через протокол *HTTPS*, що дозволяє тестувати веб-застосунки, що використовують шифрування даних, забезпечуючи їхню безпеку.

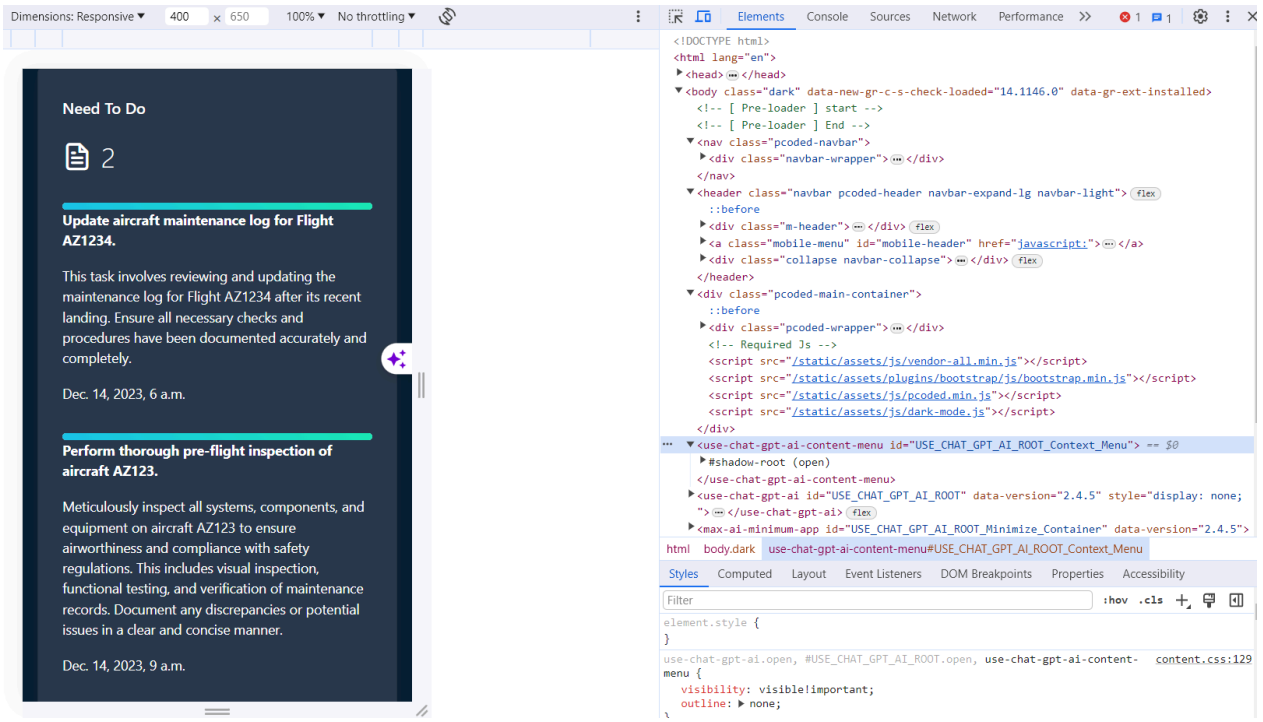


Рис. 2.8. *Chrome Developer Tools*

## 2.5. Висновки до розділу

У даному розділі було проаналізовано декілька мов програмування таких як *JavaScript, Java, Ruby* та *Python*. Хоча *Java* відома своєю швидкістю та безпекою, її використання може бути ефективним для більших проектів та систем, що потребують високого рівня захисту. Водночас, *JavaScript*, який підтримується всіма основними веб-браузерами, є універсальним вибором для створення інтерактивних веб-елементів та має велику кількість фреймворків, які полегшують процес розробки. Також, *Ruby*, особливо у фреймворку *Ruby* on *Rails*, використовується як для фронтенду, так і для бекенду, і є зручним вибором для розробки веб-додатків.

Однак у контексті логістики авіакомпанії, *Python* стає гарним вибором завдяки своїй дружній для користувача природі, легкості розуміння коду та здатності інтегрувати штучний інтелект та машинне навчання для оптимізації бізнес-процесів,

що може бути користним для подальшого розвитку комплексу обліку роботи працівників.

Мова програмування *Python* є відмінним вибором для логістичної авіакомпанії у контексті обліку роботи працівників. Її простота, легкість вивчення та зрозумілий синтаксис роблять її особливо зручною та корисною для розробки програм, спрямованих на управління персоналом, оптимізацію складських процесів та інших логістичних аспектів авіакомпанії.

Також у даному розділі проаналізовано бази даних і в результаті обрано реляційну базу даних *PostgreSQL*.

*PostgreSQL* підтримує широкий спектр типів даних, що дозволяє зберігати та обробляти різноманітні дані, включаючи текст, числа, зображення, *JSON*, геоданих тощо. Це корисно при зберіганні різних типів інформації про працівників та рейсів.

*PostgreSQL* використовує мову запитів *SQL* для отримання даних. Він також має ряд додаткових можливостей, таких як використання функцій, підзапитів, віконних функцій та інших розширених функцій, які полегшують аналітичні операції та формування звітів.

*PostgreSQL* має різноманітні інструменти для забезпечення безпеки даних, включаючи ролі користувачів, рівні доступу, шифрування даних тощо. Це дозволяє забезпечити захист конфіденційності та цілісності даних про працівників.

*Django* – популярний веб-фреймворк *Python*, обраний для розробки програмного комплексу обліку роботи працівників логістичної авіакомпанії. Його комплексність, структурована архітектура *MVT*, вбудовані можливості, такі як *ORM*, шаблонний движок та адміністративний інтерфейс, роблять *Django* ідеальним для створення масштабованих веб-додатків.

Для розробки інтерфейсу програмного комплексу обрано *Bootstrap*. Він надає зручні інструменти, компоненти та стилі *CSS* та *JavaScript* для простого та швидкого створення адаптивних веб-інтерфейсів. Заснований на концепції "*flexbox*", він пропонує гнучку систему сітки, що дозволяє створювати макети, які підлаштовуються під різні розміри екранів. Маючи велику кількість готових компонентів, *Bootstrap* значно прискорює розробку, забезпечуючи консистентний та

стильний дизайн. Також фреймворк відрізняється можливістю налаштування під конкретні потреби проекту, а його легкість і швидкість дозволяють покращити продуктивність. При цьому, наявність великої та активної спільноти розробників робить *Bootstrap* популярним та допомагає знайти відповіді на питання та готові рішення для початку роботи.

Використання *PyCharm* як інтегрованого середовища розробки надає зручність у використанні та розумінні інтерфейсу, підтримку *Django* та доступ до необхідних інструментів для створення веб-застосунків. Використання браузера *Chrome* для тестування забезпечує підтримку останніх стандартів веб-технологій та набір інструментів розробника для ефективної роботи над функціоналом веб-застосунків.

В заключенні, обрані технології – *Django*, *Bootstrap* та *PostgreSQL* – представляють собою потужну та синергетичну комбінацію для розробки програмного комплексу обліку роботи працівників логістичної авіакомпанії. *Django* надає стійкість та безпеку на бекенді, забезпечуючи високий рівень функціональності та стабільність у роботі. *Bootstrap*, у свою чергу, забезпечує динамічну та інтуїтивно зрозумілу користувацьку інтерфейсу на фронтенді, дозволяючи зберегти легкість використання та високу продуктивність. *PostgreSQL*, як потужна реляційна база даних, гарантує ефективне зберігання та обробку даних, а також сприяє розширюваності та надійності системи. Такий вибір технологій враховує потреби у безпеці, продуктивності та зручності для успішної розробки та експлуатації програмного комплексу для логістичної авіакомпанії.

# РОЗДІЛ 3
# ПРОГРАМНА РЕАЛІЗАЦІЯ КОМПЛЕКСУ

## 3.1. Вимоги до прорамного комплексу обліку роботи

### 3.1.1 Функціональні вимоги

Функціональні вимоги – це формулювання того, як система повинна поводитися, описують, що саме має робити система, щоб задовольнити потреби або очікування користувача. Функціональні вимоги можна розглядати як функції, які виявляє користувач. Вони відрізняються від нефункціональних вимог, які визначають, як система повинна працювати всередині.

Функціональні вимоги програмного комплексу обліку роботи працівників логістичної авіакомпанії:

- авторизовуватись в систему;
- додавати та зберігати персональних даних працівників;
- редагувати інформації про працівника;
- створювати та відслідковувати процесу виконання задач;
- вносити зміни у завдання;
- видаляти завдання;
- додавати та відстежувати інформації про рейси;
- редагувати інформації про рейтси;
- видаляти рейсів;
- створювати графік змін та робочих годин працівників;
- оцінювати діяльність працівників;
- створювати та управляти тегами для завдань для подальшої категоризації та організації робочих процесів;
- вийти з системи.

### 3.1.2. Системні вимоги

Розроблений веб-додаток для обліку роботи працівників логістичної авіакомпанії має певні вимоги до веб-браузерів. Браузери надають користувачам доступ до Інтернету, відвідування веб-сайтів, перегляд веб-сторінок та інші функції.

Оскільки веб-браузери постійно оновлюються, важливо мати комп'ютер з достатніми програмними та апаратними ресурсами, щоб підтримувати їхню роботу.

Мінімальні вимоги до програмного забезпечення:

- Операційна система: *Windows 10, 11, Linux (Ubuntu, Mint, Debian, OpenSuse)* або *Mac OS*
- Процесор: двоядерний процесор з частотою не менше 1,6 *ГГц*
- Оперативна пам'ять: 4 *ГБ*
- Дисковий простір: 10 *ГБ*
- Графічна карта: сумісна з *OpenGL* 2.0

Бажані вимоги до програмного забезпечення:

- Операційна система: остання версія *Windows, Linux* або *Mac OS*
- Процесор: чотириядерний процесор з частотою не менше 2,4 *ГГц*
- Оперативна пам'ять: 8 *ГБ*
- Дисковий простір: 20 *ГБ*
- Графічна карта: сумісна з *OpenGL* 3.0

Мінімальні вимоги до апаратного забезпечення:

- Монітор: роздільна здатність не менше 1024×768 пікселів
- Клавіатура
- Миша або тачпад

Для забезпечення найкращої роботи веб-браузера рекомендується використовувати комп'ютер з такими характеристиками:

- Операційна система: остання версія *Windows, Linux* або *Mac OS*
- Процесор: чотириядерний процесор з частотою не менше 2,4 *ГГц*
- Оперативна пам'ять: 16 *ГБ*
- Дисковий простір: 50 *ГБ*

− Графічна карта: сумісна з *OpenGL* 4.0

## 3.2. Структура програмного комплексу

### 3.2.1. Структура бази даних

При проектуванні комплексу обліку роботи працівників створено структуру бази даних (рис.3.1).

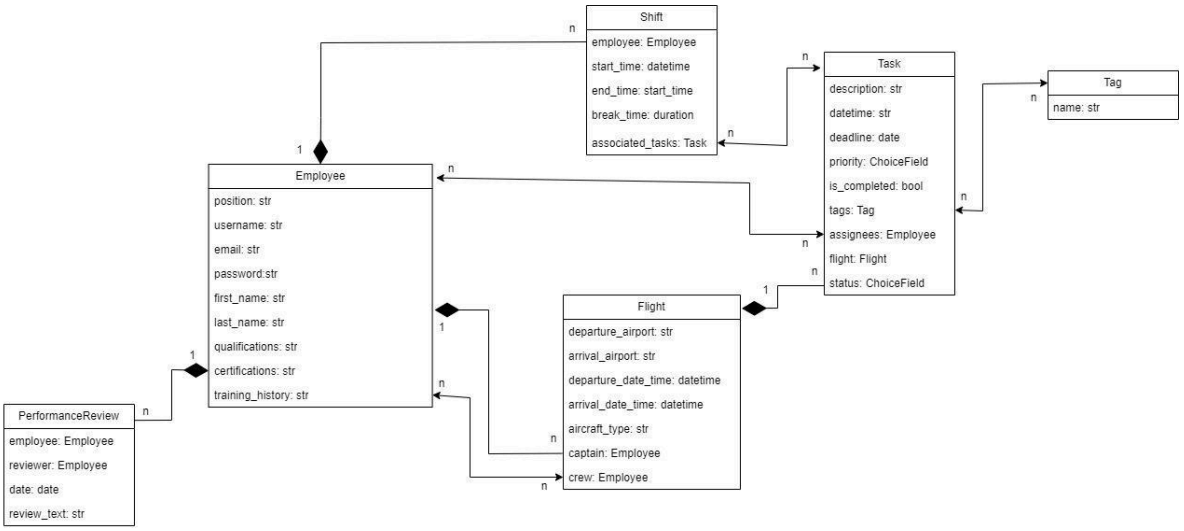


Рис. 3.1. Структура зв'язків бази даних

Структура бази даних включає в себе наступні основні таблиці:

1. Таблиця «*Employees»* (Працівники):

− Унікальний ідентифікатор (*ID*) для кожного працівника.

− Ім'я, прізвище, пароль та контактна інформація (електронна пошта).

− Деталі щодо посади, кваліфікації, сертифікати та історія навчання.

2. Таблиця «*Tasks*» (Завдання):

− Унікальний ідентифікатор завдання.

− Опис завдання та його статус (викоанти, в роботі, виконано), прироритет, дедлайн завдання.

− Теги для класифікації завдань, список співробітників, відповідальних за виконання завдання, рейс, до якого пов'язано завдання.

3. Таблиця «*Shifts*» (Зміни):

- Інформація про графік змін працівників, включаючи початок, кінець та можливу перерву.
- Завдання які необхідно виконати у дану зміну.

4. Таблиця «*Performance Reviews*» (Оцінка результативності):

- Інформація щодо оцінки роботи працівників, включаючи дату огляду та відгук.

5. Таблиця «*Flight*» (Рейси):

- Унікальний ідентифікатор рейсу.
- Інформація про аеропорти вильоту та прибуття, дату та час вильоту, прильоту, тип літака та екіпаж.

6. Таблиця «*Tag*» (Завдання):

- Унікальний ідентифікатор завдання.
- Тег для подальшої класифікації завдання.

Ця структура бази даних забезпечує ефективне управління роботою працівників логістичної авіакомпанії. Вона дозволяє зберігати та обробляти інформацію, необхідну для виконання наступних основних функцій: управління персоналом, управління завданнями.

### 3.2.2. Структура програми

Загальна структура програми має такий вигляд:

1. *Frontend* (Клієнтська частина):

- *HTML/CSS/JavaScript*: Клієнтська частина веб-застосунку буде включати розмітку (*HTML*), стилізацію (*CSS*) та взаємодію (*JavaScript*) з користувачем.
- *Bootstrap*: Використання *Bootstrap* для швидкого та зручного створення стильового та адаптивного інтерфейсу.

2. *Backend* (Серверна частина):

- *Django Framework*: Всі серверні операції, обробка запитів, маршрутизація, робота з базою даних відбувається у фреймворку *Django*.

− *Python: Django* базується на *Python*, тому програмний код серверної частини веб-застосунку буде написаний мовою *Python*.

3. База даних:

− *PostgreSQL*: Використання *PostgreSQL* в якості реляційної бази даних для зберігання даних, які використовуються в застосунку.

Така структура показує основні складові програмного комплексу.

Проект, базується на фреймворку *Django*, який використовує концепцію *MVT* (*Model-View-Template*) для організації його архітектури. (рис. 3.2)

*Django*, відомий своєю потужністю та зручністю в розробці веб-додатків, заснованих на патерні *MVT*, пропонує високорівневий підхід до будівництва програмного забезпечення. Модель-Представлення-Шаблон (*MVT*) – це архітектурний шаблон який розділяє веб-додаток на три основних компоненти: модель, представлення і шаблон. Модель відповідає за дані, представлення обробляє логіку відображення та взаємодії з користувачем, а шаблон відповідає за відображення даних. Використання *Django* у поєднанні з *MVT* дозволяє створювати зручні, модульні та масштабовані веб-додатки, спрощуючи процес розробки та підтримки проєкту.

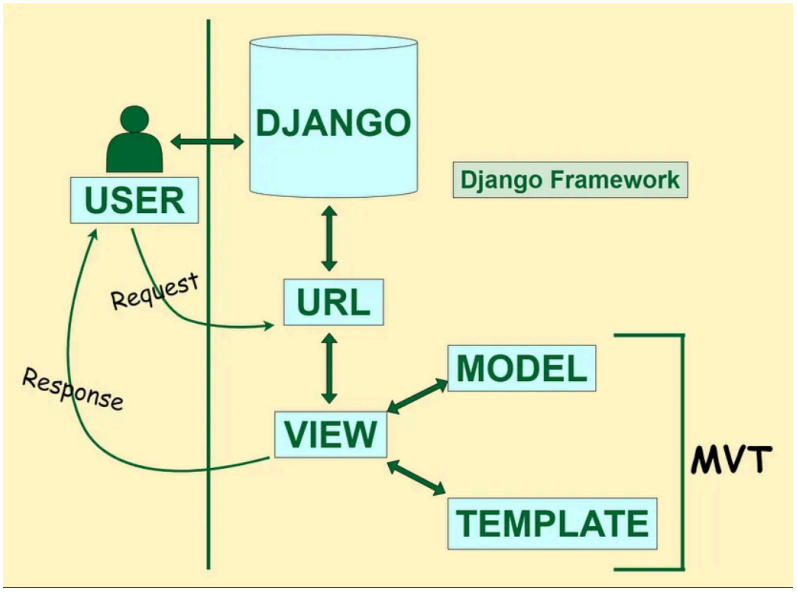


Рис. 3.2. Архітектурний шаблон веб-додатку

### 3.3. Розробка комплексу обліку працівників

Для розробки програмного комплексу обліку роботи працівників логістично авіакомпанії використано середовище програмування *PyCharm* та мова *Python*.

Спочатку створено проєкт використовуючи команду *django-admin startproject.*

Розроблено структуру програми (рис. 3.3)

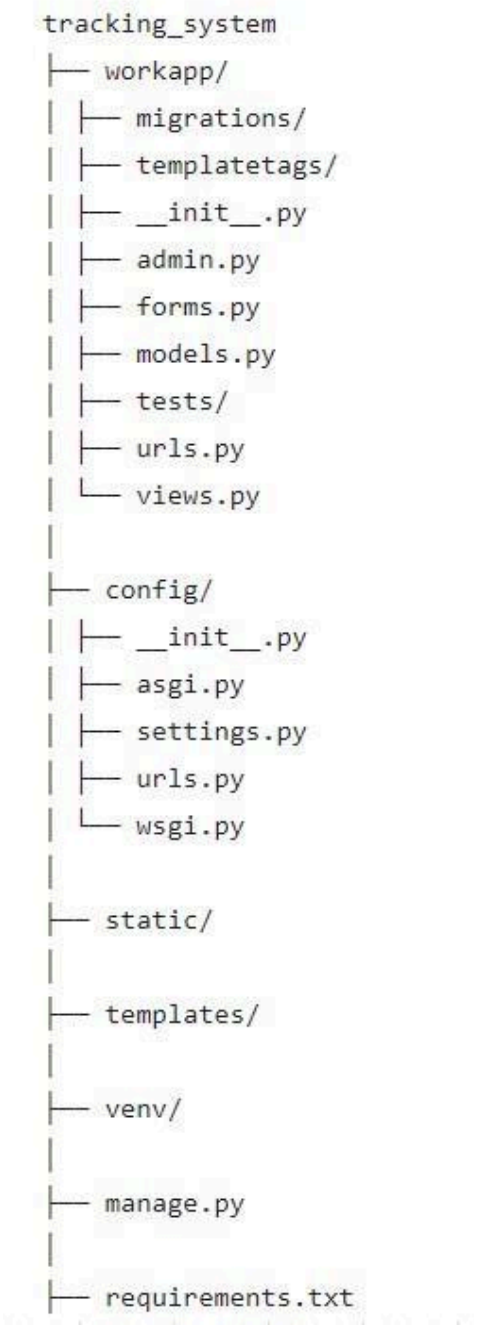


Рис. 3.3. Організація прогармного коду

Встановлено необхідні модулі та розширення:

1. *Django* – веб-фреймворк *Python* для швидкої розробки веб-додатків зі вбудованим *ORM*, шаблонами та адмін-панеллю.
2. *django-bootstrap4* – модуль для *Django*, який надає можливість використовувати *Bootstrap 4* у веб-сторінках *Django*.
3. *django-crispy-forms* – розширення для *Django*, яке дозволяє легко створювати стильні форми

Налаштовано основні параметри проєкту в файлі налаштувань *settings.py*.

Визначено структуру бази даних за допомогою *Django* моделей, використовуєчи *ORM Django* для зв'язків між об'єктами та моделями бази даних. Це включає визначення класів моделей, які представляють сутності системи та їх взаємозв'язки.

Список класів та їх опис:

1. *Employee* (Працівник):

− Вміщує дані про працівника, такі як посада, ім'я користувача, ім'я, прізвище, електронна пошта, пароль, номер телефону, кваліфікація, сертифікації, історія навчання.

2. *Tag* (Мітка):

− Представляє окрему мітку, яка може використовуватися для класифікації завдань.

3. *Task* (Завдання):

− Описує конкретне завдання з назвою, описом, датою та часом створення, дедлайном, пріоритетом, статусом, прив'язаними працівниками та мітками.

```
class Task(models.Model):
    PRIORITY_CHOICES = (
        ("High", "High"),
        ("Medium", "Medium"),
```

```
        ("Low", "Low"),
    )
    STATUS_CHOICES = (
        ("Pending", "Pending"),
        ("In Progress", "In Progress"),
        ("Completed", "Completed"),
    )
    title = models.CharField(max_length=255, default="Need to do")
    description = models.TextField()
    datetime = models.DateTimeField(auto_now_add=True)
    deadline = models.DateTimeField(blank=True, null=True)
    priority = models.CharField(
        max_length=255, choices=PRIORITY_CHOICES, default="Medium"
    )
    is_done = models.BooleanField(default=False)
    assigned_employees = models.ManyToManyField(Employee)
    tags = models.ManyToManyField(Tag, related_name="tasks")
    flight = models.ForeignKey(
        "Flight", on_delete=models.CASCADE, null=True, blank=True)
    status = models.CharField(max_length=255, choices=STATUS_CHOICES,
default="Pending")
    class Meta:
        ordering = ["datetime"]
```

*def __str__(self):*

*return f"{self.description} (deadline: {self.deadline})"*

4. *Flight* (Рейс):

− Містить інформацію про політ, таку як аеропорт відправлення та прибуття, дату та час відправлення та прибуття, тип повітряного судна, капітана та екіпаж.

5. *Shift* (Зміна):

− Представляє окрему зміну працівника з початковим та кінцевим часом, часом перерви та пов'язаними завданнями.

6. *PerformanceReview* (Оцінка ефективності):

− Зберігає відгуки про ефективність працівників, включаючи дату, текст відгуку, працівника та рецензента.

Кожен з цих класів моделей використовується для зберігання конкретних типів даних, які використовуються в системі обліку працівників авіакомпанії.

Зареєстровано моделі в адмін панелі для можливості вносити дані та керувати процесами з прав адміністратора. Програма адміністратора використовує моделі для автоматичного створення області сайту, яку можна використовувати для створення, перегляду, оновлення та видалення записів. Це заощаджує час під час розробки, спрощуючи тестування моделей і визначення того, чи є у вас правильні дані.

Додаток адміністратора також корисний для керування даними у виробництві. Адмін панель рекомендовано використовувати лише для внутрішнього керування даними (тобто лише для використання адміністраторами), оскільки модельно-орієнтований підхід не обов’язково є найкращим можливим інтерфейсом для всіх користувачів і відкриває багато непотрібних деталей про моделі.

Опис того, що роблять адміністративні панелі для кожної моделі:

1. *TagAdmin*:

− Дозволяє адміністраторам переглядати та управляти мітками (тегами).

- Відображає назву мітки.

2. *TaskAdmin*:

- Надає можливість переглядати та керувати завданнями.
- Показує заголовок, опис, дату та час створення, дедлайн, пріоритет, статус, пов'язаний рейс та інші поля.

3. *EmployeeAdmin*:

- Дозволяє адміністраторам керувати інформацією про працівників.
- Відображає інформацію про працівника, таку як ім'я користувача, посаду, електронну пошту, номер телефону та інші дані.

4. *FlightAdmin*:

- Надає можливість керувати рейсами.
- Надає можливість фільтрувати рейси.
- Показує інформацію про рейс: аеропорти відправлення та прибуття, тип літака, капітана та екіпаж.

```
@admin.register(Flight)
class FlightAdmin(admin.ModelAdmin):
    list_display = (
        "departure_airport",
        "arrival_airport",
        "departure_date_time",
        "arrival_date_time",
        "aircraft_type",
        "captain",
    )
    list_filter = (
        "departure_airport",
        "arrival_airport",
        "departure_date_time",
        "arrival_date_time",
        "aircraft_type",
```

*)*

*search_fields = ("departure_airport", "arrival_airport", "aircraft_type")*

*filter_horizontal = ("crew",)*

5. *ShiftAdmin*:

- Дозволяє керувати графіками роботи (змінами) працівників.
- Показує інформацію про початок та закінчення зміни, перерву та пов'язані завдання.

6. *EquipmentAdmin*:

- Надає можливість керувати обладнанням.
- Показує інформацію про тип, модель, серійний номер та розташування обладнання.

7. *PerformanceReviewAdmin*:

- Дозволяє керувати відгуками про ефективність працівників.
- Відображає інформацію про працівника, якого оцінюють, рецензента та дату відгуку.

Кожна з цих адміністративних панелей *Django Admin* дозволяє адміністраторам виконувати операції створення, оновлення, видалення та перегляду даних для відповідних моделей додатку.

Створено представлення (*View*) - це частини, що відповідають за обробку запитів та підготовку даних для відображення. Опис того, що роблять деякі з цих в'юшок:

1. *IndexView*:

- Відповідає за відображення головної сторінки адміністративної панелі.
- Виконує підрахунок кількості завдань для кожного статусу (очікування, в процесі, завершено) і передає цю інформацію на головну сторінку.

2. *EmployeeListView*:

- Відображає список всіх працівників.
- Дозволяє виконати пошук за ім'ям користувача.

*class EmployeeListView(LoginRequiredMixin, generic.ListView):*

*model = Employee*

```
paginate_by = 5
def get_context_data(self, *, object_list=None, **kwargs):
    context = super(EmployeeListView, self).get_context_data(**kwargs)
    username = self.request.GET.get("username", "")
            context["search_form"] = EmployeeSearchForm(initial={"username":
username})
    return context
def get_queryset(self):
    queryset = Employee.objects.all()
    username = self.request.GET.get("username")
    if username:
        return queryset.filter(username__icontains=username)
    return queryset
```

3. *EmployeeDetailView*:
- Показує деталі про конкретного працівника.
4. *EmployeeCreateView*:
- Дозволяє створити новий запис про працівника.
5. *EmployeeUpdateView:*
- Надає можливість оновити існуючий запис про працівника.
6. *EmployeeDeleteView*:
- Дозволяє видалити запис про працівника.
7. *TagListView, TagCreateView, TagUpdateView, TagDeleteView*:
- Виконують ті самі функції для обробки тегів, які можуть бути прив'язані до завдань.
8. *TaskListView, TaskDetailView, TaskCreateView, TaskUpdateView, TaskDeleteView:*
- Реалізують функціонал для роботи з завданнями, таким як створення, оновлення, видалення та перегляд.
9. *FlightListView, FlightDetailView, FlightCreateView, FlightUpdateView, FlightDeleteView*:

- Відповідають за операції з рейсами: перегляд, створення, оновлення, видалення.

10. *ShiftListView, ShiftDetailView, ShiftCreateView, ShiftUpdateView, ShiftDeleteView*:

- Забезпечують можливість працювати зі змінами (графіками роботи), включаючи створення, оновлення та видалення.

Використовуючи шаблони (*Templates*), створено вигляди, які представляють дані у вигляді, зрозумілому для користувача, розробляємо *HTML* шаблони, використовуючи можливості фреймворку *Bootstrap*.

Реалізовано форми для введення даних користувачами на сторінках.

Лістинг програми знаходиться в Додатку А.

### 3.4. Тестування програмного комплексу

Тестування програмного забезпечення – це процес перевірки та оцінки програми з метою виявлення помилок, недоліків, а також визначення відповідності програми вимогам та очікуванням користувачів. Цей процес включає в себе запуск програми з метою виявлення помилок, перевірку функціональності та коректності роботи програми.

Тестування програмного продукту є важливою частиною процесу розробки. При розробці програмного комплексу обліку роботи працівників логістичної авіакомпанії було використано як мануальне так і автоматизоване тестування. Користування як мануальним, так і автоматизованим тестуванням є важливим для забезпечення якості програмного продукту перед його випуском на ринок. Мануальне тестування дозволяє виявити аспекти, які можуть бути пропущені в автоматизованих тестах, водночас автоматизоване тестування забезпечує швидкість та повторюваність виконання тестів.

Середовище програмування *PyCharm* та фреймворк *Django* надають можливості створювати тести безпосередньо в проєкті без застосування спеціальних плагінів. Для написання була використана вбудована бібліотека unitest. Вона

дозволяє писати тести для перевірки окремих компонентів програми або її модулів. У *Django* вбудована підтримка *unitest*, що спрощує написання тестів для перевірки відповідності функцій, видів і моделей програми вимогам та очікуванням користувачів.

Було перевірено роботу окремих модулів програми та взаємодію один з одним. Для тестування були використані такі сценарії:

- Редагування інформації працівника: Тестує, чи система дозволяє редагувати дані працівника, такі як ім'я, прізвище, контактні дані, посада тощо.
- Видалення працівника: Тестує, чи система дозволяє видаляти працівника з бази даних, при цьому видаляючи всі його пов'язані дані (зміни, завдання, рейси).
- Авторизація працівника: Тестує, чи система дозволяє працівнику ввійти в систему з правильними логіном та паролем, а також блокує вхід з неправильними даними.
- Створення нового завдання: Тестує, чи система дозволяє коректно вводити назву, опис, термін виконання, пріоритет та призначати виконавців.
- Редагування завдання: Тестує, чи система дозволяє редагувати завдання, змінювати його статус, додавати або видаляти виконавців та теги.
- Виконання завдання: Тестує, чи система дозволяє виконавцю відмітити завдання як виконане, а також відображає виконані завдання в окремому списку.
- Пошук завдань: Тестує, чи система дозволяє шукати завдання за назвою.
- Створення нового рейсу: Тестує, чи система дозволяє коректно вводити інформацію про рейс, таку як аеропорти відправлення та прибуття, дату та час, тип літака, капітана та екіпаж.
- Призначення працівників на рейс: Тестує, чи система дозволяє призначати працівників на рейс відповідно до їхніх ролей та кваліфікації.
- Відстеження статусу рейсу: Тестує, чи система дозволяє відстежувати статус рейсу в режимі реального часу.

- Пошук рейсів: Тестує, чи система дозволяє шукати рейси за датою, аеропортом, типом літака тощо.
- Зміна складу екіпажу: Тестує, чи система дозволяє змінювати склад екіпажу рейсу в разі потреби.
- Створення нового робочого графіка: Тестує, чи система дозволяє створювати робочі графіки для працівників, вказуючи час початку та завершення зміни, перерви тощо.
- Призначення працівників на зміну: Тестує, чи система дозволяє призначати працівників на робочі зміни відповідно до їхніх графіків та доступності.
- Створення нового оцінювання працівника: Тестує, чи система дозволяє оцінювати роботу працівника за певний період, вказуючи його сильні та слабкі сторони, рекомендації для розвитку.

При розробці тестових сценаріїв та при процесі самого тестування було враховувано: чи система відповідає очікуванням користувачів та чи перевіряє найбільш важливі функції системи.

### 3.3. Висновки до розділу

У цьому розділі було визначено функціональні вимоги до програмного комплексу обліку роботи працівників логістичної авіакомпанії. Вимоги включають створення, збереження та редагування персональних даних працівників, контроль виконання завдань, ведення інформації про рейси, організацію графіку змін працівників та оцінку їхньої продуктивності. Ці вимоги є важливими для забезпечення ефективного функціонування обліку робочого процесу.

Показано схема зв’язків бази даних, включаючи таблиці, які забезпечують збереження та обробку важливих даних, таких як інформація про працівників, завдання, зміни, оцінки результативності та інші. Ці зв’язки в базі даних дозволяють ефективно управляти інформацією та виконувати основні функції програмного комплексу. Описано таблиці бази даних.

Оскільки веб-браузери постійно оновлюються, важливо мати комп'ютер з достатніми програмними та апаратними ресурсами, щоб підтримувати їхню роботу, тому було розписано мінімальні та бажані вимоги до програманого та апаратного забезпечення.

Було наведено архітектурний шаблон веб-додатку, де було показано клієнтську та серверну частини. Проект базується на фреймворку *Django*, використовуючи концепцію *MVT* для побудови його архітектури. *MVT* допомагає розділити логіку додатка на окремі, керовані частини, що робить його більш модульним, повторно використовуваним та легше зрозумілим. Кожну частину додатку (модель, вигляд, шаблон) було протестовано окремо, що полегшує написання та обслуговування тестів вподальшому. Також було використано мануальне тестування для виявити аспекти, які можуть бути пропущені в автоматизовани. Помилки які були виявленні під час тестів були успішно вирішені.

Було показано організація програмного коду. Описано та реалізовано класи моделей, які використовуються для зберігання конкретних типів даних в системі обліку працівників логістичної авіакомпанії та адміністративна панель для кожної моделі. Описано та створено представлення, що відповідаютьь за обробку запитів. Реалізовано форми для введеня працівниками даних. За допомогою фреймворку *Bootstrap* розроблено *HTML* шаблони, для представлення даних у вигляді зрозумілому користувачу.

Наведно частини коду для кращого розуміння розробки. В результаті було розроблено програмний комплекс обліку працівників логістичної авіакомпанії. Інтерфейс веб-застосунку зроблено інтуїтивно зрозумілим користувачу, що робить його зручним у використанні.

# РОЗДІЛ 4
# ПРИКЛАД ЗАСТОСУВАННЯ

## 4.1. Можливості роботи працівників з веб-застосунком

Що б скористатись програмним комплексом необхідно перейти за відповідним посиланням та ввести логін та пароль, який надається керівництвом (рис. 4.1).

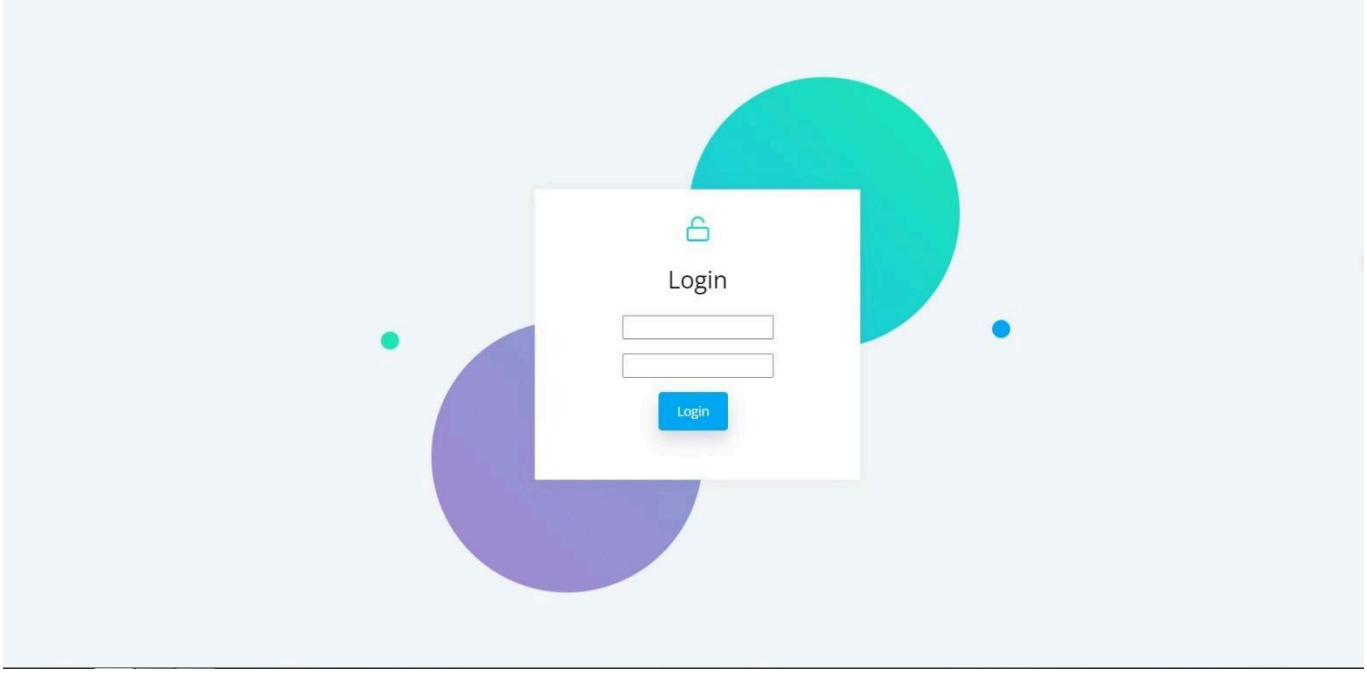


Рис. 4.1. Вхід в систему

Після входу в систему на головній сторінці знаходиться панель з завданнями, таймер, який запускається після натискання “*Start the Working Day*” та можливість призупинити таймер (рис. 4.2). По всій програмі є можливість встановлювати світлу та темну тему.

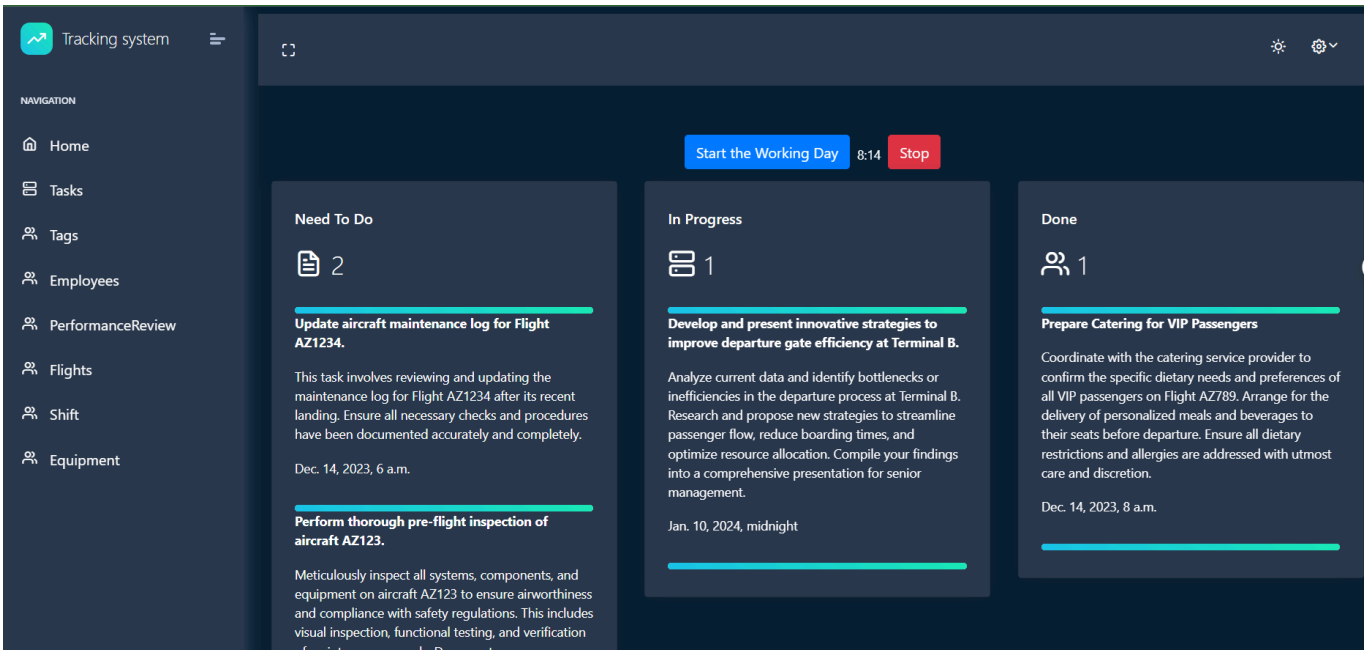


Рис. 4.2. Головна сторінка

Усі завдання також можна переглянути по вкладці “*Tasks*” (рис. 4.3) та у особистому кабінеті (рис. 4.4).

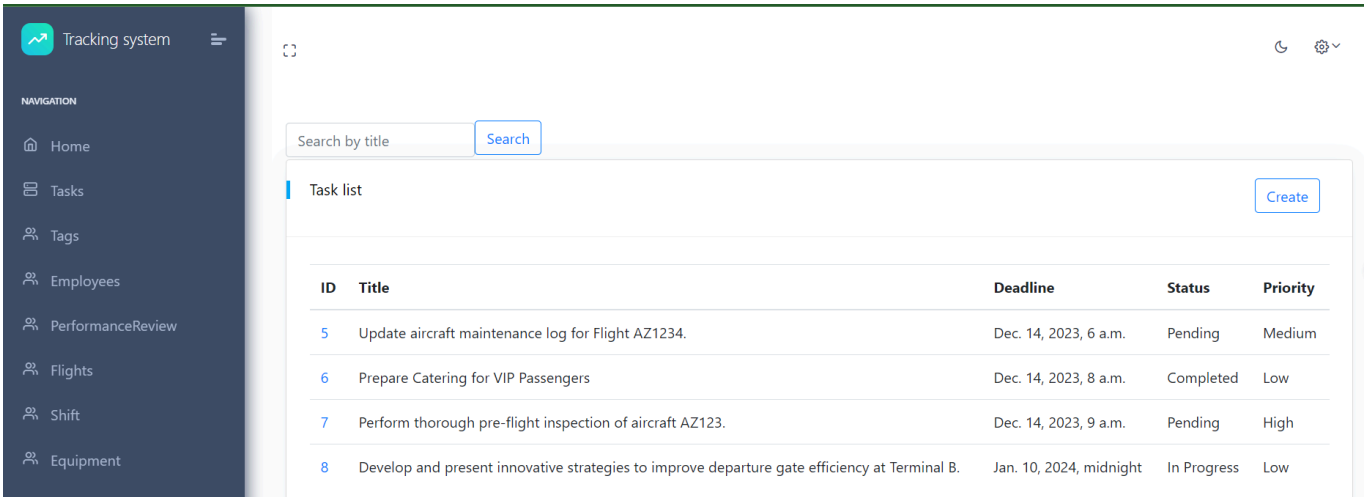


Рис. 4.3. Сторінка “*Tasks*”

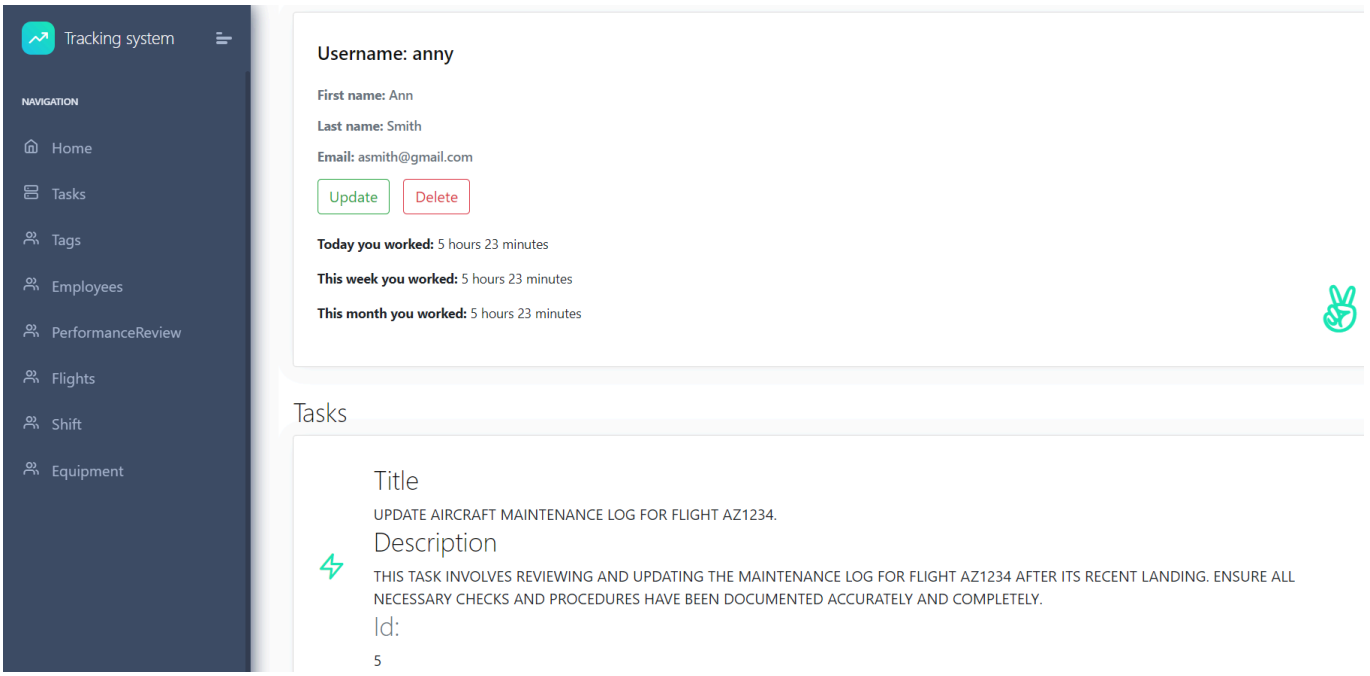


Рис. 4.4. Особистий кабінет

Список усіх співробітників може переглянутий кожен зареєстрований, але детальну інформацію про співробітника можна бачити лише з правами керівника (рис. 4.5).

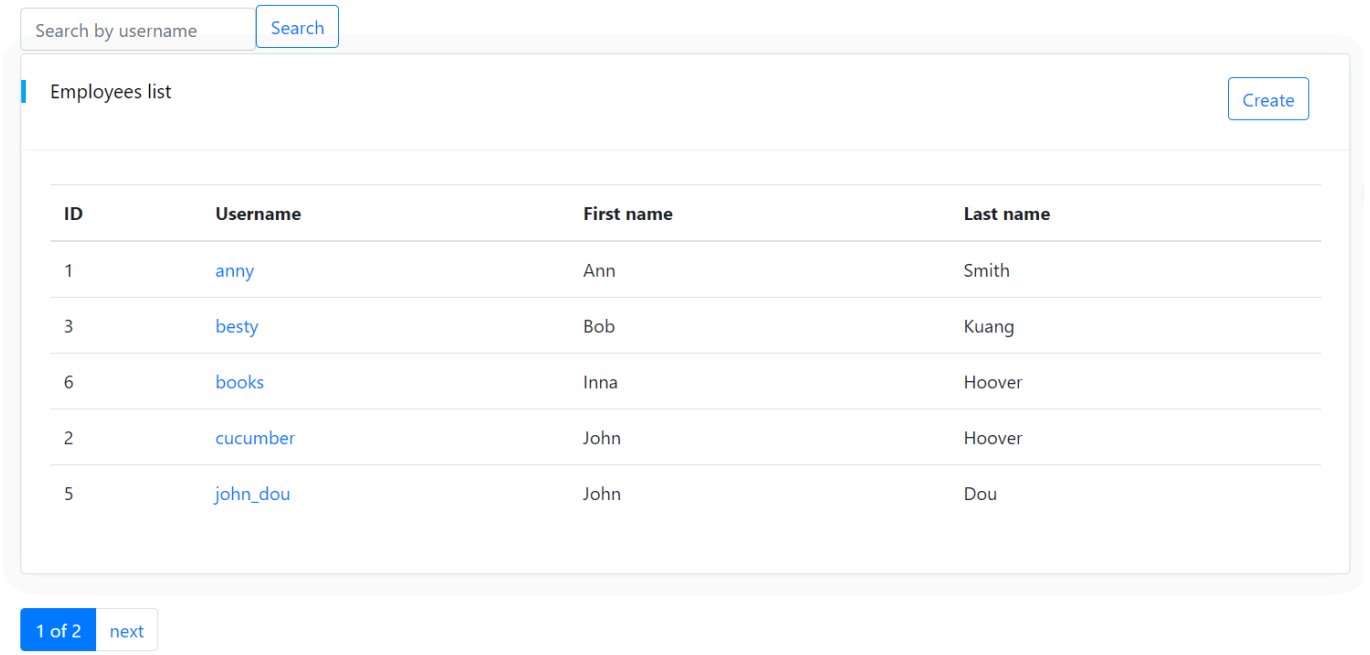


Рис. 4.5. Список співробітників

Перейшовши на вкладку “*Flights*” можна моніторити відправку, прибуття літаків (рис. 4.6), натиснувши на сам рейс можна побачити детільну інформацію про нього (рис. 4.7) та на цій сторінці оновлювати інформацію про політ (рис. 4.8) та видаляти рейс.

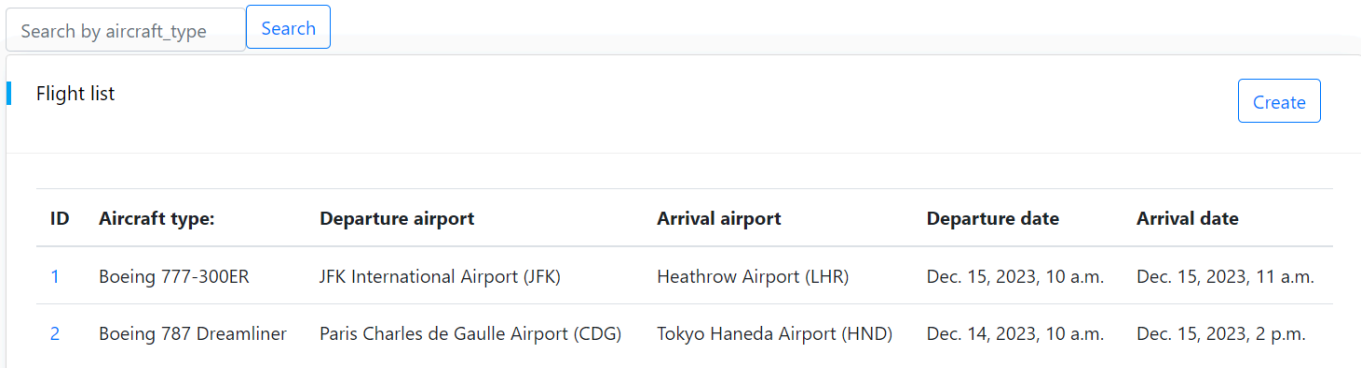


Рис. 4.6. Список рейсів

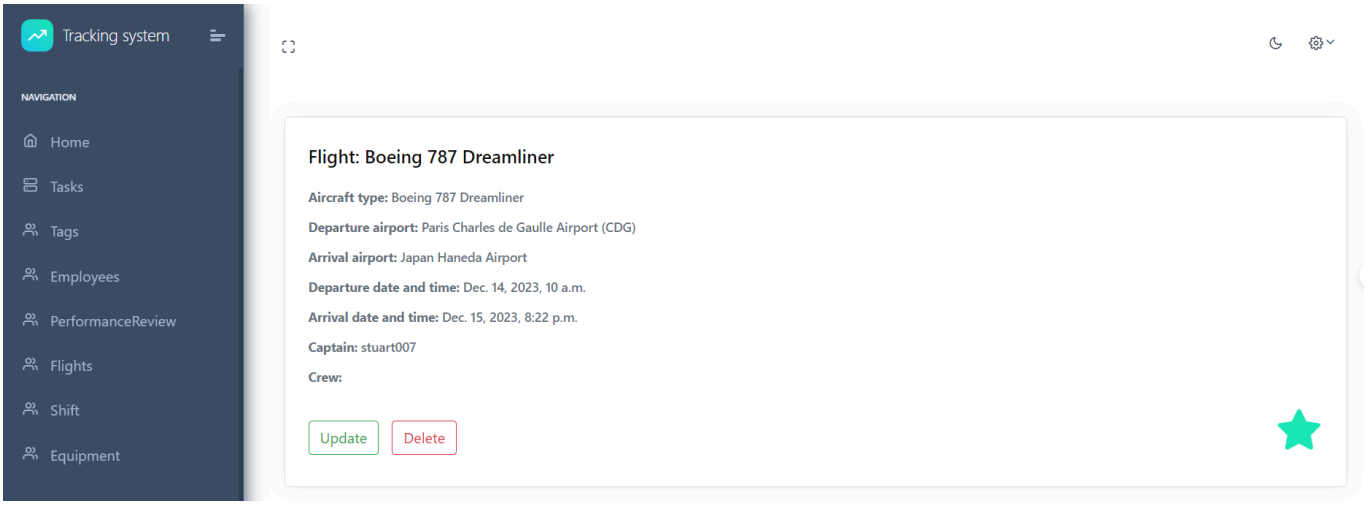


Рис. 4.7. Детальна інформація про рейс

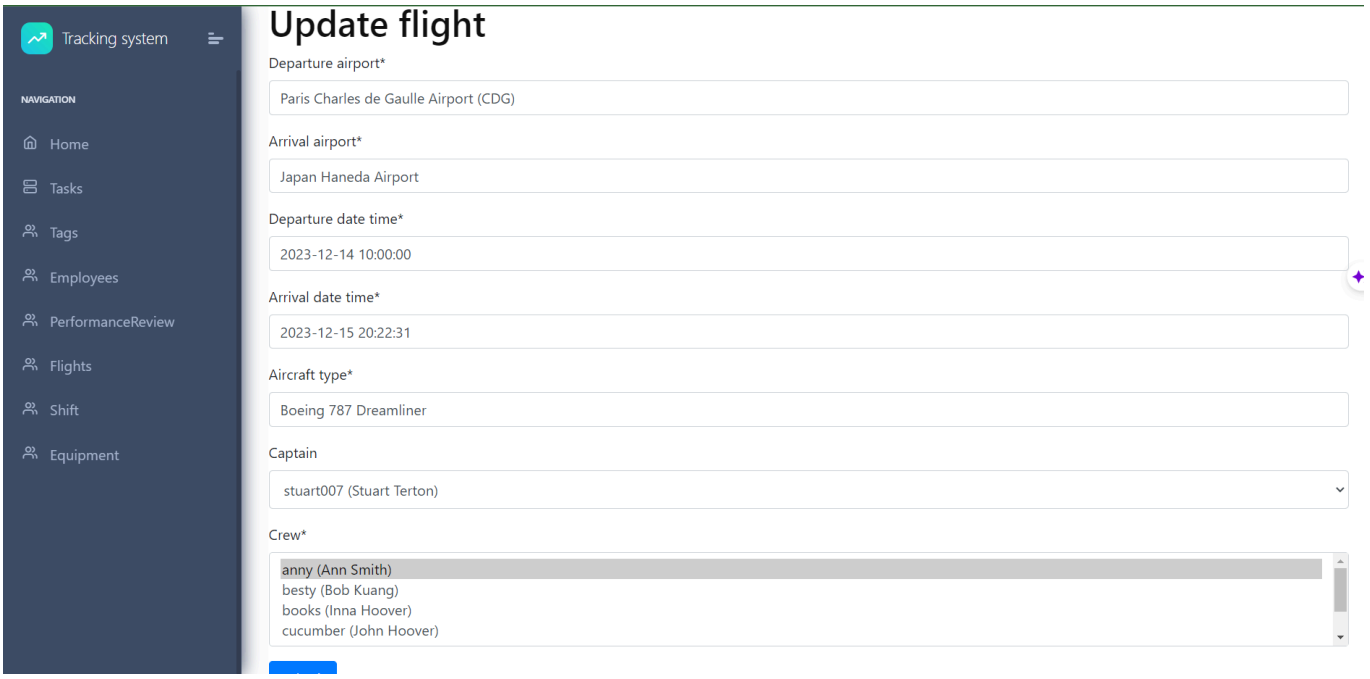


Рис. 4.8. Оновлення інформації про рейс

З правами доступу керівника можна назначати зміни працівникам (рис. 4.9).

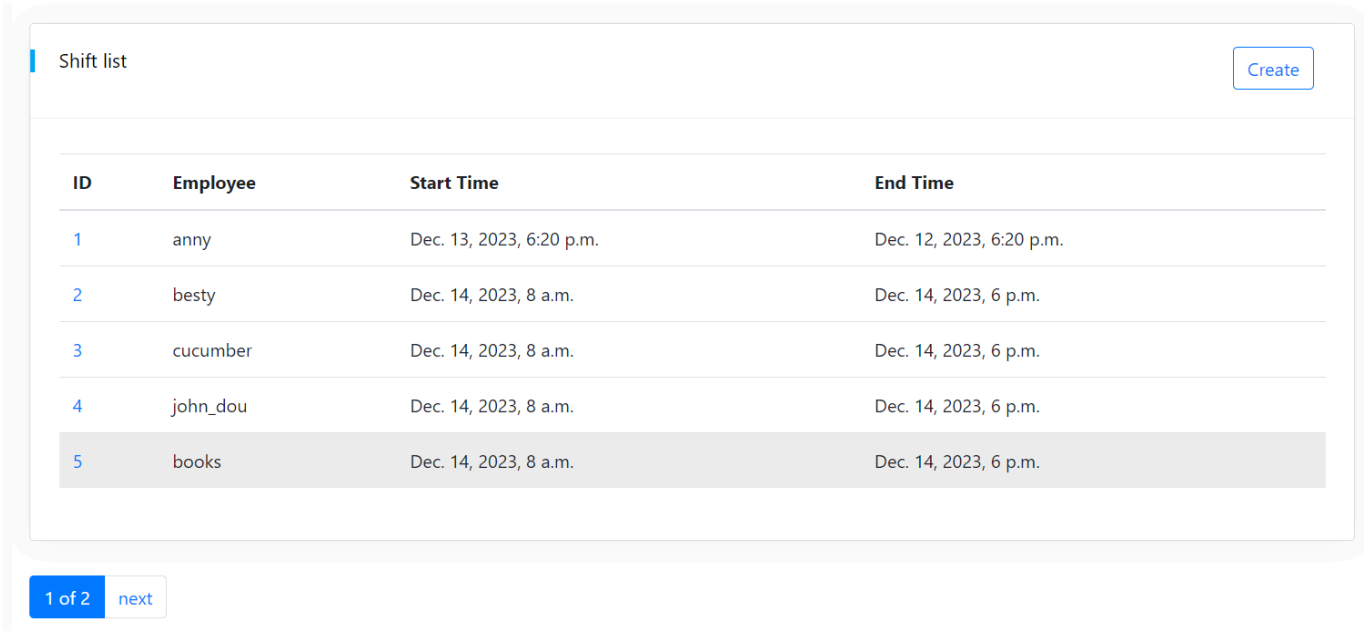


Рис. 4.9. Список змін

### 4.2. Можливості роботи адміністратора з веб-застосунком

Перейшовши по відповідному *endpoint* адміністратора (/*admin*) необхідно ввести логін та пароль, який завчасно створюється та надається розробниками або відповідальною особою (рис. 4.10).

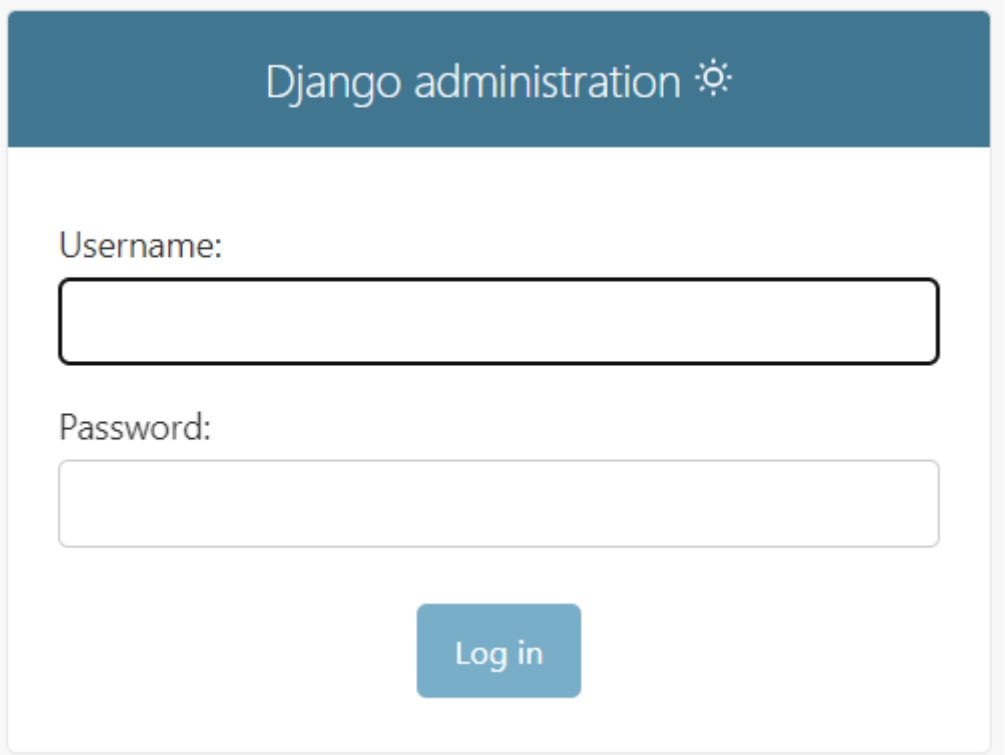


Рис. 4.10. Сторінка логіну адміністратора

Після успішного входу на головні сторінці (рис. 4.11) відображена інформація про останні дії адміністратора та панель швидкого доступу для створення дозволів різних рівнів та доступу до моделей для створення, перегляду, оновлення та видалення записів.

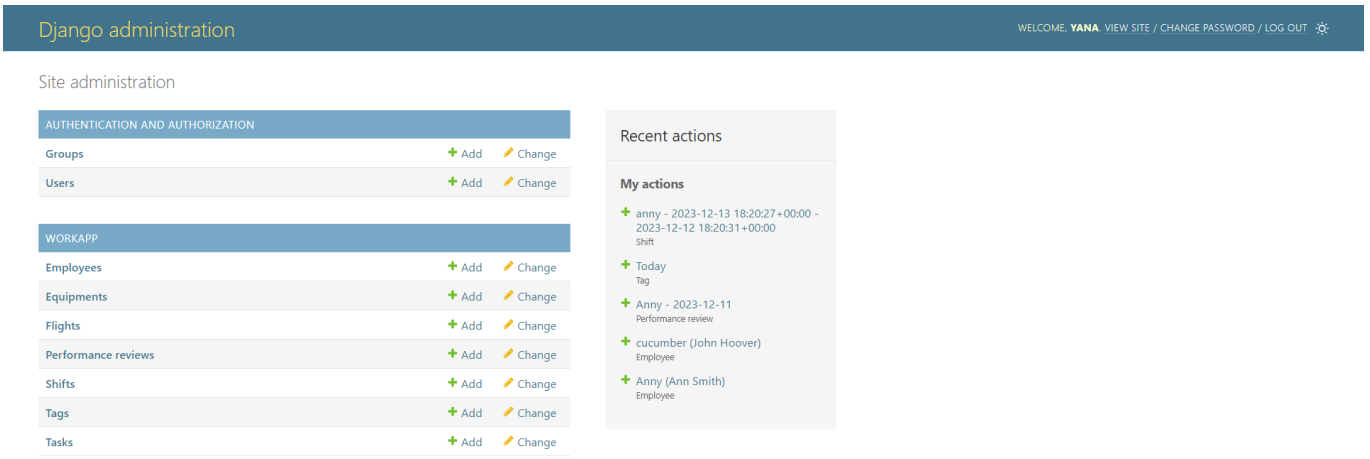


Рис. 4.11. Головна сторінка адміністратора

Перейшовши по моделі Flights одразу потрапляємо на список усіх рейсів, які знаходяться в базі даних, з детальною інформацією про кожен з них. Також по кожному з пуктів можна провести фільтрацію для швидшого пошуку або ж якщо відомий точний рес здійснити пошук за допомогою пошукової строки (рис. 4.12).

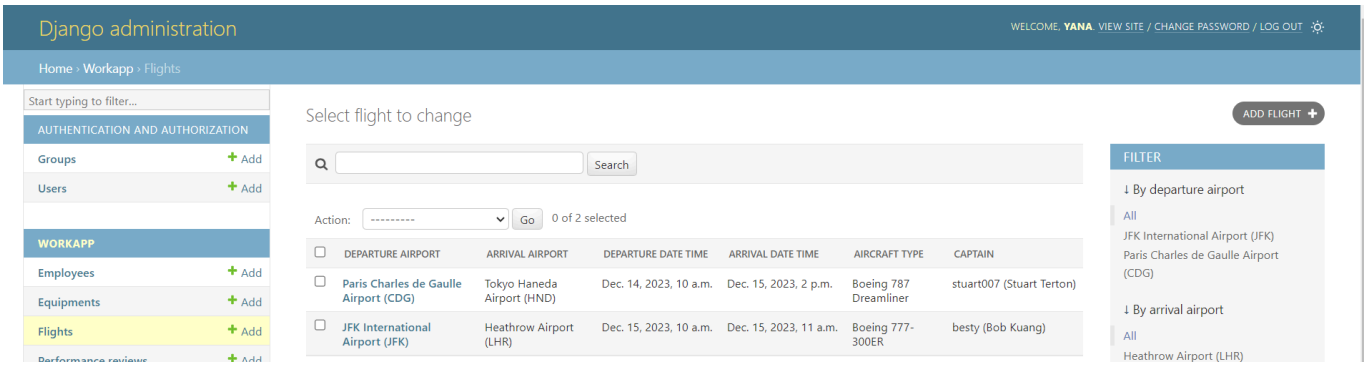


Рис. 4.12. Список рейсів у адмін панелі

За допомогою адмін панелі можна створювати та редагувати наявні записи (рис. 4.13).

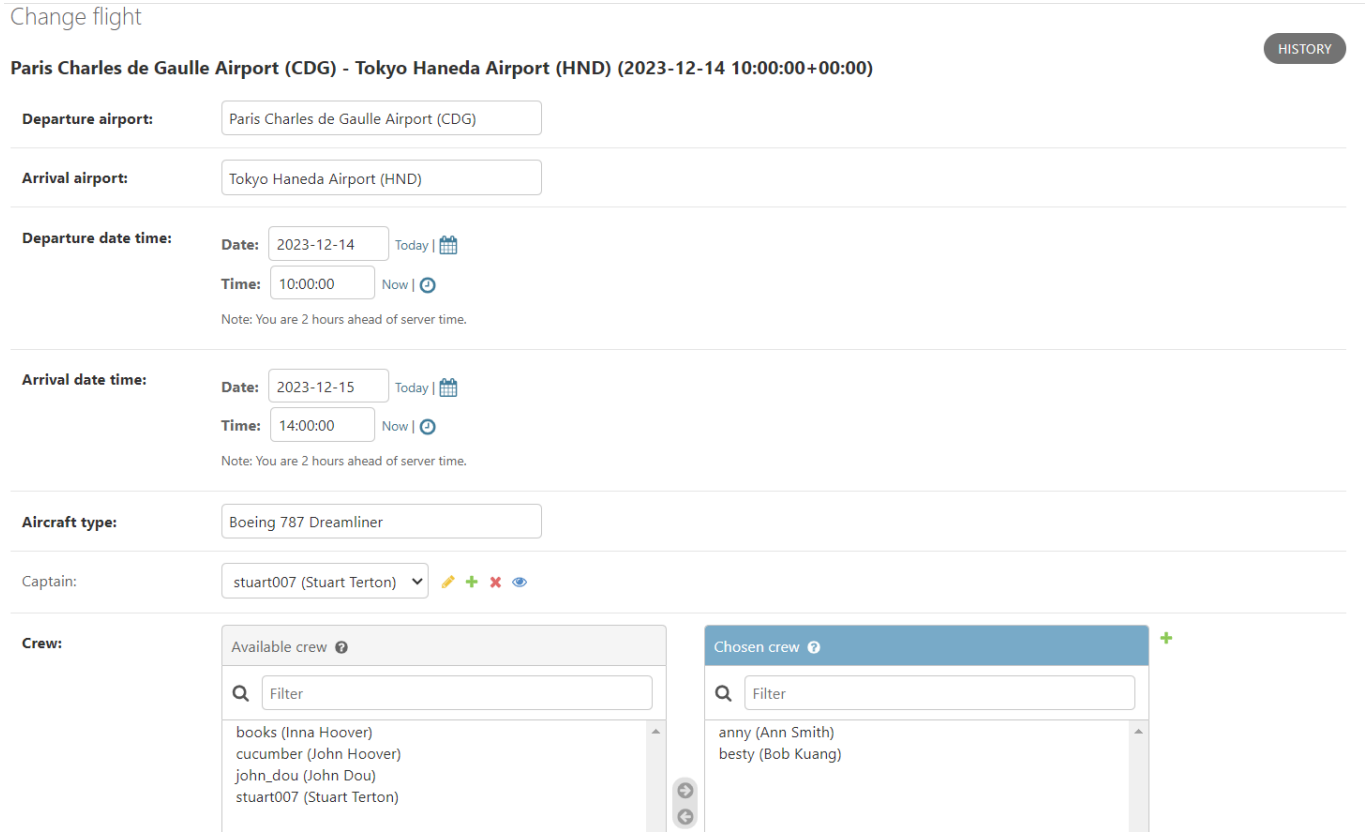


Рис. 4.13. Створення/редагування запису

Переглянути історії змін обраного запису також можливо після натискання "*History*" (рис. 4.14).


Рис. 4.14. Історія змін

При зміні, оновленні інформації в полях система сповіщає про їх успішне виконання. (рис. 4.15)

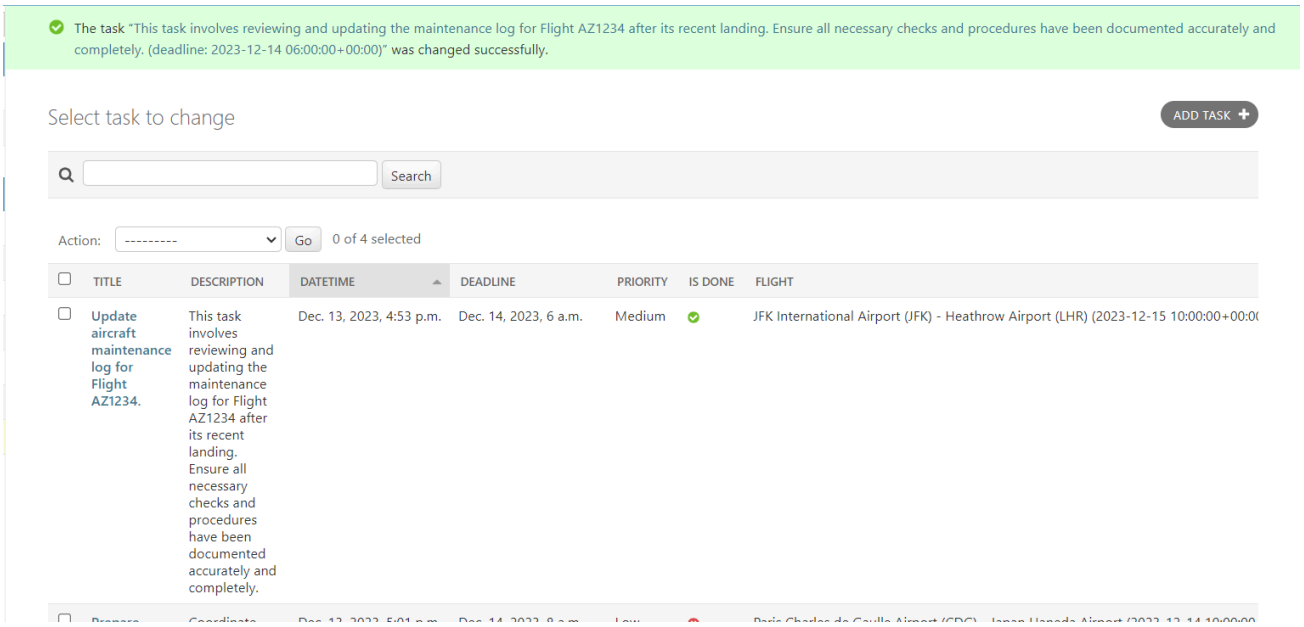


Рис. 4.15. Сповіщення про успішне виконання

Подібним чином адмін панель працює і для інших моделей. Навіть якщо у самому застосунку не можливо створити записи для певних моделей, то з правами адміністратора та адмін панелі *Django* можна легко доступитись до будь-якої інформації та створити, оновити чи видалити запис. Всі зміни які вносяться у панелі адміністратора автоматично відбуваються і на самому сайті. Тому вжливо пам’ятати, що досуп до адмін панель має бути лише у відповідної особи.

## 4.3. Висновки до розділу

У даному розділі було представлено можливості використання програмного комплексу обліку роботи працівників логістичної авіакомпанії для співробітників та адміністраторів.

Для співробітників веб-застосунок пропонує можливості доступу до головної сторінки, де вони можуть переглядати та виконувати, редагувати, видаляти завдання, вести облік робочого часу. Також отримувати доступ до особистого кабінету, де можна відредагувати особисту інформацію та побачити завдання які необхідно

виконати. Додатково усі завдання можна бачити та відслудковувати перейшовши по вкладці “*Tasks*”. Усі рейси літаків можна переглянути перейшовши по відповідній вкладці. Побачити додаткову інформацію про рейс можна обравши бажаний рейс зі списку або ж через пошук.

Для адміністраторів, система надає більш широкі можливості керування: від редагування та видалення записів до надання дозволів та управління правами доступу до різних моделей бази даних. Адміністратор може переглядати історію змін та впливати на роботу системи. Під час розробки адміністративну панель зручно використовувати для тестуваня системи.

Таким чином, надані можливості веб-застосунку для співробітників та адміністраторів спрощують рутинні завдання, сприяють впорядкуванню робочих процесів та забезпечують зручний та ефективний контроль за важливими аспектами роботи.

# ВИСНОВКИ

У сучасному світі, що швидко розвивається в технологічному плані, технології впливають на майже кожен аспект нашого життя. Зростання важливості технологій у сфері авіації та управління вимагає постійного вдосконалення та оптимізації різних процесів, включаючи облік робочого часу та продуктивності працівників. Логістичні авіакомпанії, незалежно від їхнього розміру, все більше розуміють, що вивчення, аналіз та ефективне управління робочим часом та діяльністю персоналу є критично важливими для досягнення успіху та конкурентоспроможності.

Технологічний прогрес приводить до впровадження новітніх інструментів та систем, спрямованих на полегшення та оптимізацію управління персоналом. Облік роботи та впровадження систем ефективного контролю стають важливими аспектами для підвищення продуктивності, збільшення якості роботи та впровадження новаторських стратегій у сфері управління людськими ресурсами.

В результаті кваліфікаційнрї роботи було створено програмний комплекс обліку роботи працівників логістичної авіакомпанії. Було виявлено, що розробка веб-застосунку для обліку роботи працівників у логістичних авіакомпаніях є важливим кроком у відповідь на потреби сучасного бізнесу. Вимоги до функціональності програмного забезпечення включають широкий спектр функцій, починаючи від збереження персональних даних працівників і закінчуючи контролем робочих завдань та оцінкою продуктивності.

У першому розділі визначена важливість обліку роботи працівників. Розглянуто існуючі програмні рішення, визначено їх переваги та надоліки. Проаналізовано можливі технології для розробки та визначено створювати саме веб-застосунок так як він відповідає сучасному підходу до життя.

У другому розділі було проаналізовано та обрано технології для розробки, а саме *Django*, *Bootstrap* та *PostgreSQL*. Django надає стійкість та безпеку на бекенді, забезпечуючи високий рівень функціональності та стабільність у роботі. *Bootstrap*, у свою чергу, забезпечує динамічну та інтуїтивно зрозумілу користувацьку інтерфейсу

на фронтенді, дозволяючи зберегти легкість використання та високу продуктивність. *PostgreSQL*, як потужна реляційна база даних, гарантує ефективне зберігання та обробку даних, а також сприяє розширюваності та надійності системи.

Визначено середовища розробки, обрано *PyCharm*, так як він задовляє основним вимогам таких як доступність, інтегрованість середовища, підтримка багатьох бібліотек, фреймворків, зручність. Як основнйи веб браузер для розробки, налаштування та тестування було обрано *Chrome*, так як він являється одним з найпопулярніш рішень та має переваги такі як вбудований інструмент для відлагоджування застосунків та постіне оновлення браузера.

У тертьому розділі були визанчені функціональні вимоги до програмного комплексу облік уроботи працівників, такі як створення, збереження та редагування персональних даних працівників, створення та контроль виконання завдань, ведення інформації про рейси, організацію графіку змін працівників та оцінку їхньої продуктивності.

Також було надано мінімальні та рекомендовані системні вимоги для кращого функціонування програмного комплексу облік уроботи працівників логістичної авіакомпанії.

Розглянуть структуру бази даних на якій показано таблиці та взаємодії між ними. У даній роботі використовувалась база даних *PosgresSQL* так як вона є одною з найнадійніших та найпростіших у використанні. Ця структура бази даних дозволяє ефективно управляти інформацією та виконувати основні функції програмного комплексу.

Так як програмний комплес базується на фреймворку *Django*, була використовуючи концепцію *MVT* для побудови його архітектури. Архітектура *MVT* складається з трьох компонентів — моделі, перегляду та шаблону. Ці компоненти є дуже важливими, оскільки вони працюють разом для обробки логіки, даних і презентації нашого веб-додатку. Він забезпечує чіткий розподіл завдань, що полегшує розуміння, підтримку та тестування коду. Він також забезпечує багаторазове використання та гнучкість, що робить його хорошим вибором для програмного комплесу обліку роботи працівників логістичної авіакомпанії.

Було показано структуру програми та розроблено сам програмний комплекс обліку працівників логістичної авіакомпанії. Проведено тестування програмного комплексу використовуючи мануальне та автоматизоване тестування. Для тестування було підготовлені тестові сценарії та написані тест кейси у автотестах.

У четвертому розділі було продемонстровано роботу програмного комплесу, основні елементи та можливості для працівників та адміністраторів. За допопомогою даного веб-додатку можна легко та зручно відслідковувати час роботи, процес виконання завдань, відслідковування та створення рейсів, змін працівників, залишати відгуки по роботі для покращення продуктивності працівника у майбутньому. Інтерфейс додатку зручний та простий у використані, має темну тему для меншого напруження зору під час роботи.

## СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ


1. ПОЛОЖЕННЯ ПРО ДИПЛОМНІ РОБОТИ (ПРОЕКТИ) ВИПУСКНИКІВ НАЦІОНАЛЬНОГО АВІАЦІЙНОГО УНІВЕРСИТЕТУ. Київ: НАУ, 2017.
2. ДОКУМЕНТАЦІЯ. ЗВІТИ У СФЕРІ НАУКИ І ТЕХНІКИ. Структура і правила оформлення. ДСТУ 3008-95. Київ.
3. Андрєєв А. В., Сєров В. М., Скрипченко В. С. Проектування та розробка інформаційних систем: Підручник. – К.: Видавничий дім "Ніка-Центр", 2015. – 400 с.
4. Бойко В. В., Мельник С. А., Мельник В. В. Інформаційні технології в управлінні: Підручник. – К.: КНЕУ, 2016. – 480 с.
5. Даниленко В. П., Марченко В. В., Пономаренко В. В. Інформаційні системи та технології: Підручник. – К.: КНЕУ, 2016. – 448 с.
6. *How to Build Task Management App.* [Електронний ресурс].- режим доступу: *https://www.freshcodeit.com/freshcode-post/how-to-create-task-management-app-mvp* (дата звернення 25.10.2023). - Назва з екрана.
7. Гриценко А. А. Інформаційні технології в управлінні підприємством. Київ: КНЕУ, 2015. 403 с.
8. *Build a Simple Tracker Program Using Python and SQL* [Електронний ресурс].- режим доступу: *https://algakovic.medium.com/build-a-simple-tracker-program-using-python-and-sql-video-fb839fbd97c3* (дата звернення 22.10.2023). - Назва з екрана.
9. *How to automate working time control?* [Електронний ресурс].- режим доступу: *https://www.c4r.eu/blog/digital-hr/kontrol-rabochego-vremeny/* (дата звернення 25.10.2023). - Назва з екрана.
10. *What is Aviation Logistics?* [Електронний ресурс].- режим доступу: *https://cob.unt.edu/lom/what-is-aviation-logistics* (дата звернення 12.10.2023). - Назва з екрана.

11. *Air Logistics International.* [Електронний ресурс].- режим доступу: *https://www.airlogisticsinternational.com/* (дата звернення 15.10.2023). - Назва з екрана.
12. *Logistic Tracking System: How to Optimize and Streamline Your Business Logistics?* [Електронний ресурс].- режим доступу: *https://www.eliftech.com/insights/logistic-tracking-system-for-e-commerce/* (дата звернення 20.10.2023). - Назва з екрана.
13. Про організацію роботи працівників, режим та облік робочого часу. [Електронний ресурс].- режим доступу: *https://ibuhgalter.net/news/4778* (дата звернення 15.10.2023). - Назва з екрана.
14. Пушников А. Ю. Введення в системи управління базами даних: навч. посібник. Київ: Просвіта, 2013. 85 с.
15. Папенко Л. М. Підходи до визначення поняття «клієнтоорієнтованість» у контексті управління сервісним підприємством. Ч. 3. Науковий вісник Херсонського державного університету. 2020. С. 45–79.
16. Управління логістичними бізнес-процесами з використанням інтелектуальних технологій на підприємствах авіаційної галузі: дипломна робота на здобуття освітнього ступеня магістр логістики. Київ: Національний авіаційний університет, 2015. 239 с.
17. *Your business challenges, solved.* [Електронний ресурс].- режим доступу: *https://www.microsoft.com/en-us/dynamics-365/solutions/small-business* (дата звернення 4.10.2023). - Назва з екрана.
18. Transportation Management. [Електронний ресурс].- режим доступу: *https://www.oracle.com/scm/logistics/transportation-management/* (дата звернення 4.10.2023). - Назва з екрана.
19. *SAP* [Електронний ресурс].- режим доступу: *https://www.sap.com/products/erp.html* (дата звернення 15.10.2023). - Назва з екрана.
20. Все, що вам потрібно знати про облік робочого часу працівників. [Електронний ресурс].- режим доступу:

*https://yaware.com.ua/uk/blog/vse-shho-vam-potribno-znati-pro-oblik-robochogo-chasu-praczivnikiv/* (дата звернення 4.10.2023). - Назва з екрана.

21. *IBM Support* [Електронний ресурс].- режим доступу: *https://www.ibm.com/support/pages/about-sterling-tms* (дата звернення 5.10.2023). - Назва з екрана.
22. Вожжова, К. А. Удосконалення логістики міжнародних авіа-перевезень : дипломна робота бакалавра Київ КПІ, 2020. 90 с.
23. Офіційний веб-сайт Державної служби статистики України. [Електронний ресурс].- режим доступу: *http://www.ukrstat.gov.ua/* (дата звернення 25.10.2023). - Назва з екрана.
24. *9 IT Trends Shaping the Aviation Industry in 2023. NCS.* [Електронний ресурс].- режим доступу: *https://www.networkcablingservices.com/9-it-trends-shaping-the-aviation-industry/* (дата звернення 25.10.2023). - Назва з екрана.
25. *The MVT Design Pattern of Django.* [Електронний ресурс].- режим доступу: *https://python.plainenglish.io/the-mvt-design-pattern-of-django-8fd47c61f582/* (дата звернення 25.10.2023). - Назва з екрана.
26. *Django*. [Електронний ресурс].- режим доступу: *https://www.djangoproject.com/* (дата звернення 1.11.2023). - Назва з екрана.
27. Формування логістичної стратегії авіакомпанії в умовах глобалізації [Електронний ресурс].- режим доступу: *http://www.kdpu-nt.gov.ua/uk/content/formuvannya-logistychnoyi-strategiyi-aviakompaniyi-v-umovah-globalizaciyi* (дата звернення 14.10.2023). - Назва з екрана.
28. Про захист персональних даних: Закон України від 1 червня 2010 р. № 2297-VI. Верховна Рада України. [Електронний ресурс].- режим доступу: *https://zakon.rada.gov.ua/laws/show/2297-17#Text* (дата звернення 25.10.2023). - Назва з екрана.
29. *Django Tutorial Part 4: Django admin site.* [Електронний ресурс].- режим доступу:*https://developer.mozilla.org/en-US/docs/Learn/Serverside/Django/Admin_site* (дата звернення 25.10.2023). - Назва з екрана.

30. *Desktop or Web Application: What to Develop* [Електронний ресурс].- режим доступу: *https://exoft.net/desktop-or-web-application-what-to-develop/* (дата звернення 1.11.2023). - Назва з екрана.
31. *The 7 Best Web Development Languages* [Електронний ресурс].- режим доступу: *https://www.bairesdev.com/blog/best-programming-languages-web-development/* (дата звернення 3.11.2023). - Назва з екрана.
32. *What are the best database options for web development?* [Електронний ресурс].- режим доступу: *https://www.linkedin.com/advice/1/what-best-database-options-web-development-skills-web-development* (дата звернення 5.11.2023). - Назва з екрана.
33. *Best Database for Web Applications* [Електронний ресурс].- режим доступу: *https://successive.tech/blog/best-database-for-web-applications/* (дата звернення 3.11.2023). - Назва з екрана.
34. *Features of Python Django Framework* [Електронний ресурс].- режим доступу: *https://pythongeeks.org/features-of-python-django-framework/* (дата звернення 3.11.2023). - Назва з екрана.
35. *Features of Bootstrap* [Електронний ресурс].- режим доступу: *https://www.sitesbay.com/bootstrap/bootstrap-features-of-bootstrap* (дата звернення 5.11.2023). - Назва з екрана.
36. *What Is SAP ERP?* [Електронний ресурс].- режим доступу: *https://www.forbes.com/advisor/business/what-is-sap-erp/* (дата звернення 20.11.2023). - Назва з екрана.
37. *What Is Oracle Transportation Management?* [Електронний ресурс].- режим доступу: *https://oracle.argano.com/oracle-blogs/what-is-oracle-transportation-management/* (дата звернення 20.11.2023). - Назва з екрана.
38. *The Complete List of Features of Microsoft Dynamics 365 to Build the Brand & Brain of your Business* [Електронний ресурс].- режим доступу: *https://medium.com/@siddharth.parakh/the-complete-list-of-features-of-microsoft-d*

*ynamics-365-to-build-the-brand-brain-of-your-business-9974baf62321* (дата звернення 20.11.2023). - Назва з екрана.

39. *IBM® Sterling Transportation Management System* [Електронний ресурс].- режим доступу:*chromeextension://mhnlakgilnojmhinhkckjpncpbhabphi/pages/pdf/web/viewer.html?file=https%3A%2F%2Fpublic.dhe.ibm.com%2Fsoftware%2Fdata%2Fsw-library%2FChemLogix_Data_Sheet_8-16-12.pdf* (дата звернення 20.11.2023). - Назва з екрана.
40. *Understanding Workday software: Features and Benefits* [Електронний ресурс].- режим доступу: *https://www.linkedin.com/pulse/understanding-workday-software-features-benefits-zeneesha/* (дата звернення 11.11.2023). - Назва з екрана.

1.

**Лістинг коду**

model.py

```
from django.db import models


class Employee(models.Model):
    position = models.CharField(max_length=255)
    username = models.CharField(max_length=255)
    first_name = models.CharField(max_length=255)
    last_name = models.CharField(max_length=255)
    email = models.EmailField(max_length=255)
    password = models.CharField(max_length=255)
    phone_number = models.CharField(max_length=15)
    qualifications = models.TextField(blank=True)
    certifications = models.CharField(max_length=255, blank=True)
    training_history = models.TextField(blank=True)

    class Meta:
        verbose_name = "employee"
        verbose_name_plural = "employees"
        ordering = ["username"]

    def __str__(self):
        return f"{self.username} ({self.first_name} {self.last_name})"
```

```
class Tag(models.Model):
    name = models.CharField(max_length=255, unique=True)

    def __str__(self):
        return self.name


class Task(models.Model):
    PRIORITY_CHOICES = (
        ("High", "High"),
        ("Medium", "Medium"),
        ("Low", "Low"),
    )
    STATUS_CHOICES = (
        ("Pending", "Pending"),
        ("In Progress", "In Progress"),
        ("Completed", "Completed"),
    )
    title = models.CharField(max_length=255, default="Need to do")
    description = models.TextField()
    datetime = models.DateTimeField(auto_now_add=True)
```

```
    deadline = models.DateTimeField(blank=True, null=True)
    priority = models.CharField(
        max_length=255, choices=PRIORITY_CHOICES, default="Medium"
    )
    is_done = models.BooleanField(default=False)
    assigned_employees = models.ManyToManyField(Employee)
    tags = models.ManyToManyField(Tag, related_name="tasks")
    flight = models.ForeignKey(
        "Flight", on_delete=models.CASCADE, null=True, blank=True
    )
    status = models.CharField(max_length=255, choices=STATUS_CHOICES,
default="Pending")

    class Meta:
        ordering = ["datetime"]

    def __str__(self):
        return f"{self.description} (deadline: {self.deadline})"


class Flight(models.Model):
    departure_airport = models.CharField(max_length=255)
    arrival_airport = models.CharField(max_length=255)
```

```
    departure_date_time = models.DateTimeField()
    arrival_date_time = models.DateTimeField()
    aircraft_type = models.CharField(max_length=255)
    captain = models.ForeignKey(
        Employee,
        on_delete=models.SET_NULL,
        related_name="captain_of_flights",
        null=True,
        blank=True,
    )
    crew = models.ManyToManyField(Employee, related_name="crew_of_flights")

    def __str__(self):
        return f"{self.departure_airport} - {self.arrival_airport}
({self.departure_date_time})"



class Shift(models.Model):
    employee = models.ForeignKey(Employee, on_delete=models.CASCADE)
    start_time = models.DateTimeField()
    end_time = models.DateTimeField()
    break_time = models.DurationField(null=True, blank=True)
    associated_tasks = models.ManyToManyField(Task, blank=True)
```

```
    def __str__(self):
        return f"{self.employee.username} - {self.start_time} - {self.end_time}"


class Equipment(models.Model):
    type = models.CharField(max_length=255)
    model = models.CharField(max_length=255)
    serial_number = models.CharField(max_length=255)
    maintenance_history = models.TextField(blank=True)
    assigned_location = models.CharField(max_length=255)

    def __str__(self):
        return f"{self.type} - {self.model} ({self.serial_number})"


class PerformanceReview(models.Model):
    employee = models.ForeignKey(Employee, on_delete=models.CASCADE)
    reviewer = models.ForeignKey(
        Employee,
        on_delete=models.SET_NULL,
        related_name="reviewer_of_reviews",
        null=True,
```

```
)
date = models.DateField()
review_text = models.TextField()


def __str__(self):
    return f"{self.employee.username} - {self.date}"
```