

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавра

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ОБЛІКУ РОБОТИ

(назва теми)

ВІДДІЛУ ОХОРОНИ ТОРГІВЕЛЬНОГО ПІДПРИЄМСТВА
SOFTWARE FOR RECORDING THE WORK OF THE SECURITY
DEPARTMENT OF A COMMERCIAL ENTERPRISE

Виконав(ла): студент(ка) 4 курсу, групи КНТ-120

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ТОЦЬКИЙ Е.Ф.

(ПРІЗВИЩЕ та ініціали)

Керівник ПОЛЯКОВ М.О.

(ПРІЗВИЩЕ та ініціали)

Рецензент КАЗУРОВА А.Є.

(ПРІЗВИЩЕ та ініціали)

2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет _____ КНТ _____
Кафедра _____ програмних засобів _____
Ступінь вищої освіти _____ бакалавр _____
Спеціальність _____ 121 Інженерія програмного забезпечення _____
(код і найменування)
Освітня програма (спеціалізація) _____ Інженерія програмного забезпечення _____
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф. _____
Сергій СУББОТІН
« _____ » _____ 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ТОЦЬКОГО Едуарда Феліксовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення обліку роботи відділу охорони торгівельного підприємства. Software for Recording the Work of the Security Department of a Commercial Enterprise

керівник проєкту (роботи) д.т.н., професор ПОЛЯКОВ Михайло Олексійович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від « 24 » квітня 2024 року № 168

2. Строк подання студентом проєкту (роботи) 10 червня 2024 року

3. Вихідні дані до проєкту (роботи) технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Розробка архітектури програми. 3. Реалізація програми. 4. Експлуатація та тестування програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ПОЛЯКОВ М.О., професор		
Нормоконтроль	КАМІНСЬКА Ж.К., асистент		

7. Дата видачі завдання « 15 » квітня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2 тиждень	Розділ 1
3	Розробка архітектури програми	3 тиждень	Розділ 2
4	Розробка програми	4-5 тиждні	Розділ 3
5	Тестування та експериментальне дослідження програми	6 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	7 тиждень	Додатки
7	Захист роботи	8 тиждень	

Студент(ка)

_____ Едуард ТОЦЬКИЙ
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

_____ Михайло ПОЛЯКОВ
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 100 с., 23 рис., 2 табл., 3 дод., 11 джерел.

ВІДДІЛ ОХОРОНИ, ТОРГІВЕЛЬНЕ ПІДПРИЄМСТВО, ОБЛІК, ВЕБЗАСТОСУНОК, ФРЕЙМВОРК.

Об'єкт роботи – процес обліку роботи відділу охорони торговельного підприємства. Предмет роботи – програмне забезпечення обліку роботи відділу охорони торговельного підприємства.

Метою роботи є розробка програмного забезпечення обліку роботи відділу охорони торговельного підприємства для забезпечення обґрунтованих рішень щодо керування роботою відділу.

Матеріали, методи та технічні засоби: мова Python, база даних PostgreSQL, фреймворк Django.

Результати. У роботі забезпечено всі етапи розробки програми обліку роботи відділу охорони торговельного підприємства, включаючи вибір технологій розробки з обґрунтуванням рішень, проектування самої програми, програмну реалізацію функціональних можливостей. Це дозволило створити програму, здатну до накопичення даних стосовно різних сторін обліку роботи, включаючи облік робочого часу, виконання завдань, виявлених інцидентів з можливістю подальшого опрацювання накопичених даних.

Висновки. Розроблена програма призначена для забезпечення обліку роботи відділу охорони торговельного підприємства, включаючи реєстрацію перебування працівників на робочому місці, графіків роботи, визначення завдань з охорони, інцидентів з безпеки, які сталися, отриманих поставок та вибуття товарів, реєстрації видачі та повернення обладнання.

Галузь використання. Організація роботи відділу охорони торговельного підприємства.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 100 pages, 23 figures, 2 tables, 3 appendixes, 11 sources.

SECURITY DEPARTMENT, COMMERCIAL ENTERPRISE,
RECORDING, WEB APPLICATION, FRAMEWORK.

The object of work is the process of recording the work of the security department of a commercial enterprise. The subject of the work is software for recording the work of the security department of a commercial enterprise.

The purpose of the work is the development of recording software for the work of the security department of a commercial enterprise to provide comprehensive solutions for managing the work of the department.

Materials, methods and technical means: Python language, PostgreSQL database, Django framework.

The results. The work covers all stages of the development of a program for recording the work of the security department of a commercial enterprise, including the selection of development technologies with the justification of decisions, the design of the program itself, and the software implementation of functional requirements. This made it possible to create a program capable of accumulating data regarding various aspects of work recording, including recording of working hours, task performance, detected incidents with the possibility of further processing of the accumulated data.

Conclusions. The developed program is designed to provide recording for the work of the security department of a commercial enterprise, including registration of employees' presence at the workplace, work schedules, determination of security tasks, security incidents that have occurred, received supplies and disposal of goods, registration of issuance and return of equipment.

Field of use. Organization of the work of the security department of the commercial enterprise.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Аналіз предметної області.....	9
1.1 Аналіз програмних застосунків обліку роботи відділу охорони торгівельного підприємства	9
1.2 Порівняльний аналіз розробки та аналогів	11
1.3 Висновки за розділом 1	13
2 Розробка архітектури програми.....	14
2.1 Вибір засобів проєктування	14
2.2 Сценарії використання програми	15
2.3 Структура бази даних	19
2.4 Висновки за розділом 2	39
3 Реалізація програми	40
3.1 Визначення порядку роботи з основними сутностями	40
3.2 Результати реалізації програмних одиниць застосунку.....	41
3.3 Висновки за розділом 3	59
4 Експлуатація та тестування програми	61
4.1 Призначення програми	61
4.2 Умови виконання програми	61
4.3 Виконання і тестування програми.....	61
4.4 Висновки за розділом 4	66
Висновки	67
Перелік джерел посилання	68
Додаток А Технічне завдання	69
Додаток Б Текст програми	74
Додаток В Слайди презентації.....	94

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

QR – Quick Response;

БД – база даних;

ПЗ – програмне забезпечення.

ВСТУП

Результати роботи відділу охорони торгівельного підприємства є важливими для забезпечення фінансових результатів роботи всього підприємства. Оскільки таке підприємство забезпечує продаж товарів, то зловмисники можуть прагнути заволодіти ними безкоштовно, таким чином наносячи збитки підприємству. Будь-які такі випадки мають виявлятися, обліковуватися.

Однак виявлення не є простим процесом. Необхідно розподіляти завдання між працівниками, контролювати їх наявність на робочому місці, тобто виконувати цілий ряд завдань, які повинні мати вираження у вигляді функціональних можливостей програмного забезпечення (ПЗ), оскільки без ПЗ неможливо керувати такими процесами у великому, середньому підприємстві або навіть малому підприємстві, але на тривалому часовому горизонті. Тому в роботі необхідно проаналізувати аналоги програми, визначити ті функції, які зможуть забезпечити підтримку діяльності відділу охорони торгівельного підприємства для забезпечення ефективної роботи. Саме акцент на обліку тих робочих процесів, які є характерними для відділу охорони торгівельного підприємства, має бути зроблений в роботі.

Зважаючи на вказані фактори, метою роботи є розробка ПЗ обліку роботи відділу охорони торгівельного підприємства для забезпечення обґрунтованих рішень щодо керування роботою відділу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз програмних застосунків обліку роботи відділу охорони торговельного підприємства

Програмні застосунки, які використовуються для підтримки роботи охоронного підприємства або відділу охорони торговельного підприємства, можна розділити на цілий набір категорій:

- програмні застосунки для забезпечення відеоспостереження;
- програмні застосунки для забезпечення контролю доступу до приміщень;
- програмні застосунки для обліку робочого часу;
- програмні застосунки для диспетчеризації;
- програмні застосунки для автоматизованого оповіщення працівників відділу охорони;
- програмні застосунки для керування електронними замками;
- програмні застосунки для надання аналітики [1].

Відділ охорони торговельного підприємства загалом потребує весь спектр такого ПЗ, проте в даній роботі акцент зроблений на створенні застосунку, який реалізує базові можливості стосовно обліку робочого часу, певних засобів диспетчеризації та оповіщення працівників, що має дозволити підтримувати координацію роботи відділу на основі засобів обліку.

За результатами аналізу було виявлено цілий ряд застосунків, які можуть стосуватися забезпечення цих функціональних можливостей, включаючи зокрема Security Officer Reporting Software [2], GuardsPro [3]. Проте саме GuardsPro (рис. 1.1) можна вважати прототипом ПЗ, яке має створюватися в роботі, оскільки воно як раз здатне забезпечити логіку функціональних можливостей, описану вище. Тому було детально проаналізовано можливості даної програми, щоб виявити ті, які будуть важливими, корисними у цій сфері.

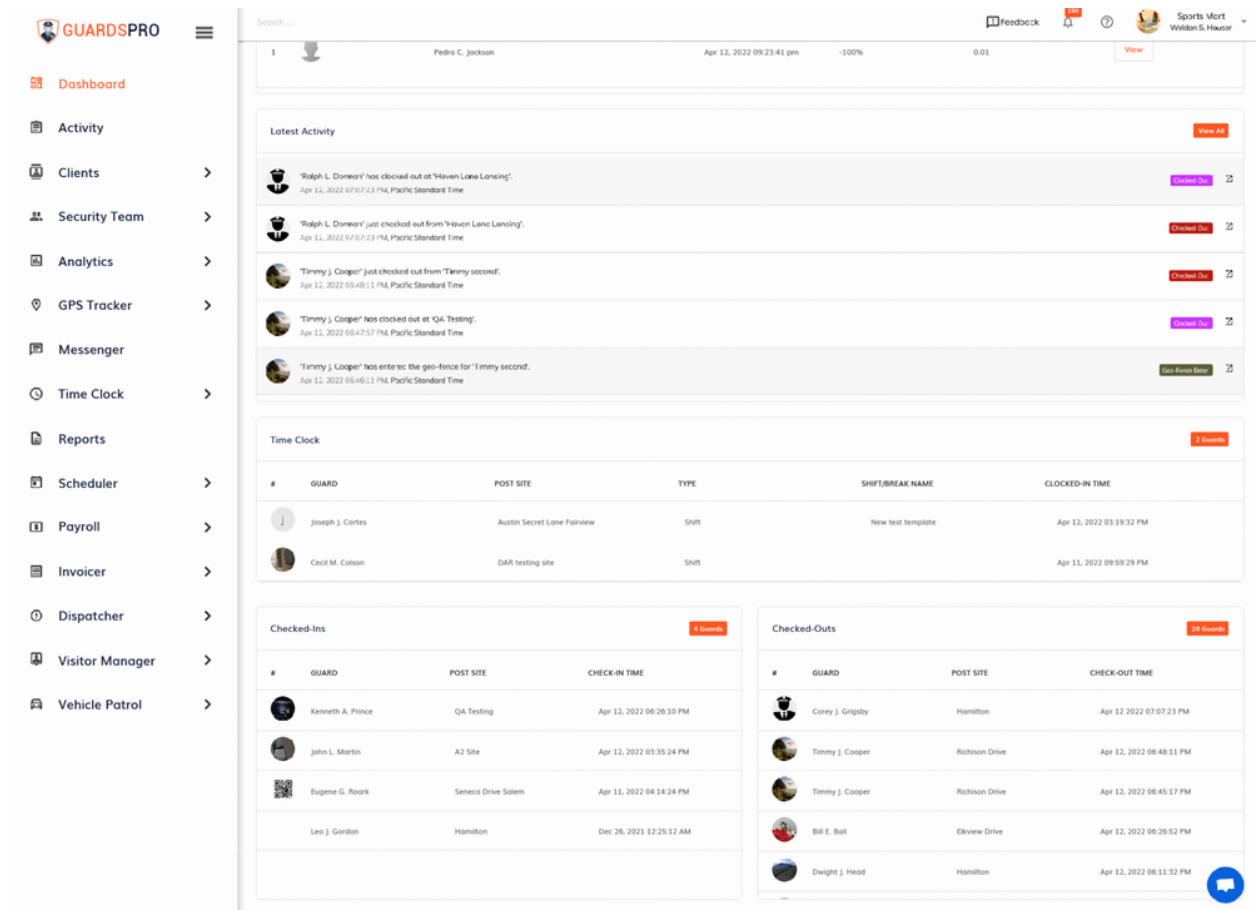


Рисунок 1.1 – Вебзастосунок GuardsPro [3]

Функціональні можливості вебзастосунку GuardsPro включають наступні:

- облік робочого часу, проведеного на певному місці працівником служби охорони;
- взаємодія на основі повідомлень між працівниками служби охорони та працівниками, які керують підтримують роботу служби;
- відслідковування виконаної доставки;
- визначення змін, за якими працюють працівники служби охорони;
- реєстрація, створення звітів про інциденти з безпеки, які виникають (рис. 1.2);
- підтвердження змін працівниками, отримання повідомлень про такі підтвердження;
- встановлення завдань для виконання працівникам служби безпеки;
- реєстрація відвідувачів;

- визначення маршрутів патрулювання за рухом автомобіля;
- відслідковування знаходження працівників служби безпеки на поточний момент [4].

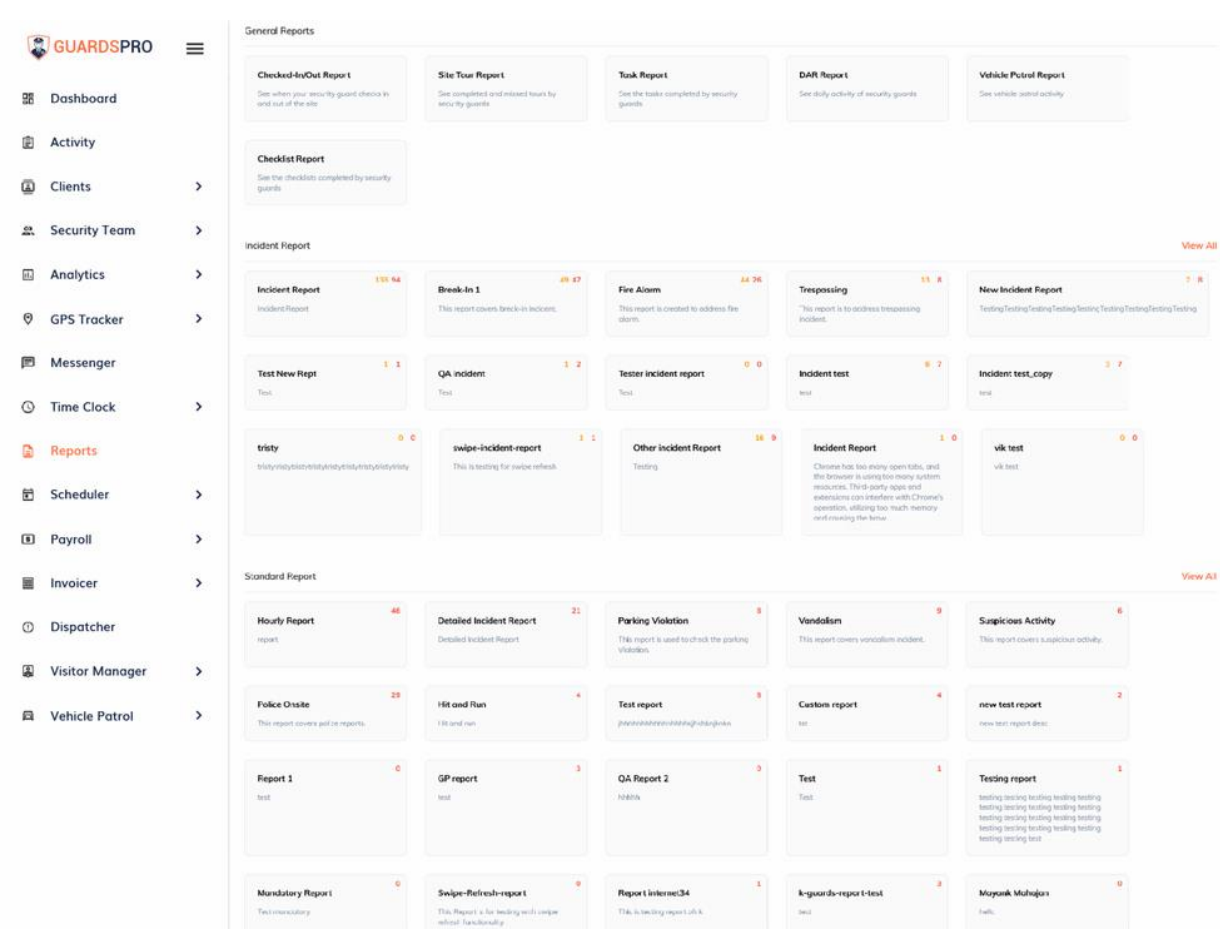


Рисунок 1.2 – Звіти про інциденти в сфері безпеки у застосунку GuardsPro [5]

Основним недоліком даного застосунку можна вважати наявність плати за використання, при чому плата залежить від кількості користувачів і є місячною. Тобто оплата реалізується не раз, а постійно.

1.2 Порівняльний аналіз розробки та аналогів

ПЗ обліку роботи відділу охорони торговельного підприємства, що позначається також як розробка, порівняно з прототипом – застосунком GuardsPro (табл. 1.1).

Таблиця 1.1 – Порівняння ПЗ обліку роботи відділу охорони торгівельного підприємства з аналогом

Критерій	GuardsPro	Розробка
Призначення завдань з охорони працівникам	Так	Так
Визначення змін роботи працівників	Так	Так
Визначення інцидентів з безпеки	Так	Так
Внесення постачань товарів	Так	Так
Надсилання повідомлень між працівниками	Так	Так
Реєстрація переміщення обладнання	Ні	Так
Визначення тренувань персоналу	Ні	Так
Оплата за використання	Щомісячна за кожного користувача	Відсутня

Слід відзначити, що за результатами аналізу функції призначення завдань з охорони працівникам та супутні до цього функції, визначення змін роботи працівників, визначення інцидентів з безпеки, внесення постачань товарів, надсилання повідомлень між працівниками віднесено до базових. Окрім того вирішено додати можливість реєстрації переміщення обладнання, що має включати цілий комплекс функцій, включаючи реєстрацію постачання, реєстрацію видачі обладнання працівнику, повернення обладнання. Також виокремлено визначення тренувань персоналу, що в одній програмі дозволяє за працівником одразу бачити, коли в нього завершується дія попереднього тренування та коли потрібно організувати наступне, щоби продовжувати мати

кваліфікованого працівника і надалі. Також важливою відмінністю є те, що розробка має бути безкоштовною, а GuardsPro є платним рішенням, при чому плата є щомісячною за кожного користувача, тобто чим більша кількість працівників у відділ охорони торговельного підприємства, тим більше необхідно сплачувати. Зниження витрат може бути важливим завданням особливо в умовах нестабільності економічної ситуації, яка зараз є в Україні.

1.3 Висновки за розділом 1

Завдання для виконання в роботі:

- проаналізувати предметну область обліку роботи відділу охорони торговельного підприємства;
- визначити засоби проєктування програми;
- виділити вимоги до ПЗ обліку роботи відділу охорони торговельного підприємства;
- виконати реалізацію ПЗ обліку роботи відділу охорони торговельного підприємства.

У першому розділі виконано аналіз предметної області обліку роботи відділу охорони торговельного підприємства: визначено основні категорії програмних застосунків, які можуть застосовуватися для підтримки роботи відділу охорони, встановлено ті категорії, які найбільше відповідають сукупній підтримці обліку роботи, визначено програмні аналоги з визначенням основного прототипу – GuardsPro, визначено його функціональні можливості, виділено відмінності розробки від цього застосунку.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

2.1 Вибір засобів проєктування

Засоби вибирались як для проєктування, так і розробки, оскільки вони забезпечують цілісний комплекс розробки на рівні фреймворку для веброзробки. Вибір включав Django [6]-[7] і Tornado [8]-[9] (табл. 2.1).

Таблиця 2.1 – Вибір фреймворку для розробки ПЗ обліку роботи відділу охорони торгівельного підприємства

Критерій	Tornado	Django
Зрілість технології	Менша	Більша
Продуктивність роботи	Краща за надвисоких навантажень	Висока
Підтримка опрацювання даних	Підтримує різні системи керування базами даних, об'єктно-реляційні відображення	Має потужну систему об'єктно-реляційного відображення, підтримує різні системи керування базами даних
Складність вивчення можливостей	Більша	Менша
Підтримка	Гірша	Краща
Безпека	Потребує врахування відповідних прийомів створення коду	Застосовані найкращі відомі практики

Django поступається Tornado у випадку роботи в умовах надпотужних навантажень, коли величезна кількість запитів обробляється одночасно, проте в умовах роботи відділу охорони торгівельного підприємства така кількість запитів є досить обмеженою, тож це не є проблемою і недоліком. З іншого боку

Django пропонує більш зрілу технологію, краще опрацювання даних, більшу безпеку, меншу складність вивчення можливостей, що є більшою у Tornado через необхідність забезпечення роботи в умовах надпотужних навантажень. Тож обрано Django.

2.2 Сценарії використання програми

Будь-який користувач, який використовує програму, має бути авторизованим спершу, інакше доступ до функції програми не має забезпечуватися. Реєстрація має виконуватися тільки працівниками, які вже авторизовані в програмі та які мають на це право. Враховуючи це та підтримку інших функцій, всіх користувачів програми розділено на роль базового користувача, який фактично є виконавцем завдань з охорони, та користувачами з правом керувати, який може визначати завдання, продивлятися дані стосовно різних працівників відділу охорони торгівельного підприємства.

Перелік потоку сценаріїв базового користувача:

- відображення всіх власних тренувань;
- визначення власного графіку змін роботи;
- відображення поточного графіку змін роботи всього відділу охорони;
- реєстрація на робочому місці;
- відображення власних пропусків роботи;
- відображення власних завдань з охорони;
- відображення даних профілю працівника відділу охорони;
- авторизація працівника відділу охорони;
- відображення завдання з охорони для виконання;
- визначення виявленого інциденту з безпеки;
- відображення даних клієнта торгівельного підприємства;
- пошук клієнта торгівельного підприємства;
- додавання даних клієнта торгівельного підприємства (рис. 2.1).

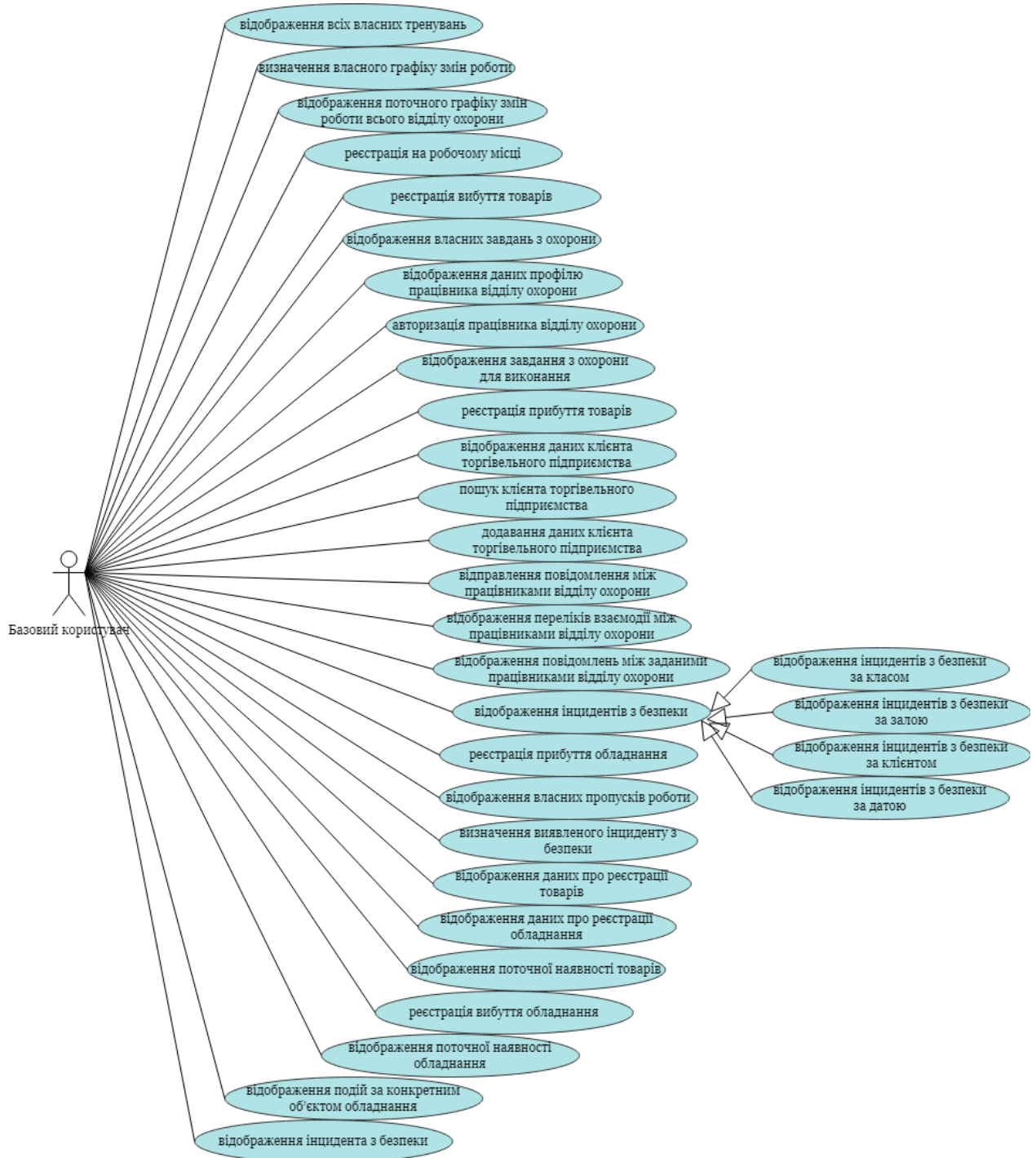


Рисунок 2.1 – Діаграма варіантів використання програми обліку роботи відділу охорони торговельного підприємства базовим користувачем

Також сценарії базового користувача:

- відправлення повідомлення між працівниками відділу охорони;
- відображення переліків взаємодії між працівниками відділу охорони;
- відображення повідомлень між заданими працівниками відділу охорони;

– відображення інцидентів з безпеки за класом, за залом, за клієнтом, за датою;

- реєстрація прибуття або вибуття товарів;
- реєстрація прибуття або вибуття обладнання;
- відображення даних про реєстрації товарів;
- відображення даних про реєстрації обладнання;
- відображення поточної наявності товарів;
- відображення поточної наявності обладнання;
- відображення подій за конкретним об'єктом обладнання.

Перелік потоку сценаріїв користувача з правом керувати додатково до сценаріїв базового користувача:

- відображення переліку працівників відділу охорони;
- додавання даних тренування працівника відділу охорони;
- відображення всіх тренувань працівника відділу охорони;
- відображення всіх тренувань всіх працівників відділу охорони;
- визначення графіку змін роботи працівника відділу охорони;
- відображення пропусків роботи працівником;
- редагування даних клієнта торговельного підприємства;
- відображення поточної наявності працівників на роботі загалом та за залами торговельного підприємства;
- керування залами торговельного підприємства;
- відображення розподілу завдань з охорони на заданий момент часу;
- призначення завдання з охорони для виконання працівнику;
- відображення завдань з охорони працівника;
- редагування даних профілю працівника відділу охорони;
- керування станом виконання завдання з охорони;
- визначення переліку типових завдань працівника;
- реєстрація працівника відділу охорони;
- відображення інцидентів з безпеки за працівником;
- редагування даних тренування працівника відділу охорони;

- пошук працівника відділу охорони;
- керування станом інциденту з безпеки;
- керування переліком постачальників;
- керування переліком товарів;
- реєстрація видачі обладнання працівнику відділу охорони;
- реєстрація повернення обладнання працівником (рис. 2.2).

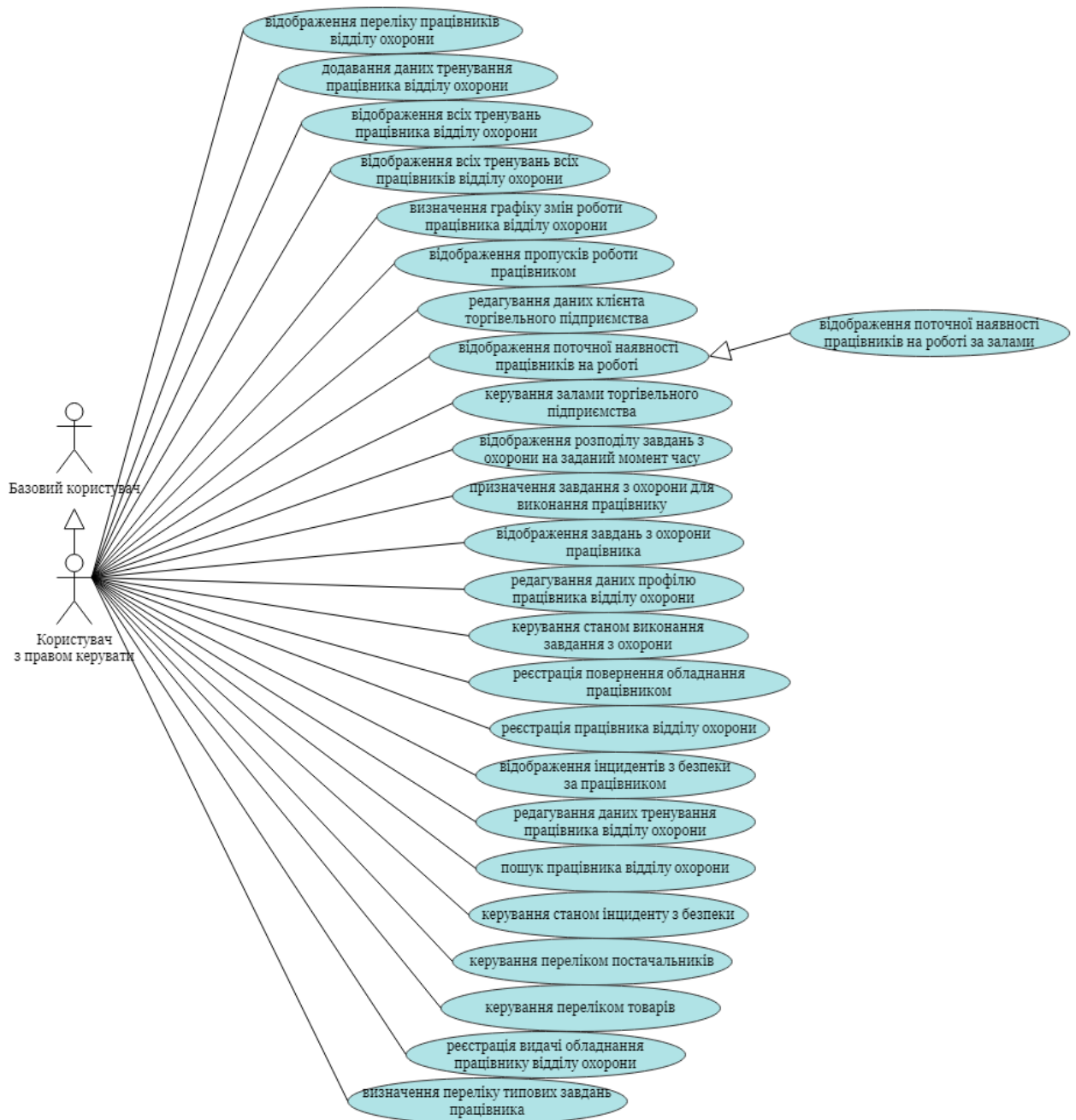


Рисунок 2.2 – Діаграма варіантів використання програми обліку роботи відділу охорони торговельного підприємства користувачем з правом керувати

2.3 Структура бази даних

База даних (БД) є основою створення цього застосунку, адже в основі забезпечення функціональних вимог є відповідні сутності та дані які ним відповідають. Створено БД та підтримка її роботи забезпечується на основі системи PostgreSQL [10], яка вибрана з врахуванням її потужності, безкоштовності.

Таблиця користувацького облікового запису в БД має назву User та стовпці:

- id – ідентифікатор відповідного користувацького облікового запису;
- username – логін, використовуваний відповідним користувацьким обліковим записом;
- first_name – ім'я людини, що працює за відповідним користувацьким обліковим записом;
- last_name – прізвище людини, що працює за відповідним користувацьким обліковим записом;
- email – адреса електронної пошти, використовувана відповідним користувацьким обліковим записом;
- password – пароль, використовуваний відповідним користувацьким обліковим записом;
- is_staff – можливість людини, що працює з відповідним користувацьким обліковим записом, працювати з адміністративною панеллю;
- is_active – можливість людини, що працює з відповідним користувацьким обліковим записом, використовувати його для роботи (використовується замість вилучення);
- last_login – дата, час, коли зафіксоване останнє використання відповідного користувацького облікового запису;
- date_joined – дата, час, коли зафіксоване перше використання відповідного користувацького облікового запису.

Таблиця працівників відділу охорони торгівельного підприємства в БД має назву Guardianuser та стовпці:

- id – ідентифікатор працівника відділу охорони торгівельного підприємства;
- user – користувацький обліковий запис, що використовується відповідним працівником відділу охорони торгівельного підприємства;
- jobphone – робочий номер телефону відповідного працівника відділу охорони торгівельного підприємства;
- personalphone – особистий номер телефону відповідного працівника відділу охорони торгівельного підприємства;
- incompanyfrom – дата, з якої цей користувач є працівником відділу охорони торгівельного цього підприємства;
- photo – фотографія відповідного працівника відділу охорони торгівельного підприємства;
- companylevel – рівень доступу під час роботи відповідного працівника відділу охорони торгівельного підприємства: виконавець завдань або доступ до встановлення завдань до зміни станів зафіксованих спостережень, об'єктів;
- about – особиста характеристика відповідного працівника відділу охорони торгівельного підприємства;
- shortname – коротке ім'я відповідного працівника відділу охорони торгівельного підприємства (наприклад, може використовуватися прізвище та ініціали або інша прийнята на підприємстві форма);
- blocked – блокування роботи цього користувача як працівника відділу охорони торгівельного підприємства або відсутність цього блокування.

Кожен працівник відділу охорони торгівельного підприємства може використовувати один користувацький обліковий запис, а спільне використання одного користувацького облікового запису різними працівниками заборонено, дозволено тільки особисте, тому зв'язані таблиці Guardianuser і User зв'язком один-до-одного.

Таблиця навчання працівників відділу охорони торгівельного підприємства в БД має назву Guardiantrain та стовпці:

- id – ідентифікатор навчання працівника відділу охорони торгівельного підприємства;
- guardianuser – відповідний працівник відділу охорони торгівельного підприємства, який пройшов навчання;
- trainfrom – дата, коли почалось навчання відповідного працівника відділу охорони торгівельного підприємства;
- trainto – дата, коли закінчилось навчання відповідного працівника відділу охорони торгівельного підприємства: може бути порожнє значення, якщо навчання не завершилось;
- trainresult – отриманий результат навчання за відповідною програмою працівником відділу охорони торгівельного підприємства (визначається текстом);
- trainprogram – програма, за якою відбувалось навчання відповідного працівника відділу охорони торгівельного підприємства (визначається текстом);
- certificatenumbr – номер сертифікату, що відповідним працівником відділу охорони торгівельного підприємства було отримано за результатами навчання;
- certificatefile – файл сертифікату, що відповідним працівником відділу охорони торгівельного підприємства було отримано за результатами навчання;
- nextraindate – орієнтовна дата, коли відповідний працівник відділу охорони торгівельного підприємства має пройти наступне навчання після даного для забезпечення необхідної для роботи кваліфікації.

Кожен працівник відділу охорони торгівельного підприємства може проходити в різний час різне навчання, тобто навчань може бути багато, а кожне навчання реалізується особисто, тобто стосовно тільки одного

працівника відділу охорони торговельного підприємства, тому зв'язані таблиці Guardianuser і Guardiantrain зв'язком один-до-багатьох.

Робота працівників відділу охорони торговельного підприємства реалізується змінами. Але підхід може бути або таким, що робота відбувається за тижнями (умовно робочі дні та вихідні на тижні), або за принципом чергувань, коли після певної кількості робочих змін (днів) поспіль слідує певна кількість неробочих змін (днів) поспіль.

Таблиця тижневого варіанта змін, за яким працює працівник відділу охорони торговельного підприємства, в БД має назву Guardianweekshift та стовпці:

- id – ідентифікатор тижневого варіанта змін, який використовується щодо роботи працівника відділу охорони торговельного підприємства;
- guardianuser – працівник відділу охорони торговельного підприємства, який працює за цим тижневим варіантом змін;
- fromdate – дата, з якої застосовується цей тижневий варіант змін для відповідного працівника відділу охорони торговельного підприємства;
- weekday – номер дня тижня, в який така зміна реалізується: 0 – понеділок, 1 – вівторок, 2 – середа, 3 – четвер, 4 – п'ятниця, 5 – субота, 6 – неділя;
- shiftstarttime – час початку зміни за цим тижневим варіантом роботи у відповідний день тижня відповідного працівника відділу охорони торговельного підприємства;
- shiftendtime – час закінчення зміни за цим тижневим варіантом роботи у відповідний день тижня відповідного працівника відділу охорони торговельного підприємства.

Кожен працівник відділу охорони торговельного підприємства може мати різні представлення тижневих варіантів змін на різні дати та різні номери днів тижнів, але кожен такий варіант застосовується лише для одного працівника відділу охорони торговельного підприємства окремо, тому зв'язані таблиці Guardianuser і Guardianweekshift зв'язком один-до-багатьох.

Таблиця варіантів змін за принципом чергування, які використовуються щодо роботи працівників відділу охорони торговельного підприємства, в БД має назву `Guardiandayshift` та стовпці:

- `id` – ідентифікатор варіанта змін за принципом чергування, який використовується щодо роботи працівника відділу охорони торговельного підприємства;

- `guardianuser` – працівник відділу охорони торговельного підприємства, який працює за цим варіантом змін за принципом чергування;

- `fromdate` – дата, з якої застосовується цей варіант змін за принципом чергування для відповідного працівника відділу охорони торговельного підприємства;

- `workdaysnumber` – кількість робочих днів, яку поспіль має відпрацювати працівник цього відділу охорони торговельного підприємства перед отримання вихідних;

- `freedaysnumber` – кількість днів відпочинку, яку поспіль має отримати працівник цього відділу охорони торговельного підприємства перед повторенням робочих днів;

- `shiftstarttime` – час початку зміни за цим варіантом роботи за принципом чергування відповідного працівника відділу охорони торговельного підприємства;

- `shiftendtime` – час закінчення зміни за цим варіантом роботи за принципом чергування відповідного працівника відділу охорони торговельного підприємства.

Кожен працівник відділу охорони торговельного підприємства може мати різні представлення варіантів змін за принципом чергування на різні дати, але кожен такий варіант застосовується лише для одного працівника відділу охорони торговельного підприємства окремо, тому зв'язані таблиці `Guardianuser` і `Guardianweekshift` зв'язком один-до-багатьох.

Таблиця реєстрації працівників відділу охорони торговельного підприємства на робочому місці, включаючи прихід на роботу, закінчення

роботи, початок перерви, закінчення перерви, в БД має назву Guardianregistration та стовпці:

- id – ідентифікатор реєстрації відповідного працівника відділу охорони торговельного підприємства на робочому місці;
- guardianuser – працівник відділу охорони торговельного підприємства, який зареєструвався на робочому місці;
- registrationtime – час, коли зафіксовано реєстрацію відповідного працівника відділу охорони торговельного підприємства на робочому місці;
- registrationtype – тип реєстрації на робочому місці: прихід на роботу, закінчення роботи, початок перерви, закінчення перерви.

Таблиця залив торговельного підприємства в БД має назву Commenterprisehall та стовпці:

- id – ідентифікатор зали торговельного підприємства;
- enterhallname – назва, якою охарактеризовано залу торговельного підприємства;
- enterhallphoto – фотографія зали торговельного підприємства;
- enterhallabout – опис зали торговельного підприємства, що характеризує його.

Таблиця завдань з охорони, які можуть встановлюватися працівникам відділу охорони торговельного підприємства, в БД має назву Securetask та стовпці:

- id – ідентифікатор завдання з охорони, яке може встановлюватися працівнику відділу охорони торговельного підприємства на виконання;
- securetaskname – назва (формулювання) завдання з охорони, яке може встановлюватися працівнику відділу охорони торговельного підприємства на виконання.

Таблиця типових завдань з охорони, які визначаються зазвичай працівникам відділу охорони торговельного підприємства, в БД має назву Guardiantypicaltask та стовпці:

– id – ідентифікатор типового завдання, яке визначається зазвичай відповідному працівнику відділу охорони торгівельного підприємства;

– guardianuser – працівник відділу охорони торгівельного підприємства, якому типово встановлено таке типове завдання з охорони на робочий день;

– securetask – завдання з охорони, яке встановлюється відповідному працівнику відділу охорони торгівельного підприємства на виконання типово;

– commententerprisehall – зала торгівельного підприємства, в якій типово виконує це завдання цей працівник відділу охорони торгівельного підприємства.

Кожен працівник відділу охорони торгівельного підприємства може мати декілька типових завдань з охорони на виконання, але кожне типове завдання з охорони стосується тільки одного конкретного працівника відділу охорони торгівельного підприємства, тобто визначається особисто, тому зв'язані таблиці Guardianuser і Guardiantypicaltask зв'язком один-до-багатьох.

Таблиця завдань з охорони, які фактично визначені працівнику відділу охорони торгівельного підприємства на виконання, в БД має назву Guardiantask та стовпці:

– id – ідентифікатор завдання з охорони, яке фактично визначено працівнику відділу охорони торгівельного підприємства на виконання;

– guardianuser – працівник відділу охорони торгівельного підприємства, якому фактично визначено на виконання відповідне завдання;

– securetask – завдання з охорони, яке встановлене на виконання відповідному працівнику відділу охорони торгівельного підприємства;

– commententerprisehall – зала торгівельного підприємства, в якій відповідний працівник відділу охорони торгівельного підприємства повинен виконувати відповідне завдання з охорони;

– taskstarttime – дата, час, з якого повинен розпочати виконувати це завдання з охорони відповідний працівник відділу охорони торгівельного підприємства;

– taskendtime – дата, час, до якого повинен закінчити виконувати це завдання з охорони відповідний працівник відділу охорони торговельного підприємства;

– taskcomment – коментар до завдання, який встановлений особою, що призначила на виконання це завдання з охорони відповідному працівнику відділу охорони торговельного підприємства;

– managerguardian – особа (користувач програми, який є працівником відділу охорони відповідного рівня), що призначила на виконання це завдання з охорони відповідному працівнику відділу охорони торговельного підприємства.

Кожен працівник відділу охорони торговельного підприємства може мати багато встановлених йому на виконання завдань з охорони, але кожне завдання з охорони, визначене працівнику на виконання, стосується тільки одного конкретного працівника відділу охорони торговельного підприємства, тобто визначається особисто, тому зв'язані таблиці Guardianuser і Guardiantask зв'язком один-до-багатьох за полем guardianuser.

Кожен працівник відділу охорони торговельного підприємства, який має керуючу роль, може встановити багато завдань з охорони на виконання іншим працівникам, але кожне завдання з охорони має тільки одного працівника відділу охорони торговельного підприємства, який його визначив, тому зв'язані таблиці Guardianuser і Guardiantask зв'язком один-до-багатьох за полем managerguardian.

Кожне завдання з охорони виконується в конкретній залі комерційного підприємства, але кожна зала комерційного підприємства в різний час може мати багато завдань з охорони, які віднесені до неї, тому зв'язані таблиці Guardiantask і Commenterprisehall зв'язком один-до-багатьох.

Кожне завдання з охорони, встановлене конкретному працівнику відділу охорони торговельного підприємства на виконання, має тільки одне відповідне загальне завдання з охорони і не може стосуватися декількох завдань одночасно, але кожне загальне завдання з охорони може виставлятися

різним працівникам відділу охорони торговельного підприємства в різний час, тому зв'язані таблиці Securetask і Guardiantask зв'язком один-до-багатьох.

Таблиця станів виконання завдань з охорони працівниками відділу охорони торговельного підприємства в БД має назву Guardiantaskstatus та стовпці:

- id – ідентифікатор зафіксованого стану виконання завдання з охорони працівником відділу охорони торговельного підприємства;
- guardiantask – завдання з охорони, стосовно якого зафіксовано цей стан виконання;
- guardianuser – працівник відділу охорони торговельного підприємства, який зафіксував цей стан виконання завдання з охорони: фіксувати автора змін необхідно через те, що перши станом є той, коли завдання створено відповідальним працівником відділу охорони, відповідно до фіксації цього стану не має ніякого відношення працівник, який буде виконувати це завдання, тому потрібно визначити це поле в таблиці;
- status – стан, який зафіксовано на цей момент при виконанні завдання з охорони працівником відділу охорони торговельного підприємства;
- statustime – дата, час, коли цей стан виконання завдання з охорони був зафіксований;
- statuscomment – коментар до зафіксованого стану виконання завдання з охорони відповідним працівником відділу охорони торговельного підприємства (може і не вказуватися, бути порожнім).

Кожен працівник відділу охорони торговельного підприємства може зафіксувати багато різних станів виконання завдань з охорони, але кожен зафіксований стан виконання завдання з охорони може бути встановлений тільки одним працівником, тому зв'язані таблиці Guardianuser і Guardiantaskstatus зв'язком один-до-багатьох за полем guardianuser.

Кожен зафіксований стан виконання завдання з охорони стосується тільки одного конкретного визначеного працівнику відділу охорони комерційного підприємства завдання, але кожне таке завдання з охорони може

мати багато зафіксованих станів, тому зв'язані таблиці `Guardiantask` і `Guardiantaskstatus` зв'язком один-до-багатьох за полем `guardianuser`.

Таблиця класів інцидентів, які можуть бути виявлені під час проведення охорони торговельного підприємства, в БД має назву `Secureincidentclass` та поля:

- `id` – ідентифікатор класу інциденту, який може бути виявлений під час проведення охорони торговельного підприємства;

- `secureincidentname` – назва класу інциденту, який може бути виявлений під час проведення охорони торговельного підприємства (наприклад, крадіжка).

Таблиця конкретного виявленого під час роботи комерційного підприємства інциденту з безпеки в БД має назву `Secureincident` та поля:

- `id` – ідентифікатор виявленого під час роботи комерційного підприємства інциденту з безпеки;

- `guardiantask` – завдання з охорони, під час виконання якого працівник відділу з охорони торговельного підприємства виявив інцидент з безпеки;

- `commenterprisehall` – зала торговельного підприємства, в якій було виявлено цей інцидент: зала необхідна для того, адже інциденти стаються зазвичай не загально невідомо де (хоча тоді це значення може бути пропущене), а в конкретних залах, проте завдання на охорону, під час виконання якого виявив інцидент працівник, могло бути не специфіковано за залом або могло відбуватися в сусідній залі, так що працівник помітив його;

- `secureincidentclass` – клас інциденту з безпеки, який було виявлено під час роботи комерційного підприємства;

- `incidentstart` – дата, час, коли, на скільки відомо, розпочався інцидент а з безпеки бо коли він стався (якщо він був не тривалим);

- `incidentend` – дата, час, коли, на скільки відомо, закінчився інцидент з безпеки;

- `incidentobsrvetime` – дата, час, коли інцидент з безпеки було виявлено;

– incidentabout – опис ситуації, що характеризує інцидент з безпеки, який стався;

– priority – пріоритет, який встановлений цьому інциденту з безпеки, тобто те, на скільки він є критичним.

Кожне завдання з охорони може включати декілька виявлених інцидентів з безпеки, але кожен інцидент може бути зафіксований у програмі тільки стосовно одного завдання з охорони, під час виконання якого був виявлений, тому зв'язані таблиці Guardiantask і Secureincident зв'язком один-до-багатьох.

Кожен інцидент з безпеки може бути виявлений тільки в одній залі торгівельного підприємства, але в кожній залі торгівельного підприємства може бути в різний час виявлено декілька інцидентів з безпеки, тому зв'язані таблиці Commenterprisehall і Secureincident зв'язком один-до-багатьох.

Кожен інцидент з безпеки може належати тільки одному класу такого інциденту, а відповідно кожен клас інцидентів з безпеки може мати багато різних випадків, тому зв'язані таблиці Secureincidentclass і Secureincident зв'язком один-до-багатьох.

Таблиця з файлами, долученими до опису інциденту з безпеки, в БД має назву Secureincidentfile та поля:

- id – ідентифікатор файлу, долученого до опису інциденту з безпеки;
- secureincident – інцидент з безпеки, до опису якого долучено цей файл;
- file – файл, який долучено до опису інциденту з безпеки (його безпосередній вміст);
- filecomment – коментар до файлу, який долучено до опису інциденту з безпеки;
- guardianuser – працівник відділу охорони комерційного підприємства, який долучив до опису інциденту з безпеки цей файл;

– filepriority – пріоритет, який встановлений цьому файлу серед всіх долучених до опису інциденту з безпеки матеріалів, включаючи фотографії, посилання та файли.

Кожен інцидент з безпеки може мати багато файлів, які додані до його опису, але кожен файл належить тільки одному інциденту з безпеки, тому зв'язані таблиці Secureincident і Secureincidentfile зв'язком один-до-багатьох.

Таблиця з фотографіями, долученими до опису інциденту з безпеки, в БД має назву Secureincidentphoto та поля:

– id – ідентифікатор фотографії, долученої до опису інциденту з безпеки;

– secureincident – інцидент з безпеки, до опису якого долучено цю фотографію;

– photo – фотографія, яку долучено до опису інциденту з безпеки (її безпосередній вміст);

– photocomment – коментар до фотографії, яку долучено до опису інциденту з безпеки;

– guardianuser – працівник відділу охорони комерційного підприємства, який долучив до опису інциденту з безпеки цю фотографію;

– photopriority – пріоритет, який встановлений цій фотографії серед всіх долучених до опису інциденту з безпеки матеріалів, включаючи фотографії, посилання та файли.

Кожен інцидент з безпеки може мати багато фотографій, які додані до його опису, але кожна фотографія належить тільки одному інциденту з безпеки, тому зв'язані таблиці Secureincident і Secureincidentphoto зв'язком один-до-багатьох.

Таблиця з посиланнями на вебресурси, долучені до опису інциденту з безпеки, в БД має назву Secureincidenturl та поля:

– id – ідентифікатор посилання на вебресурс, долученого до опису інциденту з безпеки;

– `secureincident` – інцидент з безпеки, до опису якого долучено це посилання;

– `url` – посилання на вебресурс, яке долучено до опису цього інциденту з безпеки;

– `urlcomment` – коментар до посилання на вебресурс, який долучено до опису інциденту з безпеки;

– `guardianuser` – працівник відділу охорони комерційного підприємства, який долучив до опису інциденту з безпеки це посилання на вебресурс;

– `urlpriority` – пріоритет, який встановлений цьому посиланню на вебресурс серед всіх долучених до опису інциденту з безпеки матеріалів, включаючи фотографії, посилання та файли.

Кожен інцидент з безпеки може мати багато посилань на вебресурс, які додані до його опису, але кожне посилання на вебресурс належить тільки одному інциденту з безпеки, тому зв'язані таблиці `Secureincident` і `Secureincidenturl` зв'язком один-до-багатьох.

Таблиця станів інцидентів з безпеки в БД має назву `Secureincidentstatus` та поля:

– `id` – ідентифікатор зафіксованого стану інциденту з безпеки;

– `secureincident` – інцидент з безпеки, стан якого зафіксовано;

– `guardianuser` – працівник відділу охорони торгівельного підприємства, який встановив цей стан інциденту з безпеки;

– `status` – стан, який у даному випадку зафіксовано для інциденту з безпеки;

– `statustime` – дата, час, коли було зафіксовано цей стан інциденту з безпеки;

– `statuscomment` – коментар до зафіксованого стану інциденту з безпеки відповідним працівником відділу охорони торгівельного підприємства (може і не вказуватися, бути порожнім).

Кожен інцидент з безпеки може мати багато станів, а кожен стан є станом тільки одного інциденту з безпеки, тому зв'язані таблиці Secureincident і Secureincidentstatus зв'язком один-до-багатьох.

Тільки один працівник відділу охорони торговельного підприємства може встановити конкретний стан інциденту з безпеки, але при цьому працівник може встановити за весь час багато станів інцидентів з безпеки, тому зв'язані таблиці Guardianuser і Secureincidentstatus зв'язком один-до-багатьох.

Таблиця клієнта (або просто відвідувача) торговельного підприємства в БД має назву Commenterpriseclient та поля:

- id – ідентифікатор клієнта (відвідувача) торговельного підприємства;
- clientname – ім'я клієнта (відвідувача) торговельного підприємства;
- clientdocument – документ, на основі якого ідентифіковано клієнта (відвідувача) торговельного підприємства;
- additionaldocument – додатковий документ, на основі якого ідентифіковано клієнта (відвідувача) торговельного підприємства;
- documentfile – файл документа, на основі якого ідентифіковано клієнта (відвідувача) торговельного підприємства;
- date_joined – дата, час, коли вперше було внесено дані про відповідного клієнта (відвідувача) торговельного підприємства;
- clientphoto – фотографія клієнта (відвідувача) торговельного підприємства;
- blocked – позначка про те, що доступ відповідного клієнта (відвідувача) має бути заблокований до торговельного підприємства, тобто його більше не має пропускати відділ охорони.

Таблиця клієнтів, які є причетними до інцидентів з безпеки, в БД має назву Secureincidentclient та поля:

- id – ідентифікатор причетності клієнта торговельного підприємства до інциденту з безпеки;

– `commenterpriseclient` – клієнт торговельного підприємства, який взяв участь в інциденті з безпеки;

– `secureincident` – інцидент з безпеки, в якому взяв участь клієнт торговельного підприємства;

– `guardianuser` – працівник відділу охорони торговельного підприємства, який визначив причетність відповідного клієнта до відповідного інциденту з безпеки;

– `rolecomment` – коментар працівника відділу охорони торговельного підприємства, який визначив причетність відповідного клієнта до відповідного інциденту з безпеки, щодо його участі в інцидентів;

– `addtime` – дата, час, коли інформацію щодо причетності клієнта торговельного підприємства до інциденту з безпеки було внесено.

У кожному інциденті з безпеки можуть взяти участь декілька клієнтів торговельного підприємства, а кожен клієнт може брати участь в різних інциденту з безпеки в різний час, тому зв'язані таблиці `Commenterpriseclient` і `Secureincident` зв'язком багато-до-багатьох, який реалізований таблицею `Secureincidentclient`.

Таблиця взаємодії між працівниками відділу охорони торговельного підприємства в БД має назву `Collaboration` та поля:

– `id` – ідентифікатор взаємодії між працівниками відділу охорони торговельного підприємства;

– `initialguardianuser` – працівник відділу охорони торговельного підприємства, який створив цю взаємодію;

– `receiverguardianuser` – працівник відділу охорони торговельного підприємства, який бере з іншого боку у взаємодії.

Кожен працівник відділу охорони торговельного підприємства може мати багато взаємодій з іншими працівниками, а в кожній взаємодії два учасника, тому зв'язана таблиця `Guardianuser` з собою ж через поля `initialguardianuser` і `receiverguardianuser` зв'язком багато-до-багатьох, який реалізований таблицею `Collaboration`.

Таблиця повідомлень між працівниками відділу охорони торгівельного підприємства в БД має назву Collaborationmessage та поля:

- id – ідентифікатор повідомлення між працівниками відділу охорони торгівельного підприємства;
- collaboration – взаємодія між працівниками відділу охорони торгівельного підприємства, до якої належить повідомлення;
- fullmessage – повний текст повідомлення між працівниками відділу охорони торгівельного підприємства;
- senderguardianuser – працівник відділу охорони торгівельного підприємства, який надіслав це повідомлення;
- receiversaw – позначка про те, що працівник відділу охорони торгівельного підприємства, якому направлено повідомлення, побачив його;
- sendtime – дата, час, коли це повідомлення між працівниками відділу охорони торгівельного підприємства було створено.

Кожна взаємодія між працівниками відділу охорони торгівельного підприємства може включати багато повідомлень, а кожне повідомлення належить одній взаємодії, тому зв'язані таблиці Collaboration і Collaborationmessage зв'язком один-до-багатьох.

Таблиця постачальника в БД має назву Provisor та поля:

- id – ідентифікатор постачальника;
- provisorname – назва компанії-постачальника;
- businessaddress – ділова адреса компанії-постачальника;
- provisorphone – номер телефону компанії-постачальника;
- provisorsite – вебсайт компанії-постачальника.

Таблиця товарів, які продаються, а відповідно і постачаються торгівельному підприємству, в БД має назву Commodity та поля:

- id – ідентифікатор товару, який постачається торгівельному підприємству;
- commodityname – назва товару, який постачається торгівельному підприємству.

Таблиця зареєстрованого постачання або вибуття товарів або обладнання в БД має назву Inoutregistration та поля:

- id – ідентифікатор зареєстрованого постачання або вибуття товарів або обладнання в торгівельну компанію через відділ охорони;
- provisor – постачальник товарів або обладнання, представник якого привіз товари (може бути пропущений, якщо це не постачання, а вибуття);
- provisorcontact – контактна особа, яка привезла від постачальника товари або обладнання, прийом яких або відправлення яких відбулося;
- guardianuser – працівник відділу охорони торгівельного підприємства, який виконав реєстрацію постачання або вибуття товарів або обладнання;
- registertime – дата, час реєстрації постачання або вибуття товарів або обладнання в торгівельну компанію через відділ охорони;
- registercomment – коментар працівника відділу охорони торгівельного підприємства, який виконав реєстрацію постачання або вибуття товарів або обладнання;
- documentumber – номер документа, на основі якого було оформлено постачання або відправлення товару або обладнання;
- documentphoto – фотографія документа, на основі якого було оформлено постачання або відправлення товару або обладнання;
- inout – позначка про те, чи це було постачання товару або обладнання, чи вибуття.

Кожне постачання або вибуття товарів або обладнання реєструється одним працівником відділу охорони торгівельної компанії, але кожен працівник може мати багато різних зареєстрованих ним постачань, тому зв'язані таблиці Guardianuser і Inoutregistration зв'язком один-до-багатьох.

Кожен постачальник може мати багато зареєстрованих постачань його товарів або обладнання, але кожне окреме зареєстроване постачання стосується тільки одного постачальника, тому зв'язані таблиці Provisor і Inoutregistration зв'язком один-до-багатьох.

Таблиця деталей реєстрації постачання або вибуття саме товарів (без обладнання) в БД має назву Commodityregistration та поля:

- id – ідентифікатор деталей реєстрації постачання або вибуття товарів;
- inoutregistration – зареєстроване постачання або вибуття товарів в торгівельну компанію через відділ охорони, яке деталізовано цим записом;
- commodity – товар, який входить в це постачання або вибуття товарів;
- commoditynumber – кількість одиниць цього товару в постачанні або вибутті.

Кожна реєстрація постачання або вибуття товарів може містити різні товари, а кожен товар може входити в різні постачання, тому зв'язані таблиці Inoutregistration і Commodity зв'язком багато-до-багатьох, який реалізований таблицею Commodityregistration.

Таблиця класів обладнання, яке використовується в роботі торгівельного підприємства, в БД має назву Equipmentclass та поля:

- id – ідентифікатор класу обладнання;
- equipmentclassname – назва класу обладнання.

Таблиця обладнання, яке наявне в торгівельному підприємстві та використовується в роботі (не продається), в БД має назву Equipment та поля:

- id – ідентифікатор обладнання, яке наявне в торгівельному підприємстві та використовується в роботі;
- equipmentclass – клас цього обладнання, яке наявне в торгівельному підприємстві та використовується в роботі;
- equipmentname – назва цього обладнання, яке наявне в торгівельному підприємстві та використовується в роботі.

Кожен клас обладнання представлений різним обладнанням, яке наявне в торгівельному підприємстві, але кожне обладнання в торгівельному підприємстві належить одному класу, тому зв'язані таблиці Equipmentclass і Equipment зв'язком один-до-багатьох.

Таблиця конкретних об'єктів обладнання, які використовуються в роботі торгівельного підприємства, в БД має назву Equipmentobject та поля:

– id – ідентифікатор конкретного об'єкта обладнання, який використовуються в роботі торгівельного підприємства;

– equipmentregistration – деталізація постачання обладнання в торгівельну компанію через відділ охорони, за якою цей об'єкт обладнання потрапив у торгівельну компанію;

– inventnumber – інвентарний номер об'єкта обладнання;

– objectphoto – фотографія об'єкта обладнання;

– state – стан об'єкта обладнання.

Кожен конкретний об'єкт обладнання поставляється через одне постачання, а кожне постачання може містити різні об'єкти, тому зв'язані таблиці Equipmentregistration і Equipmentobject зв'язком один-до-багатьох.

Таблиця видачі та використання об'єктів обладнання працівниками відділу охорони торгівельного підприємства в БД має назву Guardianequipment та поля:

– id – ідентифікатор видачі та використання об'єкта обладнання працівником відділу охорони торгівельного підприємства;

– equipmentobject – об'єкт обладнання, який виданий та використовується працівником відділу охорони торгівельного підприємства;

– guardianuser – працівник відділу охорони торгівельного підприємства, який використовує цей об'єкт обладнання;

– usedfrom – дата, час, коли об'єкт обладнання був виданий працівнику відділу охорони торгівельного підприємства;

– usedto – дата, час, коли об'єкт обладнання був повернутий працівником відділу охорони торгівельного підприємства.

Кожен конкретний об'єкт обладнання може надаватися в різний час заново для використання, але кожна видача стосується одного об'єкта окремо, тому зв'язані таблиці Equipmentobject і Guardianequipment зв'язком один-до-багатьох.

Кожен працівник відділу безпеки торгівельного підприємства в різний час може брати різні об'єкти обладнання, але кожен об'єкт видається на

використання одному працівнику, тому зв'язані таблиці Guardianuser і Guardianequipment зв'язком один-до-багатьох (рис. 2.3).

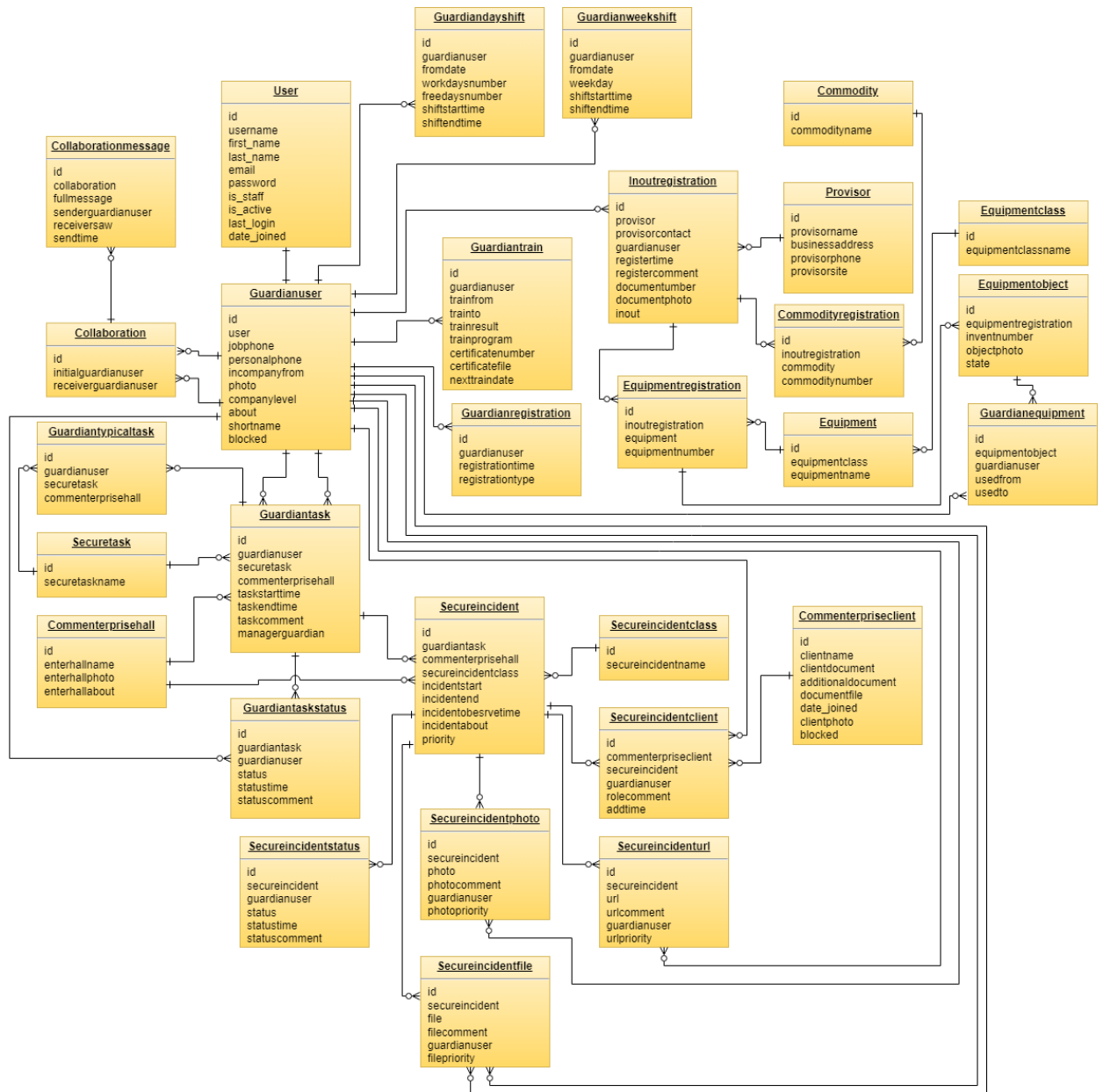


Рисунок 2.3 – Схема БД ПЗ обліку роботи відділу охорони торговельного підприємства

Таблиця деталізації постачання обладнання в торговельне підприємство в БД має назву Equipmentregistration та поля:

– id – ідентифікатор деталізації постачання обладнання в торговельне підприємство;

– inoutregistration – зареєстроване постачання обладнання в торгівельну компанію через відділ охорони, за яким це обладнання потрапило в торгівельне підприємство;

– equipment – обладнання, яке входить до цього постачання;

– equipmentnumber – кількість одиниць обладнання, постачання яких відбулось.

Кожне постачання обладнання може складатися з різного обладнання, а різне обладнання входить в різні постачання, тому зв'язані таблиці Equipment і Inoutregistration зв'язком багато-до-багатьох, який реалізований таблицею Equipmentregistration.

2.4 Висновки за розділом 2

Проектування програми реалізовано повністю, включаючи як визначення повної структури роботи з даними на основі системи PostgreSQL, виділення функціональних вимог до програми та визначення засобів розробки, основою яких є Python-фреймворк Django.

3 РЕАЛІЗАЦІЯ ПРОГРАМИ

3.1 Визначення порядку роботи з основними сутностями

Двома найважливішими сутностями, з якими пов'язана охорона торговельного підприємства і які змінюються протягом свого існування, вимагаючи відповідної підтримки під час функціонування програми, є завдання з охорони (рис. 3.1) та інциденти з безпеки (рис. 3.2).

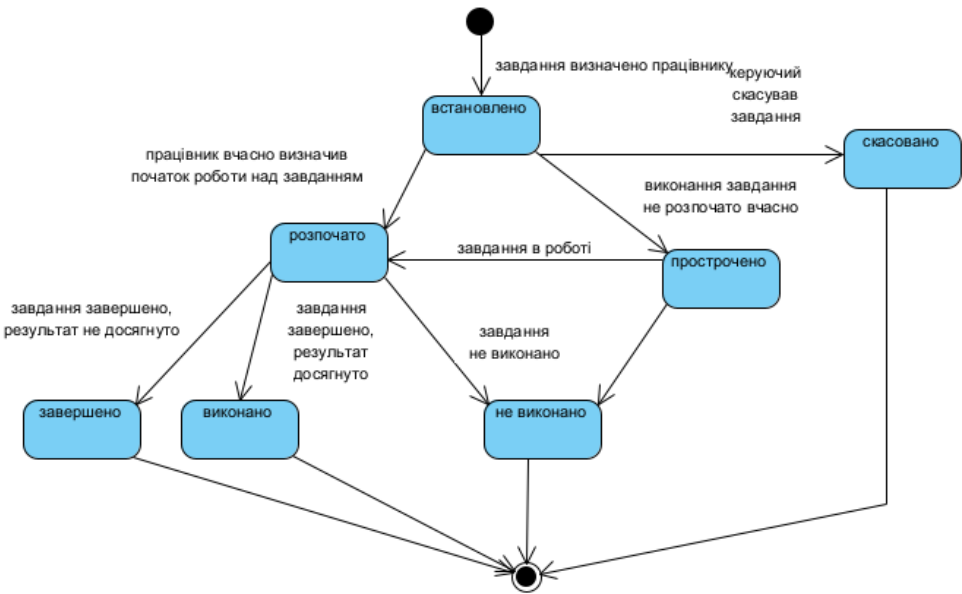


Рисунок 3.1 – Діаграма станів завдання з охорони

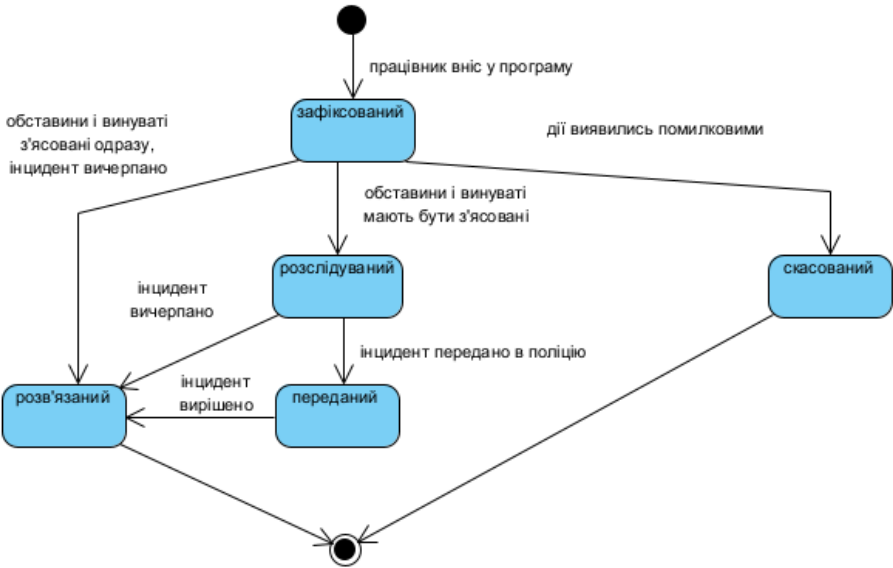


Рисунок 3.2 – Діаграма станів інциденту з безпеки

Для кожної з цих двох сутностей визначено порядок зміни станів, що відповідним чином підтримано в реалізації програми.

3.2 Результати реалізації програмних одиниць застосунку

Тип програмних одиниць, які мали бути забезпечені, визначається вибором Django для реалізації застосунку, тому потрібно було створити ряд представлень, а в роботі цих представлень мали використовуватися створені моделі для роботи з даними та створені шаблони вебсторінок для забезпечення інтерфейса користувача. Шаблони вебсторінок мають оформлення, яке визначене безкоштовним шаблоном [11].

Представлення, які реалізують подібну логіку, реалізовані однією функцією, яка об'єднує в собі відповідні варіанти роботи: наприклад, відображення всіх власних тренувань реалізується для базового користувача, а для користувача з правом керувати це приймає форму відображення всіх тренувань працівника відділу охорони. Реалізацію обох цих функцій забезпечено програмою функцією `guardiantrains_view` наступним чином. По-перше, вона має параметр `guardianuserId`, який може бути не заданий, тоді функція працює для відображення всіх власних тренувань, а якщо заданий, то перевіряється, що користувач має право керувати, а тоді вже за цим користувачем відображаються його тренування. Отримані дані тренувань вносяться в шаблон вебсторінки `guardiantrains.html`. Також перевіряється наявність повідомлень, які надіслані користувачу, який працює з функцією, наявність нових інцидентів з безпеки, які були тільки зафіксовані протягом того ж дня й не переведені до інших станів (якщо користувач з правом керувати) або які були зафіксовані й стосуються тієї зали торгівельного підприємства, в якій виконує завдання з охорони працівник (якщо це базовий користувач), а також прострочені завдання з охорони для базового користувача.

Подібна логіка використана при реалізації інших функцій, тому акцент далі зроблений здебільшого на особливостях, а приведені основні функції.

Відображення власних завдань з охорони та відображення завдань з охорони працівника реалізовано функцією `guardian_tasks_view`, в якій є параметри: `period` – період, за який завдання з охорони мають відображатися (0 – тільки майбутні, 1 – тільки ті, які завершені, 2 – ті, які за часом виконання виконуються на даний момент), `taskstatus` – стан завдання з охорони (значення -1 визначає всі завдання), `taskorder` – порядок сортування завдань (0 – спершу старіші, 1 – спершу новіші), `secure_task` – ідентифікатор загального завдання з охорони, якому мають відповідати відібрані завдання, `guardianuserId` – ідентифікатор працівника відділу охорони.

Відображення завдання з охорони для виконання реалізовано функцією `guardian_task_view`, параметр якої `guardiantaskId` задає номер завдання з охорони. За завданням з охорони витягаються всі властивості, а також додається історія зміни станів завдання з охорони. Усі ці дані надаються тільки в тому випадку, якщо користувач має право керувати, а таке завдання з охорони існує, або якщо користувач є базовим, а завдання направлено цьому працівнику відділу охорони.

Відображення всіх тренувань всіх працівників відділу охорони реалізовано функцією `allguardiantrains_view`, де окрім надання цих даних перевіряється, що користувач має право керувати.

Відображення переліку працівників відділу охорони реалізовано функцією `guardianusers_view` без параметрів. У цій функції також відбувається перевірка прав доступу. Аналогічно це відбувається у інших функціях, де це потрібно.

Відображення даних працівника відділу охорони реалізовано функцією `guardianuser_view`, де параметр `guardianuserId` визначає ідентифікатор працівника відділу охорони, а до даних входять окрім його властивостей також поточні завдання з охорони, графік роботи, останні виявлені інциденти з безпеки.

Відображення інцидентів з безпеки за класом, за залом, за клієнтом, за датою, за працівником реалізовано функцією `secureincidents_view` стосовно тільки тих інцидентів, які виявлені базовим користувачем як працівником відділу охорони, або всіх, якщо це працівник з правом керувати. Клас інциденту визначається параметром `incident_class`, зала – `hall`, ідентифікатор клієнта – `clientId`, дата інциденту – `incidentdate`, працівник відділу – `guardianuserId`. Якщо значення параметра дорівнює -1, то такий параметр не враховується (фільтрування за ним не відбувається). Обмеження стану виконується параметром `status`, напрямок сортування задається параметром `incidentorder`, а параметри `periodstart`, `periodend` задають інтервал часу.

Відображення подій за конкретним об'єктом обладнання реалізовано функцією `equipmentobject_view`, параметром `equipmentobjectId` задається ідентифікатор об'єкта обладнання, а до подій належать видача, повернення працівником, постачання.

Відображення даних про реєстрації товарів реалізовано функцією `commodityregistrations_view`, впорядковуючи реєстрації за датою від найновішої. Відображення даних про реєстрації обладнання реалізовано функцією `equipmentregistrations_view`.

Відображення поточної наявності товарів реалізовано функцією `commodity_availability_view`, яка або виводить дані за кількістю (як різницю між постачанням та вибуттям) всіх товарів, або за товаром, заданим ідентифікатором `commodityid` в параметрі функції.

Відображення поточної наявності обладнання реалізовано функцією `equipment_availability_view`, яка або виводить дані за кількістю та об'єктами обладнання загалом, або за заданим обладнанням, визначеним номером `equipmentId` у параметрі.

Пошук працівника відділу охорони реалізовано функцією `guardianuser_search_view`, в якій параметр `name` визначає повне ім'я працівника або його частину.

Пошук клієнта торговельного підприємства реалізовано функцією `client_search_view`, в якій параметр `name` визначає повне ім'я клієнта або його частину, параметр `document` задає номер документа клієнта.

Визначення власного графіку змін роботи реалізовано функцією `guardianuser_workdays_view`, яка має параметри `workdays_start` та `workdays_end`, які задають проміжок часу роботи, за яким потрібно сформувати графік. Для надання можливості функції визначення графіку змін роботи працівника відділу охорони додано параметр `guardianuserId`, Що задає ідентифікатор самого працівника.

Відображення поточного графіку змін роботи всього відділу охорони реалізовано функцією `guardianshiftsview`, при цьому збираються дані щодо поточного стану змін і за тижневим графіком, і за графіком чергування.

Реєстрація на робочому місці реалізована функцією `registrationcreateview`, яка має параметр `registrationtype`, що задає тип реєстрації, таким чином дозволяючи визначати прибуття на роботу, вибуття, початок перерви, закінчення, а під час перевірки має бути перевірено код Quick Response (QR). QR-код генерується кожен день заново випадково. Цей код через меню профілю може бути виведений на будь-якому пристрої працівника з правом керувати, а також може бути роздрукованим. Під час перевірки встановлюється, чи це коректний QR-код, а також перевіряється місцезнаходження працівника. Якщо воно співпадає з заданим з точністю у 100 метрів, то фіксується реєстрація. Сам QR-код зберігається у файлі на сервері, аналогічно і координати місцезнаходження, які вважаються робочим місцем.

Відображення власних пропусків роботи та відображення пропусків роботи працівником реалізовано функцією `absences_view`, яка має параметр `guardianuserId`, що задає номер працівника. Пропуском вважаються ситуації, коли зміна роботи працівника розпочалась, а він не зареєструвався на роботі. Якщо працівник протягом 30 хвилин зареєструвався, то це не вважається далі пропуском.

Додавання даних клієнта торговельного підприємства реалізовано функцією `commenterpriseclient_add_view`, робота якої забезпечена класом форми `Commenterpriseclientform`.

Визначення виявленого інциденту з безпеки реалізовано функцією `secureincident_add_view`, яка працює з формою `Secureincidentform` та додатковими формами `Secureincidentfileform`, `Secureincidentphotoform`, `Secureincidenturlform` для визначення відповідно файлів, фотографій, посилань на вебресурси, а стан зафіксований створюється програмно під виявлений інцидент з безпеки.

Клас `Guardianuser` реалізує модель працівників відділу охорони торговельного підприємства, включаючи наступні поля.

Поле `user` забезпечує користувацький обліковий запис, що використовується відповідним працівником відділу охорони торговельного підприємства, реалізований через клас `OneToOneField` до моделі `User`, що реалізує описану структуру таблиці БД через засоби Django.

Поле `jobphone` забезпечує робочий номер телефону відповідного працівника відділу охорони торговельного підприємства з реалізацією за класом `CharField`.

Поле `personalphone` забезпечує особистий номер телефону відповідного працівника відділу охорони торговельного підприємства з реалізацією за класом `CharField`.

Поле `incompanyfrom` забезпечує дату, з якої цей користувач є працівником відділу охорони торговельного цього підприємства, з реалізацією за класом `DateField`.

Поле `photo` забезпечує фотографію відповідного працівника відділу охорони торговельного підприємства з реалізацією за класом `ImageField`.

Поле `companylevel` забезпечує рівень доступу під час роботи відповідного працівника відділу охорони торговельного підприємства з реалізацією за класом `BooleanField`.

Поле `about` забезпечує особисту характеристику відповідного працівника відділу охорони торгівельного підприємства з реалізацією за класом `CharField`.

Поле `shortname` забезпечує коротке ім'я відповідного працівника відділу охорони торгівельного підприємства з реалізацією за класом `CharField`.

Поле `blocked` забезпечує блокування роботи цього користувача як працівника відділу охорони торгівельного підприємства або відсутність цього блокування з реалізацією за класом `BooleanField`.

Клас `Guardiantrain` реалізує модель навчання працівників відділу охорони торгівельного підприємства, включаючи наступні поля.

Поле `guardianuser` забезпечує відповідного працівника відділу охорони торгівельного підприємства, який пройшов навчання, реалізованого через клас `ForeignKey` до моделі `Guardianuser`.

Поле `trainfrom` забезпечує дату, коли почалось навчання відповідного працівника відділу охорони торгівельного підприємства, за класом `DateField`.

Поле `trainto` забезпечує дату, коли закінчилось навчання відповідного працівника відділу охорони торгівельного підприємства, за класом `DateField`.

Поле `trainresult` забезпечує отриманий результат навчання за відповідною програмою працівником відділу охорони торгівельного підприємства за класом `CharField`.

Поле `trainprogram` забезпечує програму, за якою відбувалось навчання відповідного працівника відділу охорони торгівельного підприємства, за класом `CharField`.

Поле `certificatenumber` забезпечує номер сертифікату, що відповідним працівником відділу охорони торгівельного підприємства було отримано за результатами навчання, за класом `CharField`.

Поле `certificatefile` забезпечує файл сертифікату, що відповідним працівником відділу охорони торгівельного підприємства було отримано за результатами навчання, за класом `FileField`.

Поле `nexttraindate` забезпечує орієнтовну дату, коли відповідний працівник відділу охорони торгівельного підприємства має пройти наступне навчання після даного для забезпечення необхідної для роботи кваліфікації, за класом `DateField`.

Клас `Guardianweekshift` реалізує модель тижневого варіанта змін, за яким працює працівник відділу охорони торгівельного підприємства, включаючи наступні поля.

Поле `guardianuser` забезпечує працівника відділу охорони торгівельного підприємства, який працює за цим тижневим варіантом змін, реалізованого через клас `ForeignKey` до моделі `Guardianuser`.

Поле `fromdate` забезпечує дату, з якої застосовується цей тижневий варіант змін для відповідного працівника відділу охорони торгівельного підприємства, за класом `DateField`.

Поле `weekday` забезпечує номер дня тижня, в який така зміна реалізується, за класом `PositiveSmallIntegerField`.

Поле `shiftstarttime` забезпечує час початку зміни за цим тижневим варіантом роботи у відповідний день тижня відповідного працівника відділу охорони торгівельного підприємства за класом `TimeField`.

Поле `shiftendtime` забезпечує час закінчення зміни за цим тижневим варіантом роботи у відповідний день тижня відповідного працівника відділу охорони торгівельного підприємства за класом `TimeField`.

Клас `Guardiandayshift` реалізує модель варіантів змін за принципом чергування, які використовуються щодо роботи працівників відділу охорони торгівельного підприємства, включаючи наступні поля.

Поле `guardianuser` забезпечує працівника відділу охорони торгівельного підприємства, який працює за цим варіантом змін за принципом чергування, реалізованого через клас `ForeignKey` до моделі `Guardianuser`.

Поле `fromdate` забезпечує дату, з якої застосовується цей варіант змін за принципом чергування для відповідного працівника відділу охорони торгівельного підприємства, за класом `DateField`.

Поле `workdaysnumber` забезпечує кількість робочих днів, яку поспіль має відпрацювати працівник цього відділу охорони торговельного підприємства перед отримання вихідних, за класом `PositiveSmallIntegerField`.

Поле `freedaysnumber` забезпечує кількість днів відпочинку, яку поспіль має отримати працівник цього відділу охорони торговельного підприємства перед повторенням робочих днів, за класом `PositiveSmallIntegerField`.

Поле `shiftstarttime` забезпечує час початку зміни за цим варіантом роботи за принципом чергування відповідного працівника відділу охорони торговельного підприємства за класом `TimeField`.

Поле `shiftendtime` забезпечує час закінчення зміни за цим варіантом роботи за принципом чергування відповідного працівника відділу охорони торговельного підприємства за класом `TimeField`.

Клас `Guardianregistration` реалізує модель реєстрації працівників відділу охорони торговельного підприємства на робочому місці, включаючи наступні поля.

Поле `guardianuser` забезпечує працівника відділу охорони торговельного підприємства, який зареєструвався на робочому місці, реалізованого через клас `ForeignKey` до моделі `Guardianuser`.

Поле `registrationtime` забезпечує час, коли зафіксовано реєстрацію відповідного працівника відділу охорони торговельного підприємства на робочому місці, за класом `DateTimeField`.

Поле `registrationtype` забезпечує тип реєстрації на робочому місці за класом `PositiveSmallIntegerField`.

Клас `Commententerprisehall` реалізує модель залів торговельного підприємства, включаючи наступні поля.

Поле `enterhallname` забезпечує назву, якою охарактеризовано залу торговельного підприємства, за класом `CharField`.

Поле `enterhallphoto` забезпечує фотографію зали торговельного підприємства за класом `ImageField`.

Поле `enterhallabout` забезпечує опис зали торгівельного підприємства, що характеризує його, за класом `CharField`.

Клас `Securetask` реалізує модель завдань з охорони, які можуть встановлюватися працівникам відділу охорони торгівельного підприємства.

Поле `securetaskname` забезпечує назву (формулювання) завдання з охорони, яке може встановлюватися працівнику відділу охорони торгівельного підприємства на виконання, за класом `CharField`.

Клас `Guardiantypicaltask` реалізує модель типових завдань з охорони, які визначаються зазвичай працівникам відділу охорони торгівельного підприємства, включаючи наступні поля.

Поле `guardianuser` забезпечує працівника відділу охорони торгівельного підприємства, якому типово встановлено таке типове завдання з охорони на робочий день, реалізованого через клас `ForeignKey` до моделі `Guardianuser`.

Поле `securetask` забезпечує завдання з охорони, яке встановлюється відповідному працівнику відділу охорони торгівельного підприємства на виконання типово, реалізоване через клас `ForeignKey` до моделі `Securetask`.

Поле `commenterprishall` забезпечує залу торгівельного підприємства, в якій типово виконує це завдання цей працівник відділу охорони торгівельного підприємства, реалізоване через клас `ForeignKey` до моделі `Commenterprishall`.

Клас `Guardiantask` реалізує модель завдань з охорони, які фактично визначені працівнику відділу охорони торгівельного підприємства на виконання, включаючи наступні поля.

Поле `guardianuser` забезпечує працівника відділу охорони торгівельного підприємства, якому фактично визначено на виконання відповідне завдання, реалізованого через клас `ForeignKey` до моделі `Guardianuser`.

Поле `securetask` забезпечує завдання з охорони, яке встановлене на виконання відповідному працівнику відділу охорони торговельного підприємства, реалізоване через клас `ForeignKey` до моделі `Securetask`.

Поле `commenterprishall` забезпечує залу торговельного підприємства, в якій відповідний працівник відділу охорони торговельного підприємства повинен виконувати відповідне завдання з охорони, реалізовану через клас `ForeignKey` до моделі `Commenterprishall`.

Поле `taskstarttime` забезпечує дату, час, з якого повинен розпочати виконувати це завдання з охорони відповідний працівник відділу охорони торговельного підприємства, за класом `DateTimeField`.

Поле `taskendtime` забезпечує дату, час, до якого повинен закінчити виконувати це завдання з охорони відповідний працівник відділу охорони торговельного підприємства, за класом `DateTimeField`.

Поле `taskcomment` забезпечує коментар до завдання, який встановлений особою, що призначила на виконання це завдання з охорони відповідному працівнику відділу охорони торговельного підприємства, за класом `CharField`.

Поле `manageguardian` забезпечує особу (користувача програми, який є працівником відділу охорони відповідного рівня), що призначила на виконання це завдання з охорони відповідному працівнику відділу охорони торговельного підприємства, реалізовану через клас `ForeignKey` до моделі `Guardianuser`.

Клас `Guardiantaskstatus` реалізує модель станів виконання завдань з охорони працівниками відділу охорони торговельного підприємства, включаючи наступні поля.

Поле `guardiantask` забезпечує завдання з охорони, стосовно якого зафіксовано цей стан виконання, реалізоване через клас `ForeignKey` до моделі `Guardiantask`.

Поле `guardianuser` забезпечує працівника відділу охорони торговельного підприємства, який зафіксував цей стан виконання завдання з охорони, реалізованого через клас `ForeignKey` до моделі `Guardianuser`.

Поле `status` забезпечує стан, який зафіксовано на цей момент при виконанні завдання з охорони працівником відділу охорони торговельного підприємства, за класом `PositiveSmallIntegerField`.

Поле `statustime` забезпечує дату, час, коли цей стан виконання завдання з охорони був зафіксований, за класом `DateTimeField`.

Поле `statuscomment` забезпечує коментар до зафіксованого стану виконання завдання з охорони відповідним працівником відділу охорони торговельного підприємства за класом `CharField`.

Клас `Secureincidentclass` реалізує модель класів інцидентів, які можуть бути виявлені під час проведення охорони торговельного підприємства. Поле `secureincidentname` забезпечує назву класу інциденту, який може бути виявлений під час проведення охорони торговельного підприємства, за класом `CharField`.

Клас `Secureincident` реалізує модель конкретного виявленого під час роботи комерційного підприємства інциденту з безпеки, включаючи наступні поля.

Поле `guardiantask` забезпечує завдання з охорони, під час виконання якого працівник відділу з охорони торговельного підприємства виявив інцидент з безпеки, реалізоване через клас `ForeignKey` до моделі `Guardiantask`.

Поле `commenterprishall` забезпечує залу торговельного підприємства, в якій було виявлено цей інцидент, реалізовану через клас `ForeignKey` до моделі `Commenterprishall`.

Поле `secureincidentclass` забезпечує клас інциденту з безпеки, який було виявлено під час роботи комерційного підприємства, реалізований через клас `ForeignKey` до моделі `Secureincidentclass`.

Поле `incidentstart` забезпечує дату, час, коли, на скільки відомо, розпочався інцидент з безпеки бо коли він стався, за класом `DateTimeField`.

Поле `incidentend` забезпечує дату, час, коли, на скільки відомо, закінчився інцидент з безпеки, за класом `DateTimeField`.

Поле `incidentobesrvetime` забезпечує дату, час, коли інцидент з безпеки було виявлено, за класом `DateTimeField`.

Поле `incidentabout` забезпечує опис ситуації, що характеризує інцидент з безпеки, який стався, за класом `CharField`.

Поле `priority` забезпечує пріоритет, який встановлений цьому інциденту з безпеки, тобто те, на скільки він є критичним, за класом `PositiveSmallIntegerField`.

Клас `Secureincidentfile` реалізує модель файлів, долучених до опису інциденту з безпеки, включаючи наступні поля.

Поле `secureincident` забезпечує інцидент з безпеки, до опису якого долучено цей файл, реалізований через клас `ForeignKey` до моделі `Secureincident`.

Поле `file` забезпечує файл, який долучено до опису інциденту з безпеки, за класом `FileField`.

Поле `filecomment` забезпечує коментар до файлу, який долучено до опису інциденту з безпеки, за класом `CharField`.

Поле `guardianuser` забезпечує працівника відділу охорони комерційного підприємства, який долучив до опису інциденту з безпеки цей файл, реалізованого через клас `ForeignKey` до моделі `Guardianuser`.

Поле `filepriority` забезпечує пріоритет, який встановлений цьому файлу серед всіх долучених до опису інциденту з безпеки матеріалів, включаючи фотографії, посилання та файли, за класом `PositiveSmallIntegerField`.

Клас `Secureincidentphoto` реалізує модель фотографій, долучених до опису інциденту з безпеки, включаючи наступні поля.

Поле `secureincident` забезпечує інцидент з безпеки, до опису якого долучено цю фотографію, реалізований через клас `ForeignKey` до моделі `Secureincident`.

Поле `photo` забезпечує фотографію, яку долучено до опису інциденту з безпеки, за класом `ImageField`.

Поле `photocomment` забезпечує коментар до фотографії, яку долучено до опису інциденту з безпеки, за класом `CharField`.

Поле `guardianuser` забезпечує працівника відділу охорони комерційного підприємства, який долучив до опису інциденту з безпеки цю фотографію, реалізованого через клас `ForeignKey` до моделі `Guardianuser`.

Поле `photopriority` забезпечує пріоритет, який встановлений цій фотографії серед всіх долучених до опису інциденту з безпеки матеріалів, включаючи фотографії, посилання та файли, за класом `PositiveSmallIntegerField`.

Клас `Secureincidenturl` реалізує модель посилань на вебресурси, долучених до опису інциденту з безпеки, включаючи наступні поля.

Поле `secureincident` забезпечує інцидент з безпеки, до опису якого долучено це посилання, реалізований через клас `ForeignKey` до моделі `Secureincident`.

Поле `url` забезпечує посилання на вебресурс, яке долучено до опису цього інциденту з безпеки, за класом `URLField`.

Поле `urlcomment` забезпечує коментар до посилання на вебресурс, який долучено до опису інциденту з безпеки, за класом `CharField`.

Поле `guardianuser` забезпечує працівника відділу охорони комерційного підприємства, який долучив до опису інциденту з безпеки це посилання на вебресурс, реалізованого через клас `ForeignKey` до моделі `Guardianuser`.

Поле `urlpriority` забезпечує пріоритет, який встановлений цьому посиланню на вебресурс серед всіх долучених до опису інциденту з безпеки матеріалів, включаючи фотографії, посилання та файли, за класом `PositiveSmallIntegerField`.

Клас `Secureincidentstatus` реалізує модель станів інцидентів з безпеки, включаючи наступні поля.

Поле `secureincident` забезпечує інцидент з безпеки, стан якого зафіксовано, реалізований через клас `ForeignKey` до моделі `Secureincident`.

Поле `guardianuser` забезпечує працівника відділу охорони торговельного підприємства, який встановив цей стан інциденту з безпеки, реалізованого через клас `ForeignKey` до моделі `Guardianuser`.

Поле `status` забезпечує стан, який у даному випадку зафіксовано для інциденту з безпеки, за класом `PositiveSmallIntegerField`.

Поле `statustime` забезпечує дату, час, коли було зафіксовано цей стан інциденту з безпеки, за класом `DateTimeField`.

Поле `statuscomment` забезпечує коментар до зафіксованого стану інциденту з безпеки відповідним працівником відділу охорони торговельного підприємства за класом `CharField`.

Клас `Commenterpriseclient` реалізує модель клієнта торговельного підприємства, включаючи наступні поля.

Поле `clientname` забезпечує ім'я клієнта (відвідувача) торговельного підприємства за класом `CharField`.

Поле `clientdocument` забезпечує документ, на основі якого ідентифіковано клієнта (відвідувача) торговельного підприємства, за класом `CharField`.

Поле `additionaldocument` забезпечує додатковий документ, на основі якого ідентифіковано клієнта (відвідувача) торговельного підприємства, за класом `CharField`.

Поле `documentfile` забезпечує файл документа, на основі якого ідентифіковано клієнта торговельного підприємства за класом `FileField`.

Поле `date_joined` забезпечує дату, час, коли вперше було внесено дані про відповідного клієнта (відвідувача) торговельного підприємства, за класом `DateTimeField`.

Поле `clientphoto` забезпечує фотографію клієнта (відвідувача) торговельного підприємства за класом `ImageField`.

Поле `blocked` забезпечує позначку про те, що доступ відповідного клієнта (відвідувача) має бути заблокований до торговельного підприємства, за класом `BooleanField`.

Клас `Secureincidentclient` реалізує модель клієнта, який є причетним до інциденту з безпеки, включаючи наступні поля.

Поле `commenterpraseclient` забезпечує клієнта торговельного підприємства, який взяв участь в інциденті з безпеки, реалізованого через клас `ForeignKey` до моделі `Commenterpraseclient`.

Поле `secureincident` забезпечує інцидент з безпеки, в якому взяв участь клієнт торговельного підприємства, реалізований через клас `ForeignKey` до моделі `Secureincident`.

Поле `guardianuser` забезпечує працівника відділу охорони торговельного підприємства, який визначив причетність відповідного клієнта до відповідного інциденту з безпеки, реалізованого через клас `ForeignKey` до моделі `Guardianuser`.

Поле `rolecomment` забезпечує коментар працівника відділу охорони торговельного підприємства, який визначив причетність відповідного клієнта до відповідного інциденту з безпеки, щодо його участі в інцидентів, за класом `CharField`.

Поле `addtime` забезпечує дату, час, коли інформацію щодо причетності клієнта торговельного підприємства до інциденту з безпеки було внесено, за класом `DateTimeField`.

Клас `Collaboration` реалізує модель взаємодії між працівниками відділу охорони торговельного підприємств, включаючи наступні поля.

Поле `initialguardianuser` забезпечує працівника відділу охорони торговельного підприємства, який створив цю взаємодію, реалізованого через клас `ForeignKey` до моделі `Guardianuser`.

Поле `receiverguardianuser` забезпечує працівника відділу охорони торговельного підприємства, який бере з іншого боку у взаємодії, реалізованого через клас `ForeignKey` до моделі `Guardianuser`.

Клас `Collaborationmessage` реалізує модель повідомлень між працівниками відділу охорони торговельного підприємства, включаючи наступні поля.

Поле `collaboration` забезпечує взаємодію між працівниками відділу охорони торговельного підприємства, до якої належить повідомлення, реалізовану через клас `ForeignKey` до моделі `Collaboration`.

Поле `fullmessage` забезпечує повний текст повідомлення між працівниками відділу охорони торговельного підприємства за класом `CharField`.

Поле `senderguardianuser` забезпечує працівника відділу охорони торговельного підприємства, який надіслав це повідомлення, реалізованого через клас `ForeignKey` до моделі `Guardianuser`.

Поле `receiversaw` забезпечує позначку про те, що працівник відділу охорони торговельного підприємства, якому направлено повідомлення, побачив його, за класом `BooleanField`.

Поле `sendtime` забезпечує дату, час, коли це повідомлення між працівниками відділу охорони торговельного підприємства було створено, за класом `DateTimeField`.

Клас `Provisor` реалізує модель постачальника, включаючи наступні поля.

Поле `provisorname` забезпечує назву компанії-постачальника за класом `CharField`.

Поле `businessaddress` забезпечує ділову адресу компанії-постачальника за класом `CharField`.

Поле `provisorphone` забезпечує номер телефону компанії-постачальника за класом `CharField`.

Поле `provisorsite` забезпечує вебсайт компанії-постачальника за класом `URLField`.

Клас `Commodity` реалізує модель товарів, які постачаються торговельному підприємству. Поле `commodityname` забезпечує назву товару, який постачається торговельному підприємству, за класом `CharField`.

Клас `Inoutregistration` реалізує модель зареєстрованого постачання або вибуття товарів або обладнання, включаючи наступні поля.

Поле `provisor` забезпечує постачальника товарів або обладнання, представник якого привіз товари, реалізованого через клас `ForeignKey` до моделі `Provisor`.

Поле `provisorcontact` забезпечує контактну особу, яка привезла від постачальника товари або обладнання, прийом яких або відправлення яких відбулося, за класом `CharField`.

Поле `guardianuser` забезпечує працівника відділу охорони торговельного підприємства, який виконав реєстрацію постачання або вибуття товарів або обладнання, реалізованого через клас `ForeignKey` до моделі `Guardianuser`.

Поле `registertime` забезпечує дату, час реєстрації постачання або вибуття товарів або обладнання в торговельну компанію через відділ охорони за класом `DateTimeField`.

Поле `registercomment` забезпечує коментар працівника відділу охорони торговельного підприємства, який виконав реєстрацію постачання або вибуття товарів або обладнання, за класом `CharField`.

Поле `documentumber` забезпечує номер документа, на основі якого було оформлено постачання або відправлення товару або обладнання, за класом `CharField`.

Поле `documentphoto` забезпечує фотографію документа, на основі якого було оформлено постачання або відправлення товару або обладнання, за класом `ImageField`.

Поле `inout` забезпечує позначку про те, чи це було постачання товару або обладнання, чи вибуття, за класом `BooleanField`.

Клас `Inoutregistration` реалізує модель деталей реєстрації постачання або вибуття товарів, включаючи наступні поля.

Поле `inoutregistration` забезпечує зареєстроване постачання або вибуття товарів в торговельну компанію через відділ охорони, яке деталізовано цим записом, реалізоване через клас `ForeignKey` до моделі `Inoutregistration`.

Поле `commodity` забезпечує товар, який входить в це постачання або вибуття товарів, реалізований через клас `ForeignKey` до моделі `Commodity`.

Поле `commoditynumber` забезпечує кількість одиниць цього товару в постачанні або вибутті за класом `FloatField`.

Клас `Equipmentclass` реалізує модель класів обладнання, яке використовується в роботі торгівельного підприємства. Поле `equipmentclassname` забезпечує назву класу обладнання за класом `CharField`.

Клас `Equipment` реалізує модель обладнання, яке наявне в торгівельному підприємстві та використовується в роботі, включаючи наступні поля.

Поле `equipmentclass` забезпечує клас цього обладнання, яке наявне в торгівельному підприємстві та використовується в роботі, реалізований через клас `ForeignKey` до моделі `Equipmentclass`.

Поле `equipmentname` забезпечує назву цього обладнання, яке наявне в торгівельному підприємстві та використовується в роботі, за класом `CharField`.

Клас `Equipmentobject` реалізує модель конкретних об'єктів обладнання, які використовуються в роботі торгівельного підприємства, включаючи наступні поля.

Поле `equipmentregistration` забезпечує деталізацію постачання обладнання в торгівельну компанію через відділ охорони, за якою цей об'єкт обладнання потрапив у торгівельну компанію, реалізований через клас `ForeignKey` до моделі `Equipmentregistration`.

Поле `inventnumber` забезпечує інвентарний номер об'єкта обладнання за класом `CharField`.

Поле `objectphoto` забезпечує фотографію об'єкта обладнання за класом `ImageField`.

Поле `state` забезпечує стан об'єкта обладнання за класом `PositiveSmallIntegerField`.

Клас `Guardianequipment` реалізує модель видачі та використання об'єктів обладнання працівниками відділу охорони торговельного підприємства, включаючи наступні поля.

Поле `equipmentobject` забезпечує об'єкт обладнання, який виданий та використовується працівником відділу охорони торговельного підприємства, реалізований через клас `ForeignKey` до моделі `Equipmentobject`.

Поле `guardianuser` забезпечує працівника відділу охорони торговельного підприємства, який використовує цей об'єкт обладнання, реалізованого через клас `ForeignKey` до моделі `Guardianuser`.

Поле `usedfrom` забезпечує дату, час, коли об'єкт обладнання був виданий працівнику відділу охорони торговельного підприємства, за класом `DateTimeField`.

Поле `usedto` забезпечує дату, час, коли об'єкт обладнання був повернутий працівником відділу охорони торговельного підприємства, за класом `DateTimeField`.

Клас `Equipmentregistration` реалізує модель деталізації постачання обладнання в торговельне підприємство, включаючи наступні поля.

Поле `inoutregistration` забезпечує зареєстроване постачання обладнання в торговельну компанію через відділ охорони, за яким це обладнання потрапило в торговельне підприємство, реалізоване через клас `ForeignKey` до моделі `Inoutregistration`.

Поле `equipment` забезпечує обладнання, яке входить до цього постачання, реалізоване через клас `ForeignKey` до моделі `Equipment`.

Поле `equipmentnumber` забезпечує кількість одиниць обладнання, постачання яких відбулось, за класом `PositiveIntegerField`.

3.3 Висновки за розділом 3

У даному розділі розглянуто особливості впорядкування роботи над змінами станів об'єктів предметної області, що необхідно було для

забезпечення реалізації роботи над програмою обліку роботи відділу охорони торговельного підприємства, описано отримані результати під час реалізації програми.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Призначення програми

Програма призначена для забезпечення обліку роботи відділу охорони торгівельного підприємства, включаючи реєстрацію перебування працівників на робочому місці, графіків роботи, визначення завдань з охорони, інцидентів з безпеки, які сталися, отриманих поставань та вибуття товарів, реєстрації видачі та повернення обладнання.

4.2 Умови виконання програми

Умовами експлуатації є забезпечення програмних інструментів, включаючи Python, Django, PostgreSQL, та надання відповідних спеціалістів для підтримки роботи такої програмної системи в підсумку. Зокрема таким спеціалістом має бути адміністратор, який зможе розгорнути вебзастосунок на сервері, керувати його роботою.

Технічні засоби пропонуються мінімально забезпечувати на рівні, який включає 150 Гб доступного дискового простору на жорсткому диску для базового виконання програми, 8 Гб оперативної пам'яті, наявність 2-ядерного процесора, що має дозволити підтримку 20 запитів на секунду до сервера.

4.3 Виконання і тестування програми

У процесі тестування було перевірено та зафіксовано успішність виконання всіх функцій програми як у випадку локального розміщення вебзастосунку, так і у випадку забезпечення доступу до сервера назовні.

Логіка роботи з програмою розпочинається зі звернення до вебзастосунку за основною адресою та відображення при цьому форми авторизації. Без авторизації робота не відбувається. Якщо користувач вірно

задав логін та пароль, то відображається сторінка з завданнями з охорони працівника відділу охорони торгівельного підприємства.

Спершу потрібно розглянути можливості роботи базового користувача. Ці можливості виділені в меню головним чином. Вони розділені на верхнє меню, де відображається функція доступу до повідомлень у межах спілкування з іншими працівниками. Ця сторінка відображає всі вже створені взаємодії, після чого можна обрати потрібну і подивитись, відправити повідомлення. Звідси можна пошуком за іменем створити взаємодію з іншим працівником, який ще не доданий до спілкування. У верхній панелі біля назва повідомлення додається плюс та число, що відповідають кількості нових повідомлень, які ще не прочитав користувач, який авторизувався.

Верхнє меню також включає перехід до власного профілю та можливість вийти з програми.

Базовому користувачу доступне меню зліва, яке включає наступні розділи.

Розділ графіку роботи містить пункти: перегляду власного графіка роботи за датами, перегляду графіка роботи на поточну мить за всім відділом охорони, перегляду пропусків роботи, які допустив цей працівник.

Розділ завдань включає відображення власних завдань з охорони працівника, які йому призначені на виконання.

Розділ інцидентів включає перехід до виявлених інцидентів цим працівником відділу охорони (рис. 4.1), перехід до створення запису про новий зафіксований інцидент з безпеки.

Сторінка інцидентів з безпеки, які зафіксував працівник, що авторизувався, містить можливості керування інцидентами в межах можливості такого працівника. Потрібно обрати тип вибору, що включає або клас інциденту, або залу, або клієнта (для користувача з правом керування також працівника). Після цього обрати відповідне значення з переліку, а також встановити потрібний стан, варіант сортування та проміжок дат.

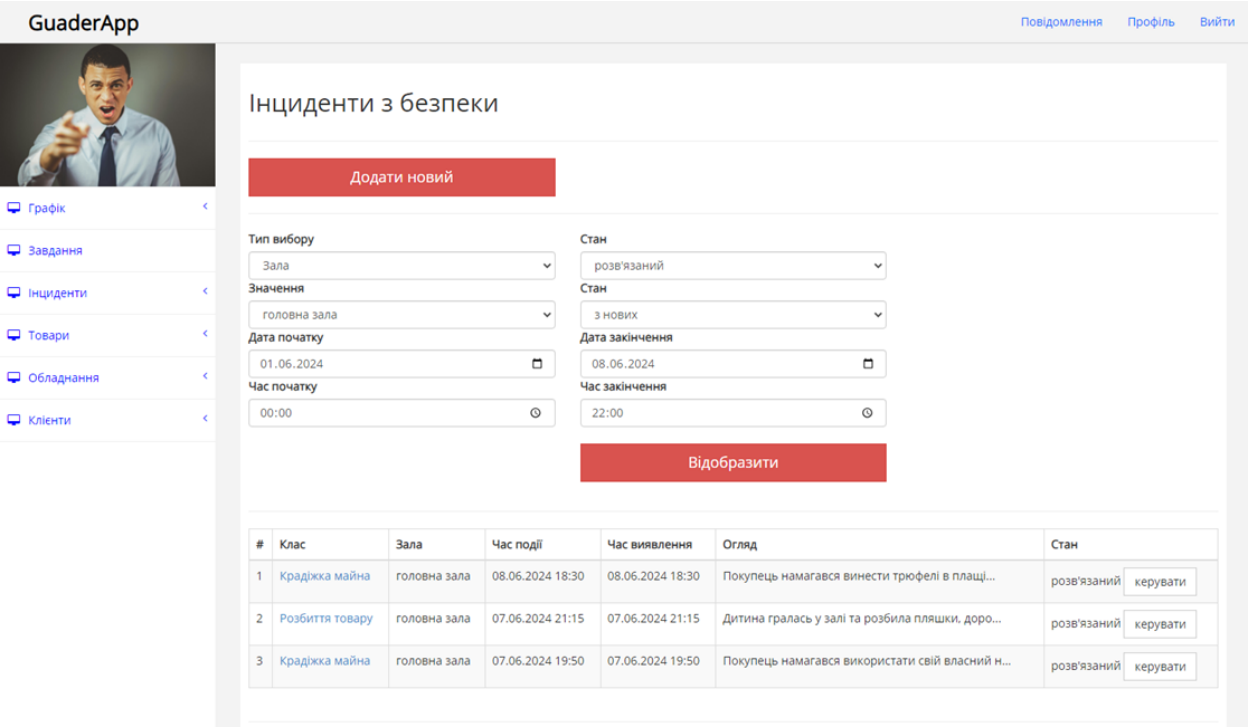


Рисунок 4.1 – Перегляд інцидентів з безпеки у програмі

Розділ товарів включає можливості реєстрації вибуття, прибуття товарів, реєстрацій, перегляду наявності товарів.

Розділ обладнання містить функції реєстрації прибуття, вибуття обладнання, визначення наявності обладнання, реєстрації.

Розділ клієнтів містить можливості пошуку клієнтів та додавання нових.

У випадку роботи користувача з правом керування з програмою меню також містить розділ каталогу, в якому можна переглядати, додавати зали, постачальників.

Окрім того користувач з правом керування містить деякі додаткові пункти в тих самих розділах меню, включаючи перегляд розподілу завдань та призначення завдань з охорони працівника на виконання, що робить цей пункт меню вже розділом повноцінно. Також це можливість видавати та повертати обладнання іншим працівникам з розділу меню обладнання. Окрім того це можливості додавати даних товарів.

Такий користувач також замість меню графіка має меню працівників, звідки він може реєструвати в програмі нових працівників, переглядати всіх працівників відділу охорони (рис. 4.2), переглядати тренування працівників, переглядати наявність працівників на робочому місці у відповідності зі змінами, виконувати пошук працівників. Також сюди входять засоби роботи з графіком роботи.

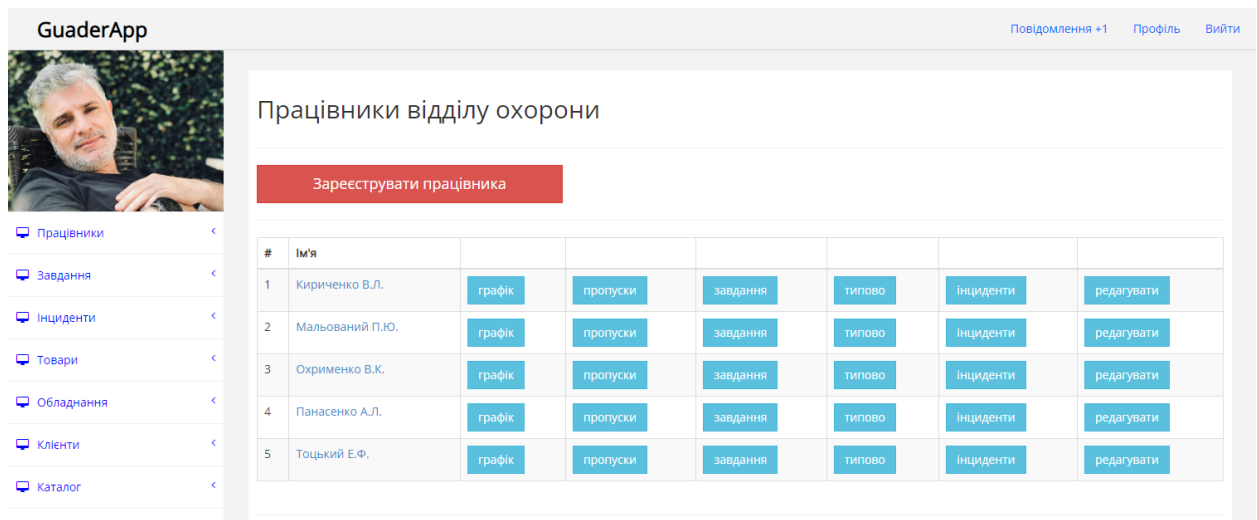


Рисунок 4.2 – Перегляд працівників відділу охорони

Під час перегляду працівників відділу охорони можна перейти до реєстрації нового працівника, а можна за кожним наявним перейти до конкретних даного про нього, включаючи профіль (рис. 4.3), натискаючи на посилання з іменем, перейти до перегляду пропусків роботи працівником, призначених йому завдань, визначених йому типових завдань, перегляду інцидентів, які він зафіксував, редагування даних користувача.

Зі сторінки профілю працівника відділу охорони доступні його основні дані, а також дані про реалізовані ним тренування (якщо тренувань більше 5, то відображається кнопка для переходу на сторінку всіх тренувань працівника, – та сама логіка використовується і для інших складових), встановлені йому завдання, його пропуски роботи, графік роботи, виявлені ним інциденти.

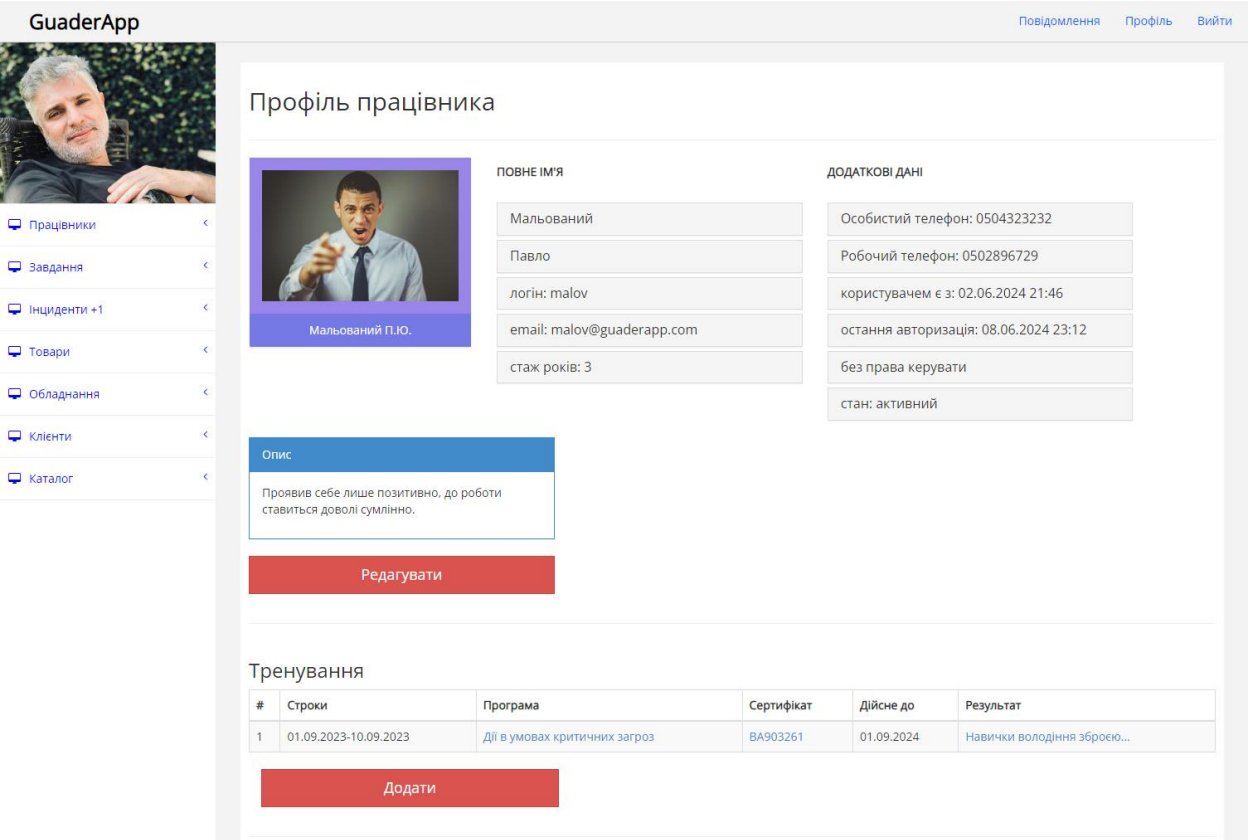


Рисунок 4.3 – Сторінка профілю працівника відділу охорони

Сторінка розподілу завдань з охорони (рис. 4.4) дозволяє задати мить, на яку такий розподіл потрібно відобразити, а за кожним завданням з охорони можна перейти до перегляду сторінки завдання.

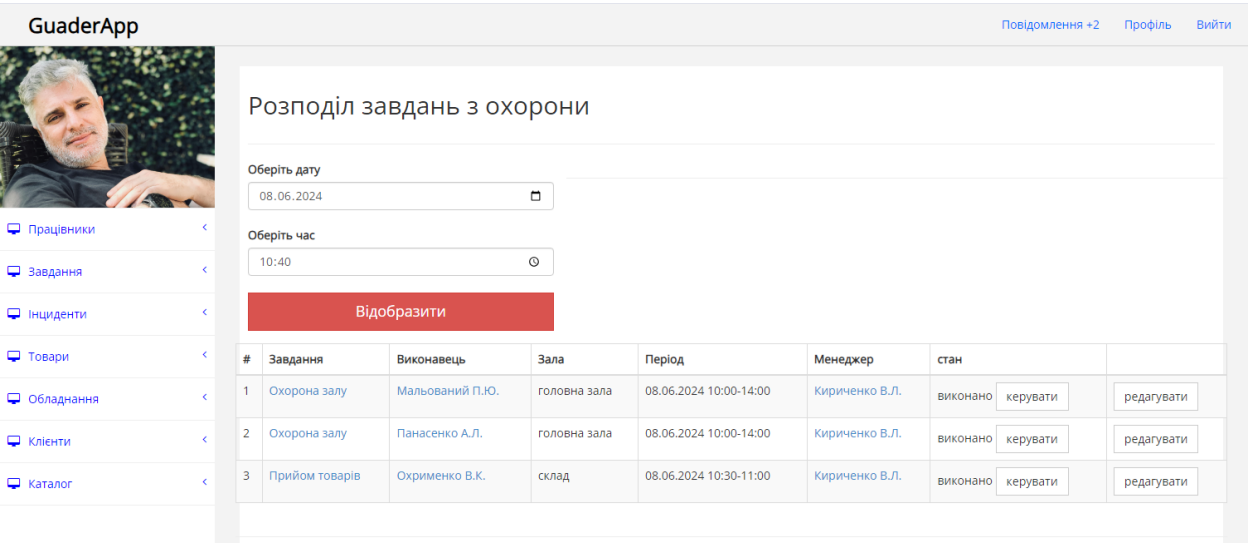


Рисунок 4.4 – Сторінка розподілу завдань з охорони

4.4 Висновки за розділом 4

Проведене тестування виявило успішну реалізацію функцій програми. Описано основні засоби, доступні для роботи з програмою працівникам відділу охорони торгівельного підприємства.

ВИСНОВКИ

Виконана робота сумарно за результатами виконання всіх завдань дозволила досягти мету роботи, якою була розробка ПЗ обліку роботи відділу охорони торгівельного підприємства для забезпечення обґрунтованих рішень щодо керування роботою відділу.

У роботі забезпечено всі етапи розробки програми обліку роботи відділу охорони торгівельного підприємства, включаючи вибір технологій розробки з обґрунтуванням рішень, що підсумково включили мову Python, базу даних на основі PostgreSQL, фреймворк Django, включаючи проєктування самої програми, програмну реалізацію функціональних можливостей. Це дозволило створити програму, здатну до накопичення даних стосовно різних сторін обліку роботи, включаючи облік робочого часу, виконання завдань, виявлених інцидентів з можливістю подальшого опрацювання накопичених даних. Наявність таких функціональних можливостей порівняно з програмами-аналогами, які потребують сплати за користування, робить її достатньо привабливою для практичного застосування, а використані технології здатні підтримувати таке практичне використання.

Розроблена програма призначена для забезпечення обліку роботи відділу охорони торгівельного підприємства, включаючи реєстрацію перебування працівників на робочому місці, графіків роботи, визначення завдань з охорони, інцидентів з безпеки, які сталися, отриманих поставок та вибуття товарів, реєстрації видачі та повернення обладнання.

Усі завдання роботи виконано, а виконане тестування підтвердило забезпечення виконання реалізації всіх запланованих функцій з отриманням тих результатів, які визначали успішне виконання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Програми для охоронного підприємства на базі OneBox OS [Електрон. ресурс]. – Режим доступу : <https://1b.app/ua/software-for/security-and-safety/>.
2. Security Guard Reporting Software. OfficerReports.com [Electronic resource]. – Access mode : <https://officerreports.com/products.html>.
3. Guard Tour System. Security Guard Management System. GuardsPro [Electronic resource]. – Access mode : <https://www.guardspro.com/guard-tour-system?tab=overview>.
4. Security Guard Management System. GuardsPro [Electronic resource]. – Access mode : <https://www.guardspro.com/>.
5. Guard Tour System. Security Guard Management System. GuardsPro [Electronic resource]. – Access mode : <https://www.guardspro.com/guard-tour-system?tab=reporting>.
6. Django Tutorial [Electronic resource]. – Access mode : <https://www.w3schools.com/django/>.
7. Getting started with Django. Django [Electronic resource]. – Access mode : <https://www.djangoproject.com/start/>.
8. Tornado Web Server – Tornado 6.4.1 documentation [Electronic resource]. – Access mode : <https://www.tornadoweb.org/en/stable/>.
9. Introduction to Tornado Framework – GeeksforGeeks [Electronic resource]. – Access mode : <https://www.geeksforgeeks.org/introduction-to-tornado-framework/>.
10. PostgreSQL Tutorial documentation [Electronic resource]. – Access mode : <https://www.postgresqltutorial.com/>.
11. Two Page Free Website Template. Free CSS Templates. Free CSS [Electronic resource]. – Access mode : <https://www.free-css.com/free-css-templates/page205/two-page>.

ДОДАТОК А
Технічне завдання

Вступ

Результати роботи відділу охорони торгівельного підприємства є важливими для забезпечення фінансових результатів роботи всього підприємства. Оскільки таке підприємство забезпечує продаж товарів, то зловмисники можуть прагнути заволодіти ними безкоштовно, таким чином наносячи збитки підприємству. Будь-які такі випадки мають виявлятися, обліковуватися. Однак виявлення не є простим процесом. Необхідно розподіляти завдання між працівниками, контролювати їх наявність, тобто виконувати цілий ряд завдань, які повинні мати вираження у вигляді функціональних можливостей програми, оскільки без програмного забезпечення неможливо керувати такими процесами у великому, середньому підприємстві або навіть малому підприємстві, але на тривалому інтервалі часу. Тому необхідно проаналізувати аналоги програми, визначити ті функції, які зможуть забезпечити підтримку діяльності відділу охорони торгівельного підприємства для забезпечення ефективної роботи.

А.1 Підстави для розробки

Дипломна кваліфікаційна робота виконувалась у відповідності та на підставі Наказу № 168 від 24 квітня 2024 р. за Національним університетом «Запорізька політехніка», яким визначено тему роботи – «Програмне забезпечення обліку роботи відділу охорони торгівельного підприємства».

А.2 Призначення розробки

Програма призначена для забезпечення обліку роботи відділу охорони торгівельного підприємства, включаючи реєстрацію перебування працівників на робочому місці, графіків роботи, визначення завдань з охорони, інцидентів з безпеки, які сталися, отриманих поставчань та вибуття товарів, реєстрації

видачі та повернення обладнання.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Функціональні характеристики ПЗ обліку роботи відділу охорони торгівельного підприємства мають включати:

- відображення переліку працівників відділу охорони;
- додавання даних тренування працівника відділу охорони;
- редагування даних тренування працівника відділу охорони;
- відображення всіх тренувань працівника відділу охорони;
- відображення всіх тренувань всіх працівників відділу охорони;
- визначення графіку змін роботи працівника відділу охорони;
- відображення поточного графіку змін роботи всього відділу охорони;
- реєстрація працівника відділу охорони на робочому місці;
- відображення пропусків роботи працівником;
- відображення поточної наявності працівників на роботі загалом та за залами торгівельного підприємства;
- керування залами торгівельного підприємства;
- відображення розподілу завдань з охорони на заданий момент часу;
- призначення завдання з охорони для виконання працівнику;
- відображення завдань з охорони працівника;
- керування станом виконання завдання з охорони;
- визначення переліку типових завдань працівника;
- відображення даних профілю працівника відділу охорони;
- авторизацію та закриту реєстрацію працівників відділу охорони;
- редагування даних профілю працівника відділу охорони;
- відображення завдання з охорони для виконання;
- визначення виявленого інциденту з безпеки;
- відображення інцидентів з безпеки за класом, за залом, за

працівником, за клієнтом, за датою, за станом;

- відображення даних клієнта торговельного підприємства;
- пошук клієнта торговельного підприємства;
- пошук працівника відділу охорони;
- керування станом інциденту з безпеки;
- додавання даних клієнта торговельного підприємства;
- редагування даних клієнта торговельного підприємства;
- відправлення повідомлення між працівниками відділу охорони;
- відображення переліків взаємодії між працівниками відділу охорони;
- відображення повідомлень між заданими працівниками відділу

охорони;

- керування переліком постачальників;
- керування переліком товарів;
- реєстрація прибуття або вибуття товарів;
- реєстрація прибуття або вибуття обладнання;
- відображення даних про реєстрації товарів;
- відображення даних про реєстрації обладнання;
- відображення поточної наявності товарів;
- відображення поточної наявності обладнання;
- реєстрація видачі обладнання працівнику відділу охорони;
- відображення подій за конкретним об'єктом обладнання;
- реєстрація повернення обладнання працівником.

A.3.2 Умови експлуатації

Умовами експлуатації є забезпечення програмних інструментів, що мають відповідати тим технологіям, які будуть обрані в роботі, та надання відповідних спеціалістів для підтримки роботи такої програмної системи в підсумку. Зокрема таким спеціалістом має бути адміністратор, який зможе розгорнути вебзастосунок на сервері, керувати його роботою.

А.3.3 Вимоги до складу та параметрів технічних засобів

Технічні засоби пропонуються мінімально забезпечувати на рівні, який включає 150 Гб доступного дискового простору на жорсткому диску для базового виконання програми, 8 Гб оперативної пам'яті, наявність 2-ядерного процесора, що має дозволити підтримку 20 запитів на секунду до сервера.

А.4 Порядок контролю та приймання

Контроль і приймання роботи регламентуються забезпеченням роботи за завданням на дипломну роботу, в календарному плані якого вказано відповідні часові проміжки, та процедурою проведення захисту дипломної роботи.

ДОДАТОК Б
Текст програми

```

import datetime
from .models import Guardianuser
from django.shortcuts import render
from .models import Guardiantrain
from .models import Guardiantask
from .models import Securetask
from .models import Guardiantaskstatus
from .models import Collaborationmessage
from django.db.models import Q
from .models import Secureincident
from .models import Secureincidentstatus

def guardianusers_view(request):
    guardianusers_datacontext = {}
    if request.user.is_active:
        guardian_user = Guardianuser.objects.filter(user =
request.user)
        if guardian_user and not guardian_user.blocked and
guardian_user.companylevel:
            guardian_users = Guardianuser.objects.all()
            guardian_users =
guardian_users.order_by("shortname")
            guardianusers_datacontext["guardians"] =
guardian_users
            collaborationmessages =
Collaborationmessage.objects.filter(Q(collaboration__initialguardianus
er = guardian_user) | Q(collaboration__receiverguardianuser =
guardian_user))
            collaborationmessages =
collaborationmessages.filter(receiversaw = False)
            collaborationmessages =
collaborationmessages.exclude(senderguardianuser = guardian_user)
            collaborationmessages =
collaborationmessages.count()
            guardianusers_datacontext["newmessages"] =
collaborationmessages
            current_time = datetime.datetime.now()
            current_time = datetime.datetime(current_time.year,
current_time.month, current_time.day, 0,0)
            current_time2 = datetime.datetime(current_time.year,
current_time.month, current_time.day, 23, 59)

```

```

        secureincidents =
Secureincidentstatus.objects.select_related("secureincident")
        secureincidents = secureincidents.filter(status = 0)
        secureincidents =
secureincidents.filter(statustime__gte = current_time, statustime__lte
= current_time2)
        secureincidents =
secureincidents.values("secureincident")
        secureincidents = secureincidents.distinct()
        secureincidents = len(secureincidents)
        guardianusers_datacontext["newincidents"] =
secureincidents
        return render(request, "guaderapp/guardians.html",
guardianusers_datacontext)
        elif guardian_user and not guardian_user.blocked and not
guardian_user.companylevel:
            guardian_users = Guardianuser.objects.all(blocked =
False)
            guardian_users =
guardian_users.order_by("shortname")
            guardianusers_datacontext["guardians"] =
guardian_users
            collaborationmessages =
Collaborationmessage.objects.filter(Q(collaboration__initialguardianus
er = guardian_user) | Q(collaboration__receiverguardianuser =
guardian_user))
            collaborationmessages =
collaborationmessages.filter(receiversaw = False)
            collaborationmessages =
collaborationmessages.exclude(senderguardianuser = guardian_user)
            collaborationmessages =
collaborationmessages.count()
            guardianusers_datacontext["newmessages"] =
collaborationmessages
            old_guardian_tasks =
Guardiantaskstatus.objects.filter(status = 2)
            old_guardian_tasks =
old_guardian_tasks.filter(guardiantask__guardianuser = guardian_user)
            newer_guardian_tasks =
Guardiantaskstatus.objects.filter(guardiantask__guardianuser =
guardian_user)
            newer_guardian_tasks =
newer_guardian_tasks.exclude(status = 1)

```

```

        newer_guardian_tasks =
newer_guardian_tasks.values("guardiantask__id")
        old_guardian_tasks =
old_guardian_tasks.exclude(guardiantask__id__in =
newer_guardian_tasks)
        old_guardian_tasks =
old_guardian_tasks.values("guardiantask")
        old_guardian_tasks_part1 = len(old_guardian_tasks)
        current_time = datetime.datetime.now()
        old_guardian_tasks =
Guardiantaskstatus.objects.filter(status = 0,
guardiantask__taskstarttime__lt = current_time)
        newer_guardian_tasks =
Guardiantaskstatus.objects.filter(guardiantask__taskstarttime__lt =
current_time)
        newer_guardian_tasks =
newer_guardian_tasks.exclude(status = 0)
        newer_guardian_tasks =
newer_guardian_tasks.values("guardiantask__id")
        old_guardian_tasks =
old_guardian_tasks.exclude(guardiantask__id__in =
newer_guardian_tasks)
        old_guardian_tasks =
old_guardian_tasks.filter(guardianuser = guardian_user)
        old_guardian_tasks =
old_guardian_tasks.values("guardiantask")
        old_guardian_tasks_part2 = len(old_guardian_tasks)
        guardianusers_datacontext["taskstodo"] =
old_guardian_tasks_part1 + old_guardian_tasks_part2
        current_task =
Guardiantask.objects.select_related("commenterprisehall")
        current_time = datetime.datetime.now()
        current_task = current_task.filter(guardianuser =
guardian_user)
        current_task =
current_task.filter(taskendtime__isnull = True)
        current_task =
current_task.filter(taskstarttime__lte = current_time)
        commenterprise_hall =
current_task.commenterprisehall
        current_time = datetime.datetime(current_time.year,
current_time.month, current_time.day, 0,0)

```

```

        current_time2 = datetime.datetime(current_time.year,
current_time.month, current_time.day, 23, 59)
        secureincidents =
Secureincidentstatus.objects.select_related("secureincident")
        secureincidents = secureincidents.filter(status = 0)
        secureincidents =
secureincidents.filter(statustime__gte = current_time)
        secureincidents =
secureincidents.filter(statustime__lte = current_time2)
        secureincidents =
secureincidents.filter(commententerprisehall = commententerprise_hall)
        secureincidents =
secureincidents.values("secureincident")
        secureincidents = secureincidents.distinct()
        secureincidents = len(secureincidents)
        guardianusers_datacontext["newincidents"] =
secureincidents

        return render(request, "guaderapp/guardians.html",
guardianusers_datacontext)
        elif guardian_user and guardian_user.blocked:
            return render(request, "guaderapp/error.html",
{"error": "Ваш доступ заблоковано. Зверніться до керівництва"})
            return render(request, "guaderapp/error.html", {"error":
"Для доступу потрібна авторизація"})

def guardiantrains_view(request, guardianuserId = None):
    guardiantrain_datacontext = {}
    if request.user.is_active:
        guardian_user = Guardianuser.objects.filter(user =
request.user)
        if guardian_user and not guardian_user.blocked and
guardian_user.companylevel and guardianuserId:
            try:
                guardian_user_search =
Guardianuser.objects.get(id = guardianuserId)
            except Guardianuser.DoesNotExist:
                return render(request, "guaderapp/error.html",
{"error": "Сталася помилка. Такого працівника не існує"})
            guardian_user_trains =
Guardiantrain.objects.filter(guardianuser = guardian_user_search)
            guardian_user_trains =
guardian_user_trains.order_by("-trainfrom")

```

```

        guardiantrain_datacontext["trains"] =
guardian_user_trains
        guardiantrain_datacontext["guardian"] =
guardian_user
        collaborationmessages =
Collaborationmessage.objects.filter(Q(collaboration__initialguardianus
er = guardian_user) | Q(collaboration__receiverguardianuser =
guardian_user))
        collaborationmessages =
collaborationmessages.filter(receiversaw = False)
        collaborationmessages =
collaborationmessages.exclude(senderguardianuser = guardian_user)
        collaborationmessages =
collaborationmessages.count()
        guardiantrain_datacontext["newmessages"] =
collaborationmessages
        current_time = datetime.datetime.now()
        current_time = datetime.datetime(current_time.year,
current_time.month, current_time.day, 0,0)
        current_time2 = datetime.datetime(current_time.year,
current_time.month, current_time.day, 23, 59)
        secureincidents =
Secureincidentstatus.objects.select_related("secureincident")
        secureincidents = secureincidents.filter(status = 0)
        secureincidents =
secureincidents.filter(statustime__gte = current_time, statustime__lte
= current_time2)
        secureincidents =
secureincidents.values("secureincident")
        secureincidents = secureincidents.distinct()
        secureincidents = len(secureincidents)
        guardiantrain_datacontext["newincidents"] =
secureincidents
        return render(request,
"guaderapp/guardiantrains.html", guardiantrain_datacontext)
        elif guardian_user and not guardian_user.blocked and not
guardian_user.companylevel:
            guardian_user_trains =
Guardiantrain.objects.filter(guardianuser = guardian_user)
            guardian_user_trains =
guardian_user_trains.order_by("-trainfrom")
            guardiantrain_datacontext["trains"] = guardian_users

```

```

guardiantrain_datacontext["guardian"] =
guardian_user
    collaborationmessages =
Collaborationmessage.objects.filter(Q(collaboration__initialguardianus
er = guardian_user) | Q(collaboration__receiverguardianuser =
guardian_user))
    collaborationmessages =
collaborationmessages.filter(receiversaw = False)
    collaborationmessages =
collaborationmessages.exclude(senderguardianuser = guardian_user)
    collaborationmessages =
collaborationmessages.count()
    guardiantrain_datacontext["newmessages"] =
collaborationmessages
    old_guardian_tasks =
Guardiantaskstatus.objects.filter(status = 2)
    old_guardian_tasks =
old_guardian_tasks.filter(guardiantask__guardianuser = guardian_user)
    newer_guardian_tasks =
Guardiantaskstatus.objects.filter(guardiantask__guardianuser =
guardian_user)
    newer_guardian_tasks =
newer_guardian_tasks.exclude(status = 1)
    newer_guardian_tasks =
newer_guardian_tasks.values("guardiantask__id")
    old_guardian_tasks =
old_guardian_tasks.exclude(guardiantask__id__in =
newer_guardian_tasks)
    old_guardian_tasks =
old_guardian_tasks.values("guardiantask")
    old_guardian_tasks_part1 = len(old_guardian_tasks)
    current_time = datetime.datetime.now()
    old_guardian_tasks =
Guardiantaskstatus.objects.filter(status = 0,
guardiantask__taskstarttime__lt = current_time)
    newer_guardian_tasks =
Guardiantaskstatus.objects.filter(guardiantask__taskstarttime__lt =
current_time)
    newer_guardian_tasks =
newer_guardian_tasks.exclude(status = 0)
    newer_guardian_tasks =
newer_guardian_tasks.values("guardiantask__id")

```

```

        old_guardian_tasks =
old_guardian_tasks.exclude(guardiantask__id__in =
newer_guardian_tasks)
        old_guardian_tasks =
old_guardian_tasks.filter(guardianuser = guardian_user)
        old_guardian_tasks =
old_guardian_tasks.values("guardiantask")
        old_guardian_tasks_part2 = len(old_guardian_tasks)
        guardianusers_datacontext["taskstodo"] =
old_guardian_tasks_part1 + old_guardian_tasks_part2
        current_task =
Guardiantask.objects.select_related("commenterprisehall")
        current_time = datetime.datetime.now()
        current_task = current_task.filter(guardianuser =
guardian_user)
        current_task =
current_task.filter(taskendtime__isnull = True)
        current_task =
current_task.filter(taskstarttime__lte = current_time)
        commenterprise_hall =
current_task.commenterprisehall
        current_time = datetime.datetime(current_time.year,
current_time.month, current_time.day, 0,0)
        current_time2 = datetime.datetime(current_time.year,
current_time.month, current_time.day, 23, 59)
        secureincidents =
Secureincidentstatus.objects.select_related("secureincident")
        secureincidents = secureincidents.filter(status = 0)
        secureincidents =
secureincidents.filter(statustime__gte = current_time)
        secureincidents =
secureincidents.filter(statustime__lte = current_time2)
        secureincidents =
secureincidents.filter(commenterprisehall = commenterprise_hall)
        secureincidents =
secureincidents.values("secureincident")
        secureincidents = secureincidents.distinct()
        secureincidents = len(secureincidents)
        guardianusers_datacontext["newincidents"] =
secureincidents
        return render(request,
"guaderapp/guardiantrains.html", guardiantrain_datacontext)
    elif guardian_user and guardian_user.blocked:

```

```

        return render(request, "guaderapp/error.html",
{"error": "Ваш доступ заблоковано. Зверніться до керівництва"})
        return render(request, "guaderapp/error.html", {"error":
"Для доступу потрібна авторизація"})

def allguardiantrains_view(request):
    guardianusers_datacontext = {}
    if request.user.is_active:
        guardian_user = Guardianuser.objects.filter(user =
request.user)
        if guardian_user and not guardian_user.blocked and
guardian_user.companylevel:
            guardian_users = Guardianuser.objects.all()
            guardian_users =
guardian_users.order_by("shortname")
            guardianusers_datacontext["guardians"] =
guardian_users
            guardian_user_trains =
Guardiantrain.objects.filter(guardianuser__in = guardian_users)
            guardian_user_trains =
guardian_user_trains.order_by("-trainfrom")
            guardiantrain_datacontext["trains"] = guardian_users
            collaborationmessages =
Collaborationmessage.objects.filter(Q(collaboration__initialguardianus
er = guardian_user) | Q(collaboration__receiverguardianuser =
guardian_user))
            collaborationmessages =
collaborationmessages.filter(receiversaw = False)
            collaborationmessages =
collaborationmessages.exclude(senderguardianuser = guardian_user)
            collaborationmessages =
collaborationmessages.count()
            guardiantrain_datacontext["newmessages"] =
collaborationmessages
            current_time = datetime.datetime.now()
            current_time = datetime.datetime(current_time.year,
current_time.month, current_time.day, 0,0)
            current_time2 = datetime.datetime(current_time.year,
current_time.month, current_time.day, 23, 59)
            secureincidents =
Secureincidentstatus.objects.select_related("secureincident")
            secureincidents = secureincidents.filter(status = 0)

```

```

        secureincidents =
secureincidents.filter(statustime__gte = current_time, statustime__lte
= current_time2)
        secureincidents =
secureincidents.values("secureincident")
        secureincidents = secureincidents.distinct()
        secureincidents = len(secureincidents)
        guardiantrain_datacontext["newincidents"] =
secureincidents
        return render(request,
"guaderapp/allguardiantrains.html", guardiantrain_datacontext)
        elif guardian_user and guardian_user.blocked:
            return render(request, "guaderapp/error.html",
{"error": "Ваш доступ заблоковано. Зверніться до керівництва"})
            return render(request, "guaderapp/error.html", {"error":
"Для доступу потрібна авторизація"})

    def guardian_tasks_view(request, period, taskstatus, taskorder,
secure_task, guardianuserId = None):
        guardiantasks_datacontext = {}
        if request.user.is_active:
            guardian_user = Guardianuser.objects.filter(user =
request.user)
            if guardian_user and not guardian_user.blocked and
guardian_user.companylevel:
                guardian_tasks =
Guardiantask.objects.select_related("securetask")
                guardian_tasks =
guardian_tasks.select_related("guardianuser")
                guardian_tasks =
guardian_tasks.select_related("commententerprisehall")
                guardian_tasks = guardian_tasks.filter(guardianuser
= guardian_user)
                if secure_task >= 0:
                    try:
                        secure_task = Securetask.objects.get(id =
secure_task)
                    except Guardianuser.DoesNotExist:
                        return render(request,
"guaderapp/error.html", {"error": "Сталася помилка. Такого варіанта
завдання з охорони не існує"})
                    guardian_tasks =
guardian_tasks.filter(securetask = secure_task)

```

```

        current_time = datetime.datetime.now()
        if period == 0:
            guardian_tasks =
guardian_tasks.filter(taskstarttime__gte = current_time)
        elif period == 1:
            guardian_tasks =
guardian_tasks.filter(taskendtime__lte = current_time)
        elif period == 2:
            guardian_tasks =
guardian_tasks.filter(taskendtime__isnull = True, taskstarttime__lte =
current_time)

        if taskstatus > -1:
            guardiantask_statuses =
Guardiantaskstatus.objects.select_related("guardiantask")
            guardiantask_statuses =
guardiantask_statuses.filter(status = taskstatus, guardiantask__in =
guardian_tasks)

            guardiantask_statuses =
guardiantask_statuses.values("guardiantask__id")
            guardian_tasks =
Guardiantask.objects.select_related("securetask")
            guardian_tasks =
guardian_tasks.select_related("guardianuser")
            guardian_tasks =
guardian_tasks.select_related("commententerprisehall")
            guardian_tasks = guardian_tasks.filter(id__in =
guardiantask_statuses)

        if taskorder == 0:
            guardian_tasks =
guardian_tasks.order_by("taskstarttime")
        else:
            guardian_tasks = guardian_tasks.order_by("-
taskstarttime")

        guardiantasks_datacontext["tasks"] = guardian_tasks
        collaborationmessages =
Collaborationmessage.objects.filter(Q(collaboration__initialguardianus
er = guardian_user) | Q(collaboration__receiverguardianuser =
guardian_user))

        collaborationmessages =
collaborationmessages.filter(receiversaw = False)
        collaborationmessages =
collaborationmessages.exclude(senderguardianuser = guardian_user)

```

```

        collaborationmessages =
collaborationmessages.count()
        guardiantasks_datacontext["newmessages"] =
collaborationmessages
        current_time = datetime.datetime.now()
        current_time = datetime.datetime(current_time.year,
current_time.month, current_time.day, 0,0)
        current_time2 = datetime.datetime(current_time.year,
current_time.month, current_time.day, 23, 59)
        secureincidents =
Secureincidentstatus.objects.select_related("secureincident")
        secureincidents = secureincidents.filter(status = 0)
        secureincidents =
secureincidents.filter(statustime__gte = current_time, statustime__lte
= current_time2)
        secureincidents =
secureincidents.values("secureincident")
        secureincidents = secureincidents.distinct()
        secureincidents = len(secureincidents)
        guardiantasks_datacontext["newincidents"] =
secureincidents
        return render(request,
"guaderapp/guardiantasks.html", guardiantasks_datacontext)
        elif guardian_user and not guardian_user.blocked and not
guardian_user.companylevel:
            authguardian_user = =
Guardianuser.objects.filter(user = request.user)
            try:
                guardian_user = Guardianuser.objects.get(id =
guardianuserId)
            except Guardianuser.DoesNotExist:
                return render(request, "guaderapp/error.html",
{"error": "Сталася помилка. Такого працівника не існує"})
            guardian_tasks =
Guardiantask.objects.select_related("securetask")
            guardian_tasks =
guardian_tasks.select_related("guardianuser")
            guardian_tasks =
guardian_tasks.select_related("commententerprisehall")
            guardian_tasks = guardian_tasks.all()
            if secure_task >= 0:
                try:

```

```

        secure_task = Securetask.objects.get(id =
secure_task)

        except Guardianuser.DoesNotExist:
            return render(request,
"guaderapp/error.html", {"error": "Сталася помилка. Такого варіанта
завдання з охорони не існує"})

            guardian_tasks =
guardian_tasks.filter(securetask = secure_task)
            current_time = datetime.datetime.now()
            if period == 0:
                guardian_tasks =
guardian_tasks.filter(taskstarttime__gte = current_time)
            elif period == 1:
                guardian_tasks =
guardian_tasks.filter(taskendtime__lte = current_time)
            elif period == 2:
                guardian_tasks =
guardian_tasks.filter(taskendtime__isnull = True, taskstarttime__lte =
current_time)

                if taskstatus > -1:
                    guardiantask_statuses =
Guardiantaskstatus.objects.select_related("guardiantask")
                    guardiantask_statuses =
guardiantask_statuses.filter(status = taskstatus, guardiantask__in =
guardian_tasks)

                    guardiantask_statuses =
guardiantask_statuses.values("guardiantask__id")
                    guardian_tasks =
Guardiantask.objects.select_related("securetask")
                    guardian_tasks =
guardian_tasks.select_related("guardianuser")
                    guardian_tasks =
guardian_tasks.select_related("commenterprisehall")
                    guardian_tasks = guardian_tasks.filter(id__in =
guardiantask_statuses)

                    if taskorder == 0:
                        guardian_tasks =
guardian_tasks.order_by("taskstarttime")
                    else:
                        guardian_tasks = guardian_tasks.order_by("-
taskstarttime")

                    guardiantasks_datacontext["tasks"] = guardian_tasks

```

```

        collaborationmessages =
Collaborationmessage.objects.filter(Q(collaboration__initialguardianus
er = guardian_user) | Q(collaboration__receiverguardianuser =
guardian_user))
        collaborationmessages =
collaborationmessages.filter(receiversaw = False)
        collaborationmessages =
collaborationmessages.exclude(senderguardianuser = authguardian_user)
        collaborationmessages =
collaborationmessages.count()
        guardiantasks_datacontext["newmessages"] =
collaborationmessages
        old_guardian_tasks =
Guardiantaskstatus.objects.filter(status = 2)
        old_guardian_tasks =
old_guardian_tasks.filter(guardiantask__guardianuser = guardian_user)
        newer_guardian_tasks =
Guardiantaskstatus.objects.filter(guardiantask__guardianuser =
guardian_user)
        newer_guardian_tasks =
newer_guardian_tasks.exclude(status = 1)
        newer_guardian_tasks =
newer_guardian_tasks.values("guardiantask__id")
        old_guardian_tasks =
old_guardian_tasks.exclude(guardiantask__id__in =
newer_guardian_tasks)
        old_guardian_tasks =
old_guardian_tasks.values("guardiantask")
        old_guardian_tasks_part1 = len(old_guardian_tasks)
        current_time = datetime.datetime.now()
        old_guardian_tasks =
Guardiantaskstatus.objects.filter(status = 0,
guardiantask__taskstarttime__lt = current_time)
        newer_guardian_tasks =
Guardiantaskstatus.objects.filter(guardiantask__taskstarttime__lt =
current_time)
        newer_guardian_tasks =
newer_guardian_tasks.exclude(status = 0)
        newer_guardian_tasks =
newer_guardian_tasks.values("guardiantask__id")
        old_guardian_tasks =
old_guardian_tasks.exclude(guardiantask__id__in =
newer_guardian_tasks)

```

```

        old_guardian_tasks =
old_guardian_tasks.filter(guardianuser = guardian_user)
        old_guardian_tasks =
old_guardian_tasks.values("guardiantask")
        old_guardian_tasks_part2 = len(old_guardian_tasks)
        guardianusers_datacontext["taskstodo"] =
old_guardian_tasks_part1 + old_guardian_tasks_part2
        current_task =
Guardiantask.objects.select_related("commenterprisehall")
        current_time = datetime.datetime.now()
        current_task = current_task.filter(guardianuser =
guardian_user)
        current_task =
current_task.filter(taskendtime__isnull = True)
        current_task =
current_task.filter(taskstarttime__lte = current_time)
        commenterprise_hall =
current_task.commenterprisehall
        current_time = datetime.datetime(current_time.year,
current_time.month, current_time.day, 0,0)
        current_time2 = datetime.datetime(current_time.year,
current_time.month, current_time.day, 23, 59)
        secureincidents =
Secureincidentstatus.objects.select_related("secureincident")
        secureincidents = secureincidents.filter(status = 0)
        secureincidents =
secureincidents.filter(statustime__gte = current_time)
        secureincidents =
secureincidents.filter(statustime__lte = current_time2)
        secureincidents =
secureincidents.filter(commenterprisehall = commenterprise_hall)
        secureincidents =
secureincidents.values("secureincident")
        secureincidents = secureincidents.distinct()
        secureincidents = len(secureincidents)
        guardianusers_datacontext["newincidents"] =
secureincidents
        return render(request,
"guaderapp/guardiantasks.html", guardiantasks_datacontext)
        elif guardian_user and guardian_user.blocked:
            return render(request, "guaderapp/error.html",
{"error": "Ваш доступ заблоковано. Зверніться до керівництва"})

```

```

        return render(request, "guaderapp/error.html", {"error":
"Для доступу потрібна авторизація"})

    def guardian_task_view(request, guardiantaskId):
        guardiantask_datacontext = {}
        if request.user.is_active:
            guardian_user = Guardianuser.objects.filter(user =
request.user)
            if guardian_user and not guardian_user.blocked and
guardian_user.companylevel:
                try:
                    guardian_task =
Guardiantask.objects.select_related("guardianuser")
                    guardian_task =
guardian_task.select_related("securetask")
                    guardian_task =
guardian_task.select_related("commententerprisehall")
                    guardian_task =
guardian_task.select_related("managerguardian")
                    guardian_task = guardian_task.get(id =
guardiantaskId)
                except Guardiantask.DoesNotExist:
                    return render(request, "guaderapp/error.html",
{"error": "Сталася помилка. Такого завдання з охорони не існує"})
                    guardianusers_datacontext["task"] = guardian_task
                    guardiantask_statuses =
Guardiantaskstatus.objects.select_related("guardianuser")
                    guardiantask_statuses =
guardiantask_statuses.filter(guardiantask = guardian_task)
                    guardiantask_statuses =
guardiantask_statuses.order_by("-statustime")
                    guardianusers_datacontext["statuses"] =
guardiantask_statuses
                    collaborationmessages =
Collaborationmessage.objects.filter(Q(collaboration__initialguardianus
er = guardian_user) | Q(collaboration__receiverguardianuser =
guardian_user))
                    collaborationmessages =
collaborationmessages.filter(receiversaw = False)
                    collaborationmessages =
collaborationmessages.exclude(senderguardianuser = guardian_user)
                    collaborationmessages =
collaborationmessages.count()

```

```

        guardianusers_datacontext["newmessages"] =
collaborationmessages
        current_time = datetime.datetime.now()
        current_time = datetime.datetime(current_time.year,
current_time.month, current_time.day, 0,0)
        current_time2 = datetime.datetime(current_time.year,
current_time.month, current_time.day, 23, 59)
        secureincidents =
Secureincidentstatus.objects.select_related("secureincident")
        secureincidents = secureincidents.filter(status = 0)
        secureincidents =
secureincidents.filter(statustime__gte = current_time, statustime__lte
= current_time2)
        secureincidents =
secureincidents.values("secureincident")
        secureincidents = secureincidents.distinct()
        secureincidents = len(secureincidents)
        guardianusers_datacontext["newincidents"] =
secureincidents
        return render(request, "guaderapp/guardians.html",
guardianusers_datacontext)
        elif guardian_user and not guardian_user.blocked and not
guardian_user.companylevel:
            try:
                guardian_task =
Guardiantask.objects.select_related("guardianuser")
                guardian_task =
guardian_task.select_related("securetask")
                guardian_task =
guardian_task.select_related("commententerprisehall")
                guardian_task =
guardian_task.select_related("managerguardian")
                guardian_task = guardian_task.get(id =
guardiantaskId, guardianuser = guardian_user)
            except Guardiantask.DoesNotExist:
                return render(request, "guaderapp/error.html",
{"error": "Сталася помилка. Такого завдання з охорони не існує або
воно не призначено вам"})
                guardianusers_datacontext["task"] = guardian_task
                guardiantask_statuses =
Guardiantaskstatus.objects.select_related("guardianuser")
                guardiantask_statuses =
guardiantask_statuses.filter(guardiantask = guardian_task)

```

```

        guardiantask_statuses =
guardiantask_statuses.order_by("-statustime")
        guardianusers_datacontext["statuses"] =
guardiantask_statuses
        collaborationmessages =
Collaborationmessage.objects.filter(Q(collaboration__initialguardianus
er = guardian_user) | Q(collaboration__receiverguardianuser =
guardian_user))
        collaborationmessages =
collaborationmessages.filter(receiversaw = False)
        collaborationmessages =
collaborationmessages.exclude(senderguardianuser = guardian_user)
        collaborationmessages =
collaborationmessages.count()
        guardianusers_datacontext["newmessages"] =
collaborationmessages
        old_guardian_tasks =
Guardiantaskstatus.objects.filter(status = 2)
        old_guardian_tasks =
old_guardian_tasks.filter(guardiantask__guardianuser = guardian_user)
        newer_guardian_tasks =
Guardiantaskstatus.objects.filter(guardiantask__guardianuser =
guardian_user)
        newer_guardian_tasks =
newer_guardian_tasks.exclude(status = 1)
        newer_guardian_tasks =
newer_guardian_tasks.values("guardiantask__id")
        old_guardian_tasks =
old_guardian_tasks.exclude(guardiantask__id__in =
newer_guardian_tasks)
        old_guardian_tasks =
old_guardian_tasks.values("guardiantask")
        old_guardian_tasks_part1 = len(old_guardian_tasks)
        current_time = datetime.datetime.now()
        old_guardian_tasks =
Guardiantaskstatus.objects.filter(status = 0,
guardiantask__taskstarttime__lt = current_time)
        newer_guardian_tasks =
Guardiantaskstatus.objects.filter(guardiantask__taskstarttime__lt =
current_time)
        newer_guardian_tasks =
newer_guardian_tasks.exclude(status = 0)

```

```

        newer_guardian_tasks =
newer_guardian_tasks.values("guardiantask__id")
        old_guardian_tasks =
old_guardian_tasks.exclude(guardiantask__id__in =
newer_guardian_tasks)
        old_guardian_tasks =
old_guardian_tasks.filter(guardianuser = guardian_user)
        old_guardian_tasks =
old_guardian_tasks.values("guardiantask")
        old_guardian_tasks_part2 = len(old_guardian_tasks)
        guardianusers_datacontext["taskstodo"] =
old_guardian_tasks_part1 + old_guardian_tasks_part2
        current_task =
Guardiantask.objects.select_related("commenterprisehall")
        current_time = datetime.datetime.now()
        current_task = current_task.filter(guardianuser =
guardian_user)
        current_task =
current_task.filter(taskendtime__isnull = True)
        current_task =
current_task.filter(taskstarttime__lte = current_time)
        commenterprise_hall =
current_task.commenterprisehall
        current_time = datetime.datetime(current_time.year,
current_time.month, current_time.day, 0,0)
        current_time2 = datetime.datetime(current_time.year,
current_time.month, current_time.day, 23, 59)
        secureincidents =
Secureincidentstatus.objects.select_related("secureincident")
        secureincidents = secureincidents.filter(status = 0)
        secureincidents =
secureincidents.filter(statustime__gte = current_time)
        secureincidents =
secureincidents.filter(statustime__lte = current_time2)
        secureincidents =
secureincidents.filter(commenterprisehall = commenterprise_hall)
        secureincidents =
secureincidents.values("secureincident")
        secureincidents = secureincidents.distinct()
        secureincidents = len(secureincidents)
        guardianusers_datacontext["newincidents"] =
secureincidents

```

```
        return render(request, "guaderapp/guardians.html",
guardianusers_datacontext)
        elif guardian_user and guardian_user.blocked:
            return render(request, "guaderapp/error.html",
{"error": "Ваш доступ заблоковано. Зверніться до керівництва"})
            return render(request, "guaderapp/error.html", {"error":
"Для доступу потрібна авторизація"})
```

ДОДАТОК В
Слайди презентації



Рисунок В.1 – Титульний слайд

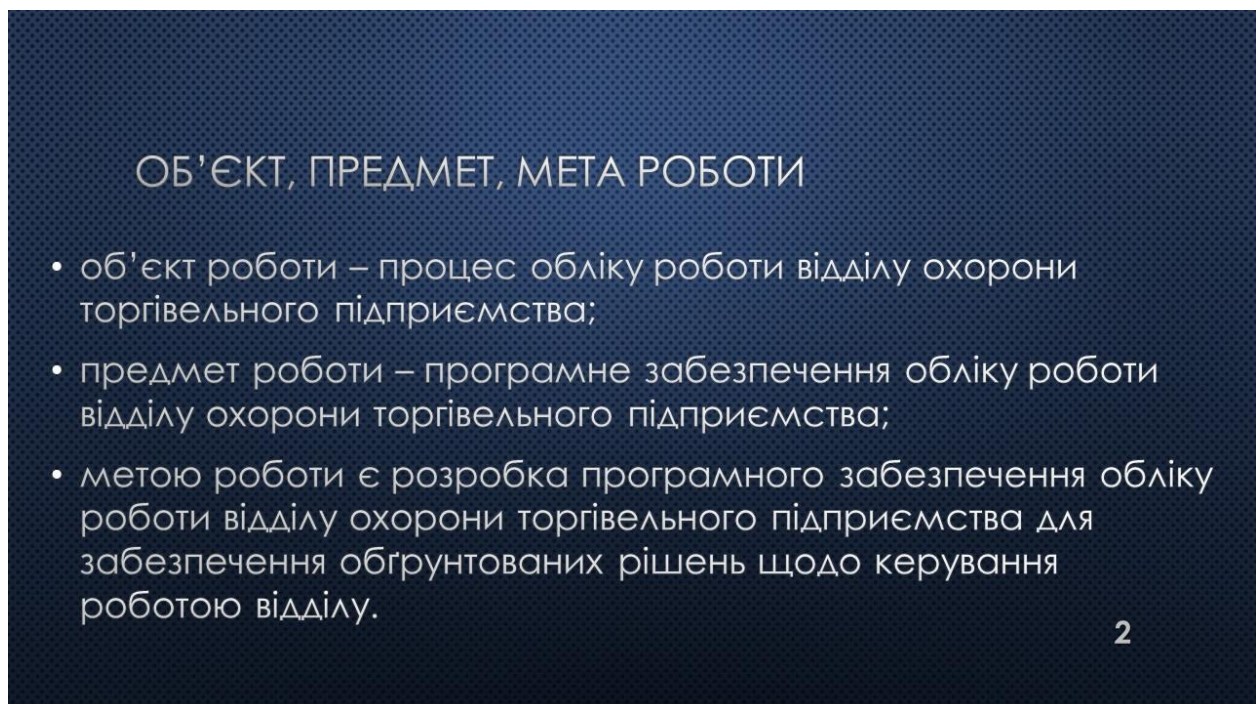


Рисунок В.2 – Об'єкт, предмет, мета роботи

ЗАВДАННЯ РОБОТИ

- проаналізувати предметну область обліку роботи відділу охорони торговельного підприємства;
- визначити засоби проектування програми;
- виділити вимоги до програмного забезпечення обліку роботи відділу охорони торговельного підприємства;
- виконати реалізацію програмного забезпечення обліку роботи відділу охорони торговельного підприємства.

3

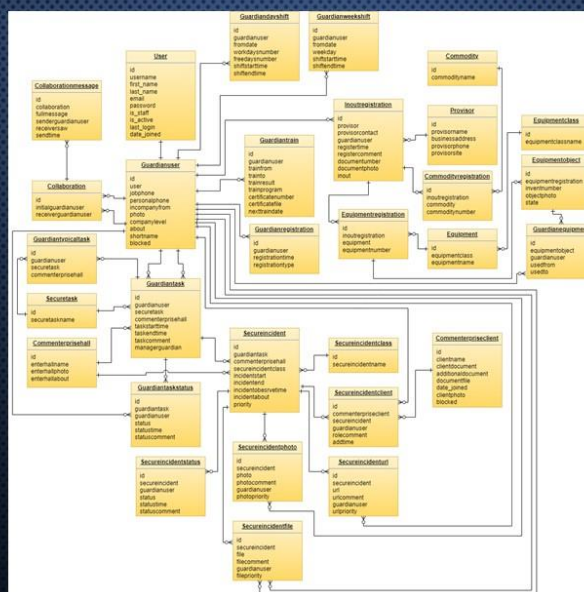
Рисунок В.3 – Завдання роботи

ПОРІВНЯННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ОБЛІКУ РОБОТИ ВІДДІЛУ ОХОРОНИ ТОРГІВЕЛЬНОГО ПІДПРИЄМСТВА З АНАЛОГОМ

Критерій	GuardsPro	Розробка
Призначення завдань з охорони працівникам	Так	Так
Визначення змін роботи працівників	Так	Так
Визначення інцидентів з безпеки	Так	Так
Внесення поставок товарів	Так	Так
Надсилання повідомлень між працівниками	Так	Так
Реєстрація переміщення обладнання	Ні	Так
Визначення тренувань персоналу	Ні	Так
Оплата за використання	Щомісячна за кожного користувача	Відсутня

4

Рисунок В.4 – Порівняння ПЗ обліку роботи відділу охорони торговельного підприємства з аналогом



7

ДІАГРАМА СТАНІВ ЗАВДАННЯ З ОХОРОНИ ТА ІНЦИДЕНТУ З БЕЗПЕКИ

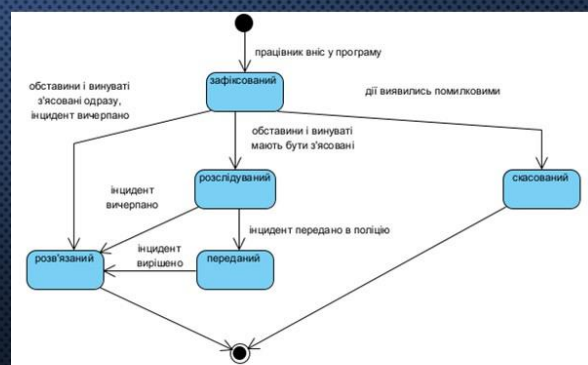
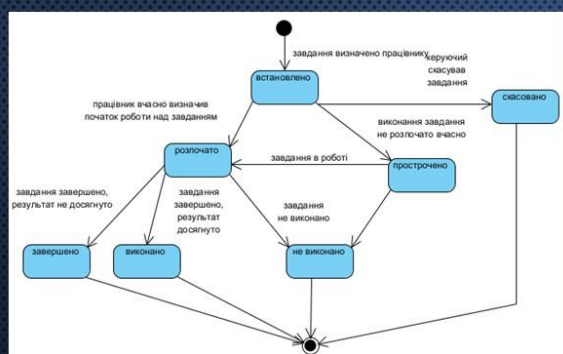


Рисунок В.8 – Діаграма станів завдання з охорони та інциденту з безпеки

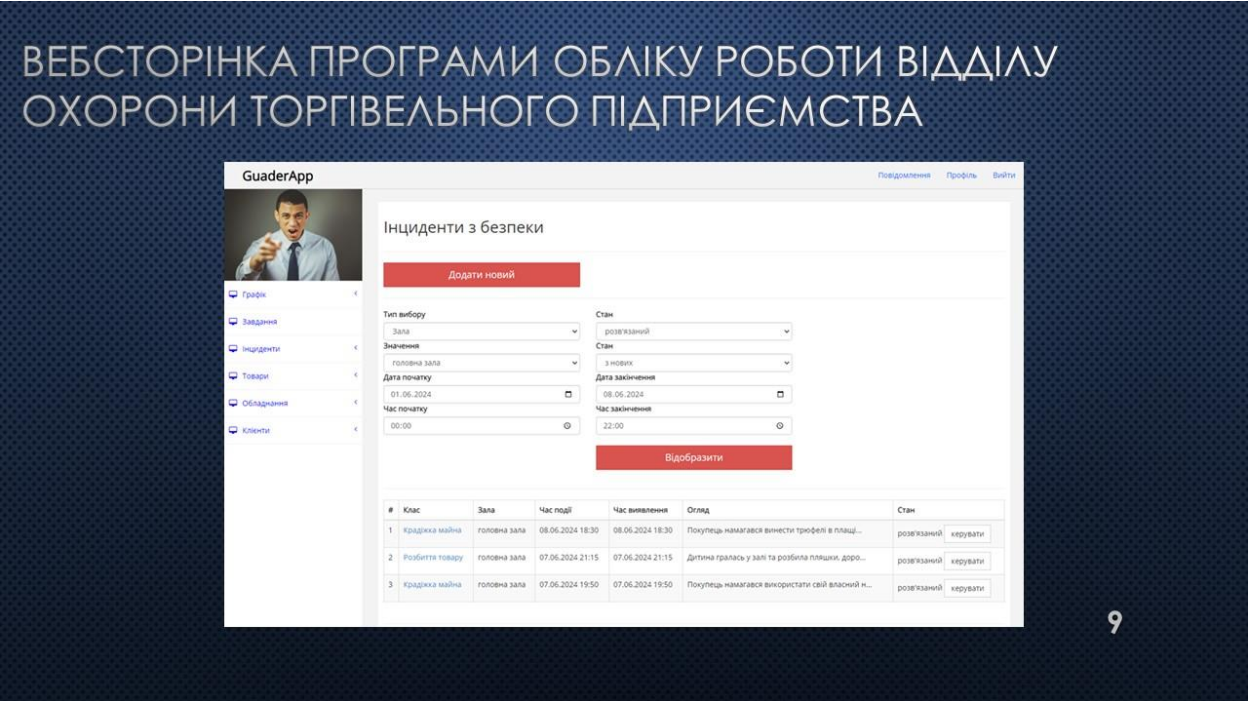


Рисунок В.9 – Вебсторінка програми обліку роботи відділу охорони торговельного підприємства

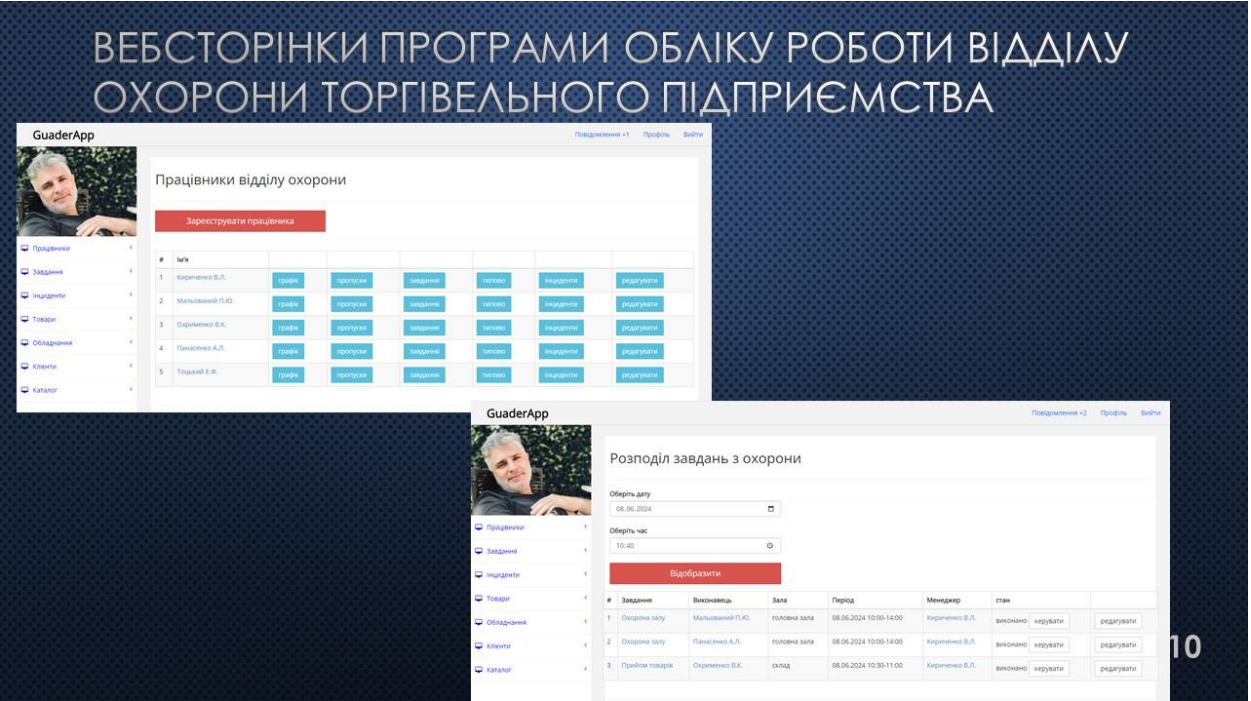


Рисунок В.10 – Вебсторінка програми обліку роботи відділу охорони торговельного підприємства

ВЕБСТОРІНКА ПРОГРАМИ ОБЛІКУ РОБОТИ ВІДДІЛУ ОХОРОНИ ТОРГІВЕЛЬНОГО ПІДПРИЄМСТВА

GuaderApp Повідомлення Профіль Вийти

Профіль працівника

Повне ім'я: Мальований Павло

Додаткові дані:

- Особистий телефон: 0504323232
- Робочий телефон: 0502896729
- користувачем є з: 02.06.2024 21:46
- остання авторизація: 08.06.2024 23:12
- без права керувати
- стан: активний

Опис:
Провівши себе лише позитивно, до роботи ставиться доволі сумлінно.

Тренування

#	Строки	Програма	Сертифікат	Дійсне до	Результат
1	01.09.2023-10.09.2023	Дії в умовах критичного загрози	BA903261	01.09.2024	Навички володіння зброєю...

Редагувати

Додати

11

Рисунок В.11 – Вебсторінка програми обліку роботи відділу охорони торговельного підприємства

ВИСНОВКИ

- виконана робота сумарно за результатами виконання всіх завдань дозволила досягти мету роботи, якою була розробка програмного забезпечення обліку роботи відділу охорони торговельного підприємства для забезпечення обґрунтованих рішень щодо керування роботою відділу;
- розроблена програма призначена для забезпечення обліку роботи відділу охорони торговельного підприємства, включаючи реєстрацію перебування працівників на робочому місці, графіків роботи, визначення завдань з охорони, інцидентів з безпеки, які сталися, отриманих поставачань та вибуття товарів, реєстрації видачі та повернення обладнання;
- усі завдання роботи виконано, а виконане тестування підтвердило забезпечення виконання реалізації всіх запланованих функцій з отриманням тих результатів, які визначали успішне виконання.

12

Рисунок В.12 – Висновки