МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) Автоматизації і комп'ютерних систем
Кафедра Інформаційних систем_____

«До захисту в ЕК»	**«До захисту допущено»**
Директор інституту(декан факультету)	Завідувач кафедри
_____ Андрій Форсюк	_____ Сергій Чумаченко
(підпис) (ім'я та прізвище)	(підпис) (ім'я та прізвище)
«___» _____ 2022 р.	«___» _____ 2022 р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

зі спеціальності 122 "Комп'ютерні науки"
(код та назва спеціальності)
освітньо-професійної програми Інформаційні управляючі системи та технології

на тему: Дослідження та розробка інформаційної системи підтримки обрання замовлень на виконання послуг системного адміністратор комп'ютерних мереж

Виконав: здобувач 2 курсу, групи ІУС-2-3М

Збаращук Павло Васильович _____
(прізвище, ім'я, по батькові повністю) (підпис)

Керівник Грибков Сергій Віталійович _____
(прізвище , ім'я та по батькові повністю) (підпис)

Консультанти _____ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

Рецензент Лідія Власенко _____
(ім'я та прізвище) (підпис)

Я як здобувач(ка) Національного університету харчових технологій розумію і підтримую політику університету з академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволеної допомоги під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Здобувач_____
(підпис)

Київ – 2022 р.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) Автоматизації і комп'ютерних систем
Кафедра Інформаційних систем
Освітній ступінь магістр
Спеціальність 122 «Комп'ютерні науки»
(код і назва)
Освітньо-професійна програма Інформаційні управляючі системи та технології
(назва)

ЗАТВЕРДЖУЮ
Завідувач
кафедри Інформаційних систем

Сергій Чумаченко
"____" _____2022 року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

Збаращук Павло Васильович
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження та розробка інформаційної системи підтримки обрання замовлень на виконання послуг системного адміністратор комп'ютерних мереж
керівник роботи Грибков Сергій Віталійович, д.т.н., доцент ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від " 11 " 11 2021 року № 884-кс
2. Строк подання здобувачем роботи 01.02.2022
3. Вихідні дані до роботи Інформація про ТОВ «Джи Ес Естейт». Положення про відділ технічного забезпечення ТОВ «Джи Ес Естейт». Посадові інструкції усіх працівників відділу технічного забезпечення. Звітна документація відділу технічного забезпечення про виконання замовлень. Сучасні публікації вітчизняних та зарубіжних авторів по створенню та розвитку інформаційних продуктів.
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
1. Вступ; 2. Дослідження предметної області та постановка задачі; 3. Дослідження методів, алгоритмів та програмних засобів розв'язання задачі; 4. Розробка та апробація інформаційної системи; 5. Основні висновки та результати; 6. Список використаних джерел; 7. Додатки
5. Перелік графічного матеріалу Діаграми функціональної моделі. Діаграми моделі даних. Скріншоти системи.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1, 2, 3	Грибков С. В., доцент	11.11.21	30.01.22

7. Дата видачі завдання 11.11.2021

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів виконання кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Дослідження предметної області та постановка задачі	11.12.2021-	Виконано
2	Дослідження методів, алгоритмів та програмних засобів розв'язання задачі	30.12.2021-	Виконано
3	Розробка та апробація інформаційної системи	30.12.2021-	Виконано
4	Оформлення роботи	30.01.2021-	Виконано
5	Підготовка автореферату	01.02.2021-	Виконано
6	Підготовка презентації		

Здобувач _____ Збаращук П. В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Грибков С. В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Збаращук П. В. — Дослідження та розробка інформаційної системи підтримки обрання замовлень на виконання послуг системного адміністратор комп'ютерних мереж

Кваліфікаційна робота на здобуття науково-освітнього ступеню магістр за спеціальністю 122 — Комп’ютерні науки — Національний університет харчових технологій — Київ, 2022 рік.

Кваліфікаційна робота на здобуття освітнього ступеню магістра присвячена розв’язанню науково-технічного завдання покращенню якості управління процесом обрання замовлень на виконання послуг системним адміністратором, на прикладі ТОВ «Джи Ес Естейт». Метою роботи є пошук алгоритмів та підходів для підтримки прийняття рішень для ефективного обрання замовлень на послуги системного адміністратора комп'ютерних мереж, а також створення інформаційної системи для підтримки розв’язання поставленої задачі. Досліджено алгоритми та підходи для підтримки прийняття рішень для ефективного обрання замовлень на послуги системного адміністратора комп'ютерних мереж. Досліджено сучасні підходи та технології по створенню програмних продуктів. Визначено оптимальний набір інструментів для реалізації інформаційної системи підтримки обрання замовлень на виконання послуг системним адміністратором комп'ютерних мереж. Розроблено інформаційна система підтримки обрання замовлень на виконання послуг системним адміністратором комп'ютерних мереж на основі обраних алгоритмів та методів.

КЛЮЧОВІ СЛОВА: ВИКОНАННЯ ЗАМОВЛЕНЬ, СИСТЕМНИЙ АДМІНІСТРАТОР, КОМП’ЮТЕРНА МЕРЕЖА, АЛГОРИТМИ, ГЕНЕТИЧНИЙ АЛГОРИТМ.

ANNOTATION

Zbarashchuk P.V. «Research and development of information system to support the selection of orders for the services of system administrator of computer networks».

Qualification work for the scientific and educational degree of Master in 122 «Computer Science» National University of Food Technologies, Kyiv, 2022.

Qualifying work for the master's degree is devoted to solving the scientific and technical problem of improving the quality of management of the process of selecting orders for services by the system administrator, on the example « Джи Ес Естейт ». The aim of the work is to find algorithms and approaches to support decision-making to effectively select orders for the services of system administrator of computer networks, as well as to create an information system to support the solution of the problem. Algorithms and approaches to support decision making for effective selection of orders for the services of system administrator of computer networks are studied. Modern approaches and technologies for creating software products are studied. The optimal set of tools for the implementation of the information system to support the selection of orders for services by the system administrator of computer networks is determined. The information system of support of selection of orders for performance of services by the system administrator of computer networks on the basis of the chosen algorithms and methods is developed.

KEY WORDS: FULFILLMENT OF ORDERS, SYSTEM ADMINISTRATOR, COMPUTER NETWORK, ALGORITHMS, GENETIC ALGORITHM.

ЗМІСТ

ВСТУП

Актуальність теми У багатьох компаніях в яких є відділ технічної підтримки та власне наявність системних адміністраторів у даному відділі гостро стоїть питання оптимізації праці. Підприємство може бути не ефективним, якщо на відділ технічної підтримки відправляється велика кількість завдань і не існує чіткої системи вибору та сортування завдань, які потрібно своєчасно і в певному порядку пріоритету виконувати.

Особливо гостро, така проблема стоїть коли такі замовлення надходять на фірму, яка займається обслуговуванням декількох офісів. Тому гостро стоїть задача про розподілення задач між виконавцями таким чином, щоб забезпечити максимальний заробіток для них.

Враховуючи складність системи необхідно підібрати підходи по їх вирішенню, а також розробити систему, яка забезпечить підтримку обрання замовлень на виконання послуг системним адміністратором комп'ютерних мереж.

Необхідно чітко спланувати виконання робіт для їх виконання, адже вибір правильної методології створення додатку прийому замовлень системним адміністратором - перший крок до успіху.

Тому актуальною науково-практичною задачею є створення інформаційної системи підтримки обрання замовлень на виконання послуг системним адміністратором комп'ютерних мереж з використанням сучасних підходів для отримання максимального прибутку виконавцем цих послуг.

Предметом дослідження є процес виконання замовлень на послуги системного адміністратора комп'ютерних мереж.

Об'єктом дослідження. Алгоритми та підходи для підтримки прийняття рішень для ефективного обрання замовлень на послуги системного

адміністратора комп'ютерних мереж.

Зв’язок роботи з науковими програмами, планами, темами.

Наукова робота виконується згідно з планом та програмою наукових досліджень на кафедрі інформаційних систем Національного університету харчових технологій (НУХТ): № ДР 0117U003475 «Дослідження та впровадження інформаційних технологій у галузях харчової промисловості та освіті» (на червень 2017 — травень 2022 р.).

Метою роботи є пошук алгоритмів та підходів для підтримки прийняття рішень для ефективного обрання замовлень на послуги системного адміністратора комп'ютерних мереж, а також створення інформаційної системи для підтримки розв’язання поставленої задачі.

Виходячи з мети, були визначені наступні завдання дослідження:

1. дослідити алгоритмів, підходів та програмних засобів для підтримки прийняття рішень для ефективного обрання замовлень на послуги системного адміністратора комп'ютерних мереж та обрати найкращі;
2. визначити оптимальний набір інструментів для реалізації інформаційної системи підтримки обрання замовлень на виконання послуг системним адміністратором комп'ютерних мереж;
3. розробити інформаційної системи підтримки обрання замовлень на виконання послуг системним адміністратором комп'ютерних мереж на основі обраних алгоритмів та методів.

Методи дослідження та розробки: методи структурно-функціонального аналізу для дослідження предметної області; методи моделювання та побудови баз даних, для аналізу та побудови інформативного забезпечення системи; метод рюкзака та модифіковані генетичні алгоритми для розв’язання задачі обрання замовлень; об’єктно-орієнтоване програмування для створення елементів системи.

Наукова новизна одержаних результатів полягає у застосуванні генетичного алгоритму для ефективного обрання замовлень на послуги системного адміністратора.

Наукове значення роботи: полягає в удосконаленні процесу прийняття рішень для ефективного обрання замовлень на послуги системного адміністратора комп'ютерних мереж на основі обґрунтованих методів розв’язання задачі про рюкзак.

Практичне значення отриманих результатів полягає у створенні інформаційної системи підтримки обрання замовлень на виконання послуг системним адміністратором комп'ютерних мереж на основі обґрунтованих методів розв’язання задачі про рюкзак.

Апробація результатів магістерської роботи. Основні положення та результати наукової роботи були подані на Міжнародну науково-практичну конференцію «Сучасні технології розвитку інформаційних систем і телекомунікаційних технологій».

Структура та обсяг роботи. Магістерська робота складається зі вступу, трьох розділів, висновків, списку використаних джерел із 23 найменувань. Загальний обсяг магістерської роботи складає 118 сторінок, робота містить 11 рисунків , 5 таблиць, 7 додатків.

РОЗДІЛ 1. ОГЛЯД ТА ДОСЛІДЖЕННЯ МЕТОДОЛОГІЙ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1. Загальна характеристика ТОВ «Джи Ес Естейт»

ТОВ «Джи Ес Естейт» – підприємство яке займається наданням кваліфікованої технічної підтримки офісів, по всій Україні. Місце знаходження м. Київ вул. Анни Ахматової, 14. Заснована компанія у вересні 2010 року.

ТОВ «Джи Ес Естейт» надає послуги з ремонту та обслуговуванні мереж та офісної техніки за замовленням від різних офісів з якими укладений договір. Послуги, що надаються компанією, як правило стосуються проблем, що виникають з технічної сторони та потребують виділення часу для їх усунення.

Компанія має кваліфікований відділ технічної підтримки в повному складі, а саме:

- керівник відділу;
- спеціаліст початкового рівня;
- системний адміністратор;
- системний архітектор інженер;
- мережевий адміністратор;
- інженер інформаційної безпеки.

Кількість кваліфікованих працівників регламентується по відношенню до розміру підприємства з яким підписується контракт.

Ціллю ТОВ «Джи Ес Естейт» є створення масштабної мережі в середні країни, яка скрадатиметеся з підприємств, які користуватимуться послугами ТОВ «Джи Ес Естейт», та розширення можливості навчання та формування команд спеціалістів які надалі будуть надавати кваліфіковану технічну підтримку. Необхідно відмітити, що потреба кваліфікованих спеціалістів зростає з кожним днем, адже весь час зростає кількість компаній та складність техніки.

1.2. Структура відділу технічної підтримки підприємства

Структура відділу технічної підтримки підприємства фактично має лінійно ієрархічну структуру, що відображена на рисунку 1.1. В залежності від складності підприємства кожен рівень може налічувати до декількох десятків чоловік.

Рис. 1.1. Структура відділу технічної підтримки підприємства

Залежно від рівня і спеціалізації підготовки розрізняють наступні типи системних адміністраторів: спеціаліст початкового рівня (AnyKey від виразу «Press any key to continue»), системний адміністратор (Helpdesk administrator),

системний архітектор-інженер, мережевий адміністратор, інженер інформаційної безпеки.

Навики системного адміністратора засновані на знаннях комп'ютерних систем і ефективних способів їх використання працівниками організації. Він повинен знати операційні системи і програмні середовища, а також мету, з якою співробітники організації застосовують їх, а також вміти усувати пошкодження апаратного та програмного забезпечення.

Необхідно відмітити, дуже часто системний адміністратор повинен виконувати задачі які можуть охоплювати технічно-апаратні елементи розташовані в різних місцях, що призводить до затримки виконання замовлень та потребує виконання їх в певному порядку. Системний адміністратор повинен швидко і правильно діагностувати й найкращим чином усувати несправності у комп'ютерних системах.

Також, необхідно відмітити, що системний адміністратор не є розробником програмного забезпечення. Зазвичай він не зобов'язаний писати прикладні чи системні програми. Однак, він повинен розуміти призначення й поведінку програмного забезпечення в разі його розгортання чи застосування, пошуку помилок у програмах, а отже знати декілька мов програмування, щоб використовувати їх для написання скриптів (сценаріїв), які дозволяють автоматизувати рутинні завдання.

При використанні інтернет-орієнтованих або бізнес-критичних систем системний адміністратор повинен особливо надійно контролювати безпеку комп’ютерної мережі. Це означає не тільки своєчасне оновлення програмного забезпечення, але й попереджувальні заходи проти зламу системи захисту і вторгнення в комп'ютерну систему. В деяких організаціях за безпеку комп'ютерної мережі й підтримку брандмауера (фаєрвола) відповідає адміністратор захисту мережі, але кожен системний адміністратор значною мірою здатен підтримувати безпеку системи.

Системні адміністратори протидіють атакам зломщиків і сприяють безпечному спілкуванню всередині інфраструктури організації, а також за її межами. Він, у певному сенсі, також комп'ютерний зломщик, бо він повинен знати всі ті способи зламу й обходу захисту (наприклад брандмауера), які застосовують зломщики. Однак у більшості організацій обов'язками системного адміністратора є не тільки спостереження за безпекою мережі організації, але й інші супутні проблеми: боротьба з комп'ютерними вірусами, налаштування програмного забезпечення користувачів та ін.

Через швидкий розвиток Інтернету і мережевих технологій системному адміністратору-одинаку щоразу складніше протистояти всім проблемам, тому постійно діють спеціалізовані інтернет-форуми й друковані видання, спрямовані на поглиблення знань системного адміністратора і надання допомоги у вирішенні різноманітних проблем.

1.3. Завдання які стоять перед системним адміністратором

До основних завдань системного адміністратора доречно віднести

- підготовка й збереження резервних копій даних, їх періодична перевірка й знищення;
- встановлення й конфігурування оновлень операційної системи і прикладного програмного забезпечення;
- встановлення й конфігурування нового апаратного й програмного забезпечення;
- створення й підтримка в актуальному стані файлу облікових записів користувачів;
- підтримання інформаційної безпеки в організації;

- документування своєї роботи;
- усунення неполадок у комп'ютерній системі;
- монтаж комп'ютерної техніки та визначення необхідності ремонту;
- участь у проектуванні та монтажі локальної мережі;
- участь у плануванні комп'ютерних систем та придбанні нової комп'ютерної техніки.

1.4. Функціональне моделювання діяльності роботи відділу технічної підтримки

Для детального вивчення та виявлення проблеми в діяльності роботи відділу технічної підтримки використовується вже створений інструментальні засоби, які використовуються для моделювання підприємства і реінжинірингу ділових процесів. Одним з представників такого сімейства інструментальних засобів є CASE-інструменти для функціонального моделювання ділових процесів Allfusion Process modeler. В основі Allfusion Process modeler закладена методологія IDEF0 (Integrated Definition Function Modeling), яка в даний час прийнята в якості федерального стандарту США [1].

В основі IDEF0 методології лежить поняття блоку, який відображає деяку бізнес-функцію. Чотири сторони блоку мають різну роль: ліва сторона має значення «входу», права – «виходу», верхня – «управління», нижня – «механізму».

Взаємодія між функціями в IDEF0 представляється у вигляді дуги, яка відображає потік даних або матеріалів, що надходить з виходу однієї функції на вхід іншої. В залежності від того, з яким боком блоку пов’язаний потік, його називають відповідно «вхідним», «вихідним», «керуючим».

При такому підході на перший план виступає проблема ефективного способу вивчення сфери діяльності замовника:

- обстеження існуючої бізнес-архітектури, ділових процесів, бізнес-правил, інформаційних потоків;
- ідентифікація проблем, «вузьких» місць, що негативно впливають на ефективність діяльності підприємства;
- розробка та реалізація заходів щодо усунення наявних проблем і зміни бізнес-архітектури підприємства, перебудові ділових процесів;
- розробка конкретного проекту корпоративної інформаційної системи, реалізація цього проекту та супровід в майбутньому.

На рисунку 1.2 представлена контекстна діаграма функціональної моделі діяльності відділу технічної підтримки.

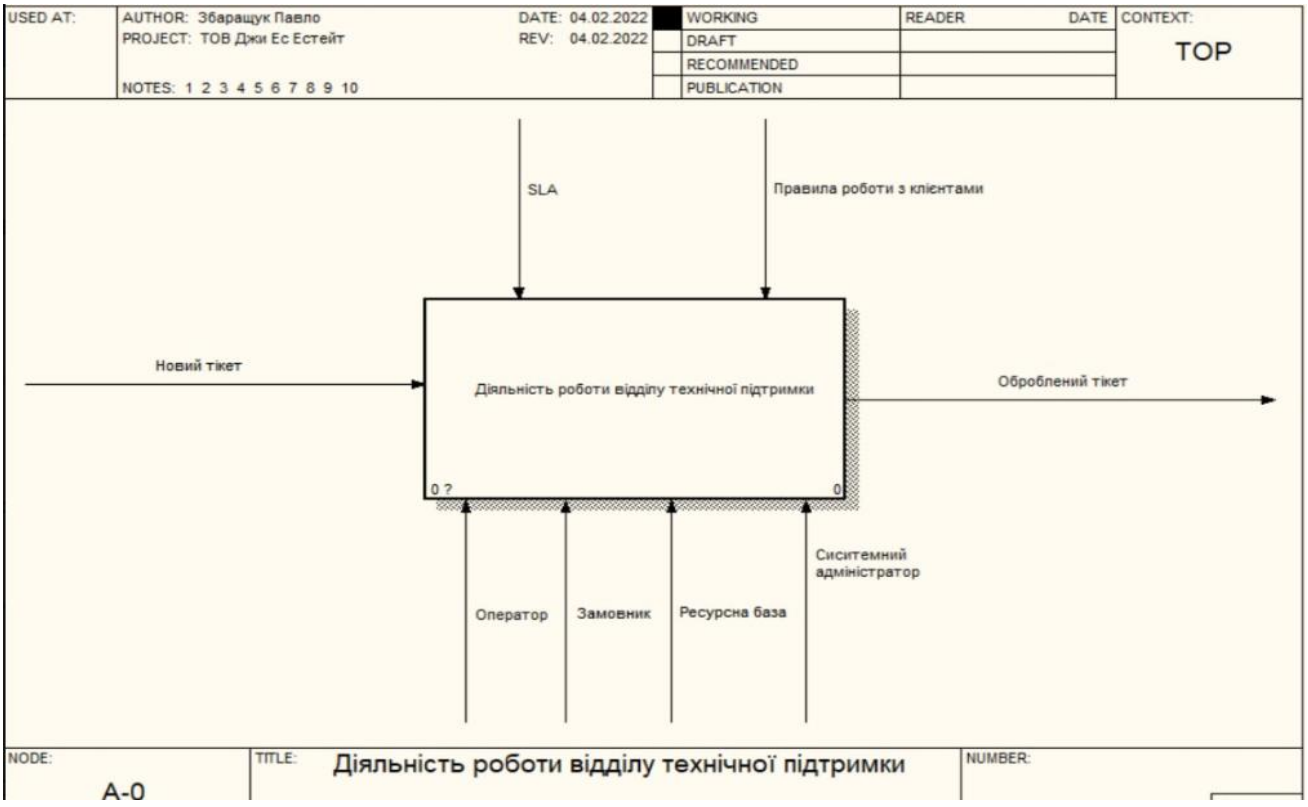

Рис. 1.2. Контекстна діаграма функціональної моделі

До вхідних даних контекстної діаграми відноситься створення тікетів,

тобто оформлення проблеми співробітника у вигляді задачі на виконання працівником відділу технічної підтримки.

До вихідних даних контекстної діаграми відноситься: оброблений тікет, тобто виконання поставленої задачі.

До управління відносять: правила роботи з клієнтами – правила комунікації з користувачами, що затверджені в договорі; SLA –коефіцієнт затримки при виконанні поставленого завдання, тобто час який перевищує виставленні рамки на виконання певного завдання. Крім того в якості управління використовується: Закони України, Кодекс законів про працю України, інформаційну безпеку, тарифно-кваліфікаційні характеристики, положення про корпоративну інформацію з обмеженим доступом.

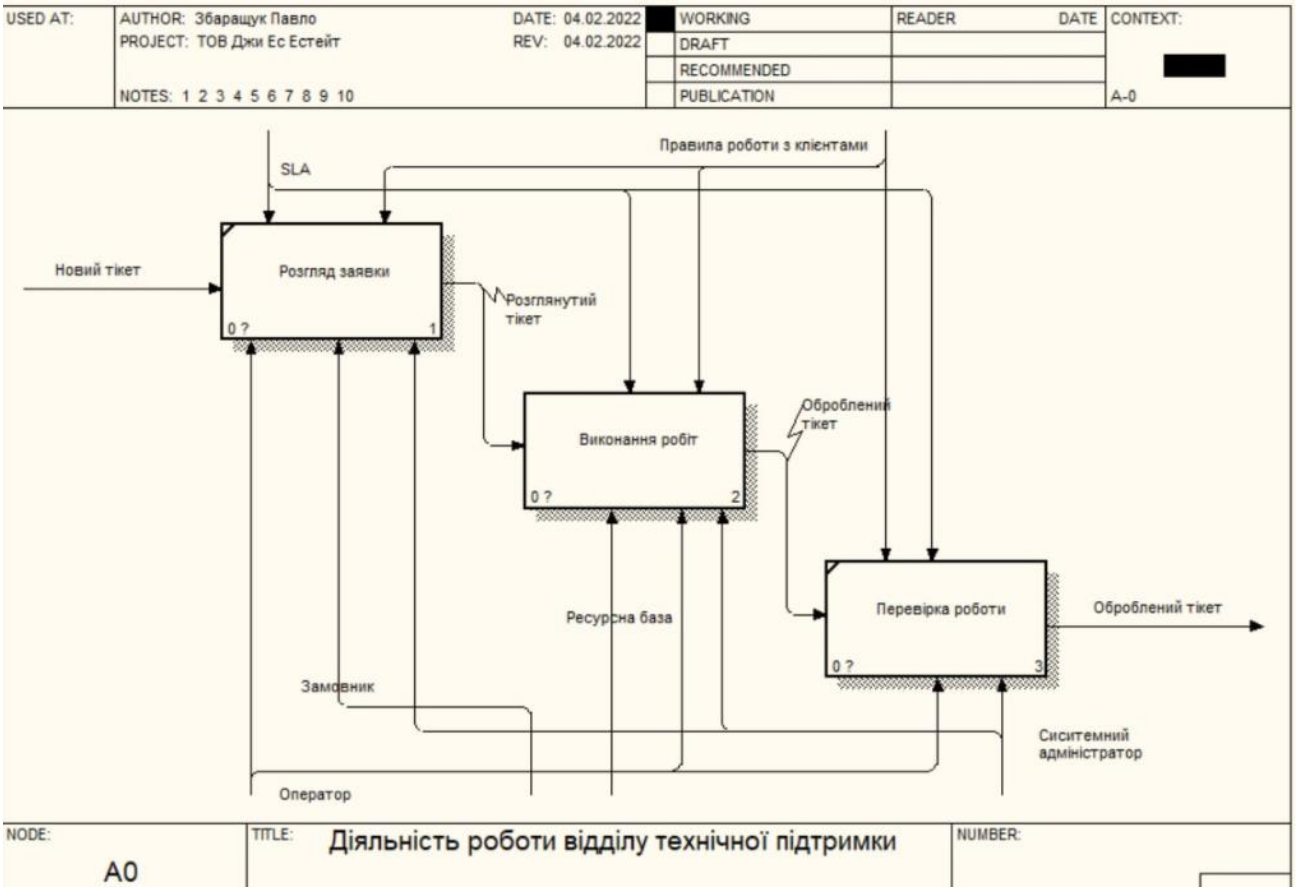

Рис. 1.3. Перший рівень декомпозиції функціональної моделі

На першому рівні декомпозиції діяльність відділу розбита на блоки робіт (рисунок 1.3): «Розгляд заявки», «Виконання робіт», «Перевірка роботи».

На рисунку 1.4 представлено декомпозиція функції «Виконання робіт».

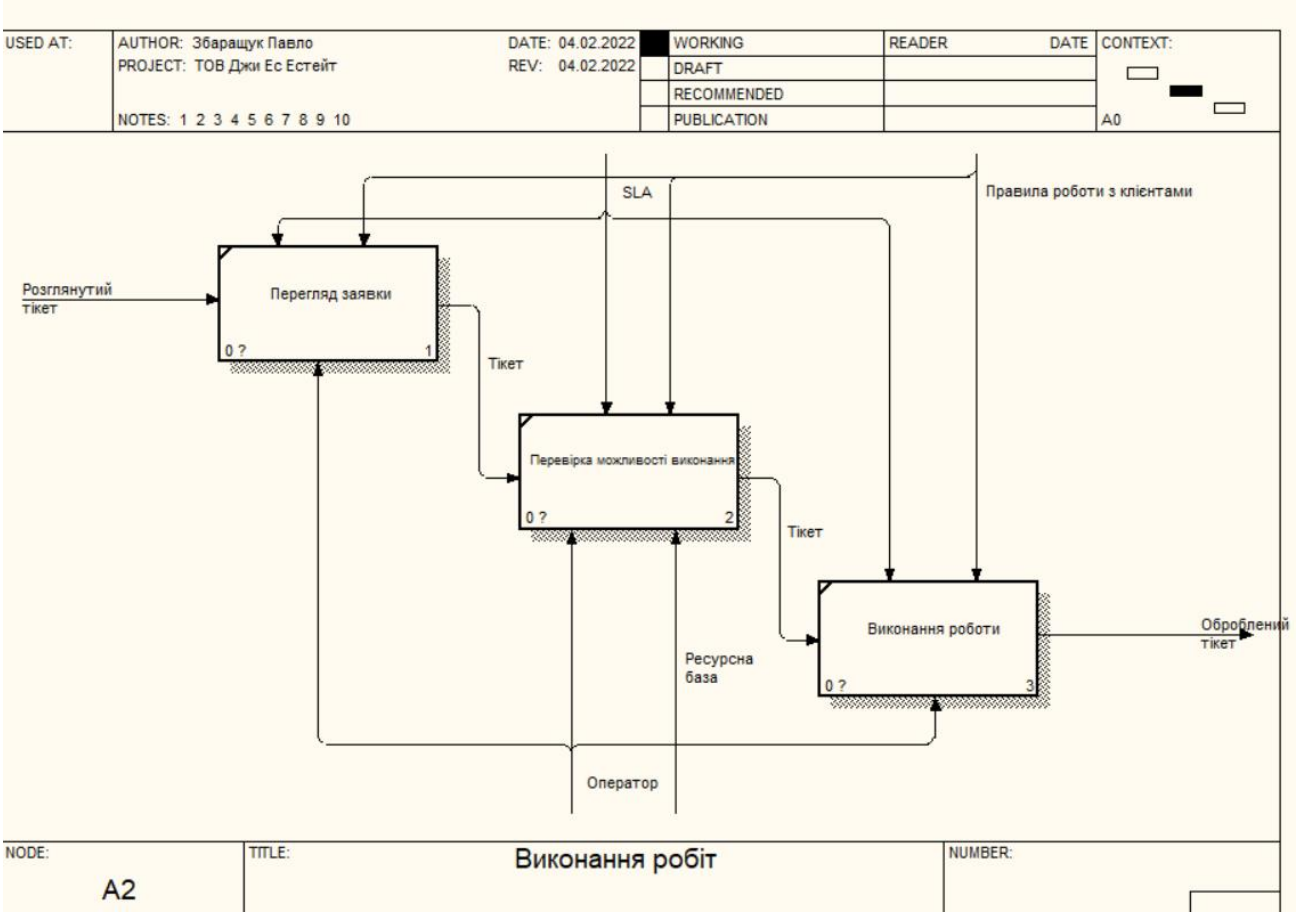

Рис. 1.4. Декомпозиція блоку «Виконання робіт»

Функціонально-вартісний аналіз (ФВА, Activity Based Costing, ABC) – метод визначення вартості та інших характеристик виробів, послуг і споживачів, що використовують як основи функції та ресурси, задіяні у виробництві, маркетингу, продажу, доставці, технічній підтримці, наданні послуг, обслуговуванні клієнтів, а також забезпечення якості [1].

ФВА-метод – один з методів, що дозволяє вказати на можливі шляхи

покращення вартісних показників в нашій системі використовується для первинного аналізу досліджуваного процесу [2].

Функціонально-вартісний аналіз дозволяє виконати наступні види робіт [1]:

1. визначення та проведення загального аналізу собівартості бізнес-процесів на підприємстві (маркетинг, виробництво продукції і надання послуг, збут, менеджмент якості, технічне та гарантійне обслуговування тощо;
2. проведення функціонального аналізу, пов'язаного з встановленням і обґрунтуванням виконуваних структурними підрозділами підприємств функцій з метою забезпечення випуску високої якості продукції та надання послуг;
3. визначення та аналіз основних, додаткових і непотрібних функціональних витрат;
4. порівняльний аналіз альтернативних варіантів зниження витрат у виробництві, збуті та управлінні за рахунок впорядкування функцій структурних підрозділів підприємства;
5. аналіз інтегрованого поліпшення результатів діяльності підприємства.

В результаті функціонального моделювання та функціонально-вартісного аналізу було виявлено, що вузьким місцем є процес розподілення та вибору задач для кожного виконавця. Тому необхідно удосконалити саме цю функцію.

1.5. Формування задач обрання задач для системного адміністратора на основі задачі про пакування рюкзака

Використання задачі про пакування рюкзаку обране тому, що адміністратори та обслуговуючий персонал обирають собі задачі в залежності від

своєї кваліфікації. У випадку, коли замовлення є терміновим, менеджер це замовлення відмічає як пріоритетне для певного виконавця.

Необхідно відзначити, що така задача відноситься до числа широко відомих задач комбінаторної оптимізації [3]. Найчастіше ця задача вирішується при необхідності розподілення обмеженої кількості ресурсів для отримання найбільшої сумарної користі, як при завантаженні човнів або літаків, вибору багажів для оптимального завантаження транспортного засобу тощо [4]. При застосуванні до нашої задачі, умовою пошуку є знаходження такого розподілення між усіма виконавцями робіт, при якому вони отримують максимальну користь від виконання замовлень, що в кінцевому результаті буде стимулювати їх як виконавців.

Для випадків, коли вхідної інформації для розв'язку задачі достатньо, а розмірність не дуже велика, доцільно застосовувати алгоритм розв'язку, представлений в роботі [5]. Вхідні дані (ВД) представлені масивом, що містять цілу вагу *W* та матеріальну цінність *P* предметів $W(1...N) > 0$ і $P(1...N) > 0$, де *N* – число предметів і $C > 0$ – вміщення рюкзака. Вихідні дані, що представляє собою масив логічних даних $X(1...N)$, де $X(i) = 1$, якщо предмет з номером *i* входить у рішення, і $X(i) = 0$, якщо предмет з номером *i* не входить у рішення.

START:

Eman 1: сортування вхідних даних в порядку зменшення окремої вартості предметів: $P(1)/W(1) >= P(2)/W(2) >= ...>= P(i)/W(i)>=... >= P(N) / W(N)$, де $P(i)>0$ – вартість предмету *i* , $W(i) >0$ – вага предмета *i*. У масиві $X(1...N)$ всі елементи початково дорівнюють 0. Для зменшення потреб в пам'яті для алгоритму визначають мінімальну вагу у наборі вхідних даних $W\_min = min(W)$.

Eman 2: Ініціалізація робочих масивів – створюємо масив дійсних чисел *LP* розмірністю $(W\_min... C)$ і масив цілих чисел *LCr* розмірністю $(W\_min... C)$. Заносимо в масив *LP* і *LCr* дані першого елемента з відсортованого списку

вхідних даних, де *P(1)* – вартість і *W(1)* - вага першого предмета у відсортованому списку вхідних даних.

Етап 3: Заповнення робочих масивів. Нехай *W(i)* та *P(i)* – вага та вартість поточного елемента вхідного набору. Створюємо порожній масив дійсних чисел *Clone* розмірністю *(W_min…C)*. Вносимо до масиву *Clone* вартість поточного елемента вхідного масиву. Копіюємо в масив *Clone* ненульові дані з масиву *LP*, додаючи вартість *P(i)* поточного елемента і збільшуючи його індекс на його вагу *W(i)* за умови, що індекс *Clone* не перевершить місткості рюкзака *C*. Проводимо модифікацію масивів *LP*, *LCr* на основі даних масиву *Clone*. Оновлюємо в масивах *LP* та *LCr* тільки ті елементи, вартість яких у *Clone* більша, ніж у *LP*.

Етап 4: Формування результату, зворотний спуск. У масиві *LP* знаходимо максимальне значення вартості *Pmax = MAX(LP)*, що і буде оптимальним рішенням. Індекс знайденого у масиві елемента дорівнює вазі рішення, позначимо його *Wr*, тобто *LP(Wr) = Pmax*. Записуємо перший знайдений елемент у X.

Повторюємо етапи 2, 3, 4 (тільки на етапі 2 масиви *LP* та *LCr* не створюємо, а заповнюємо нулями). Повторювати етап 1 не потрібно. Це по суті рекурсія, але через попереднє сортування вхідних даних вона буде не глибокою. На деяких наборах вхідних даних рекурсії взагалі може не бути. При повторенні розрахунків розглядаємо ті вхідні дані, індекс яких менше *LCr(Wr)* і знижуємо необхідний розмір рюкзака до досягнутої ваги *Wr*.

Кінець: Вартість знайденого рішення буде розрахована як *Σ P(i) X(i)*, а також вага *Σ W(i) X(i)*.

Правильність розрахунку підсумкової вартості рюкзака легко доводиться за індукцією. Відновлення оптимального набору предметів теж не викликає труднощів.

Необхідно відзначити наступні зауваження, що наводились у роботах [6-7]. Загальна складність представленого алгоритму складається із складності

сортування вхідних даних та складності виконання етапу 3 алгоритму (з урахуванням числа ітерацій). Час роботи 3-го етапу пропорційний кількості предметів на місткість рюкзака *(N * C)*. Наперед визначити кількість ітерацій досить складно. Число ітерацій може варіюватися від 0 до числа предметів у рішенні *X (i)*. На кожній ітерації, що виникає на етапі 4, обсяг обчислень на етапах 2, 3 зменшується. Верхня оцінка тимчасової складності всього алгоритму визначається *N * C * (число ітерацій + 1).*

Потреба алгоритму в пам'яті пропорційна місткості рюкзака *C* і залежить від кількості предметів у вхідному наборі даних *N*, що вигідно відрізняє його від методу динамічного програмування.

Внутрішні цикли етапу 3 легко виконуються паралельно.

При великому розкиді питомої вартості предметів, якщо на етапі 3 алгоритму у верхній частині масиву *LP* перестають відбуватися зміни, можна переривати етап 3 і не розглядати предмети, що залишилися, з низькою питомою вартістю.

Якщо місткість рюкзака досить велика, так що масиви розмірності не можуть бути створені з технічних причин або ваги предметів є дійсними числами, то запропонований алгоритм може бути легко модифікований заміною масивів зв'язаними списками [7].

1.6. Постанова задачі

Метою роботи є пошук алгоритмів та підходів для підтримки прийняття рішень для ефективного обрання замовлень на послуги системного адміністратора комп'ютерних мереж, а також створення інформаційної системи для підтримки розв’язання поставленої задачі.

Виходячи з мети, були визначені наступні завдання дослідження:

1. Дослідити алгоритмів, підходів та програмних засобів для підтримки прийняття рішень для ефективного обрання замовлень на послуги системного адміністратора комп'ютерних мереж та обрати найкращі.

2. Визначити оптимальний набір інструментів для реалізації інформаційної системи підтримки обрання замовлень на виконання послуг системним адміністратором комп'ютерних мереж;

3. Розробити інформаційної системи підтримки обрання замовлень на виконання послуг системним адміністратором комп'ютерних мереж на основі обраних алгоритмів та методів.

РОЗДІЛ 2. Дослідження методів та алгоритмів розв'язання задачі формування поставленої задачі

2.1 Дослідження методологій по створенню інформаційних систем

Для того щоб зрозуміти, що таке методологія розробки програмного забезпечення (далі ПЗ), потрібно сформулювати цілі, без яких неможливе створення ІТ продукту [8]. Головним завданням будь-якого успішного проекту є умова того, щоб на момент запуску системи і протягом усього терміну життєвого циклу можна було забезпечити:

- необхідну функціональність системи і адаптацію до зміни вимог щодо її функціональності;
- прийнятний час відгуку і реакції системи на запит;
- необхідну пропускну здатність ПЗ;
- простоту і зручність експлуатації системи;
- можливість підтримки системи;
- відмовостійкість системи в режимі експлуатації;
- необхідну безпеку.

Продуктивність системи - головний фактор, який визначає ефективність ПЗ [9]. Добре й продумане проектне рішення є основою високонавантаженої і високопродуктивної системи.

Проектування інформаційних систем включає в себе три основні області [9]:

- проектування структур даних і баз даних;
- проектування безпосередньо програм, різних елементів графічного інтерфейсу, звітів, які будуть забезпечувати доступ до даних;

- облік технологій і середовища: топології мережі, використання різних архітектур, конфігурації апаратних засобів, паралельної обробки даних або розподіленої обробки даних.

В реальних умовах комерційної розробки ПЗ проектування інформаційних систем є пошуком кращого способу забезпечення необхідних функціональних вимог до системи з урахуванням обмежень, які були задані замовником [9].

Методологія розробки ПЗ - це набір методів і правил, які використовуються для постановки завдання, планування, реалізації ІТ продукту [10]. Сам процес розробки описується моделлю, яка визначає послідовність етапів розробки і одержуваних результатів.

Протягом довгого часу процес розробки ПЗ здійснювався відповідно до методик, розроблених в інженерній галузі, тобто, це стандартизована практика поетапного створення ІТ продукту. Процес починався з складання формальних специфікацій і закінчувався поставкою товару замовнику. Існують стандарти, що описують даний процес: ГОСТ в Росії і ISO в Європі, CMM (Capability Maturity Model - поширений в США), які регламентують цей процес [10].

При розробці програмного продукту використовується велика кількість методологій розробки ПЗ, іншими словами, установлених та затверджених підходів і набору правил. Вибір залежить від специфіки конкретного проекту, бюджету, часових рамок, суб'єктивних переваг і навіть темпераменту і настрою керівника. Далі буде дано опис найбільш використовуваних методологій розробки в даний час [10].

Яку ж методологію вибрати? Зазвичай, коли необхідно прийняти рішення про вибір методології в голові занадто багато різнорідної інформації і важко зрозуміти, що найкраще для даного проекту [10].

В першу чергу потрібно відзначити, що не існує універсального набору умов для всіх ситуацій при виборі тієї чи іншої методології. У кожному разі ви повинні орієнтуватися на специфіку даного проекту [10].

По-друге, обравши методологію, не потрібно сліпо слідувати їй. Часто її необхідно адаптувати під свій проект [9].

Для даного проекту обираємо методологію Scrum, який включає набір методів і попередньо визначених ролей. Головні дійові особи — ScrumMaster, той хто опікується процесами, веде їх і працює як керівник проєкту, Власник Продукту, людина, що представляє інтереси кінцевих користувачів та інших зацікавлених в продукті сторін, та Команду, яка включає розробників[11].

Протягом кожного спринту, 15-30 денного періоду (тривалість визначається командою), працівники створюють функціональний ріст програмного забезпечення [11]. Набір можливостей, які імплементуються кожного спринту, приходять з етапу, що має назву product backlog (документація запитів на виконання робіт), який має найвищу пріоритетність за рівнем вимог до роботи, що повинна бути виконана. Запити на виконання робіт (backlog items), що визначені протягом наради з планування спринту (sprint planning meeting), переміщуються в етап спринту [11]. Протягом цієї наради Власник Продукту інформує про завдання, які він хоче, аби були виконані. Тоді Команда визначає, скільки з бажаного вони можуть зробити, щоб завершити необхідні частини протягом наступного спринту[11].

Протягом спринту команда виконує визначений фіксований список завдань (backlog items). Впродовж цього періоду ніхто не має права змінювати перелік запитів на виконання робіт, що слід розуміти, як заморожування вимог (requirements) протягом спринту [11].

2.2. Дослідження програмних засобів для формування послідовностей виконання замовлень

2.2.1. Програмний засіб Znedesk

Znedesk – популярний програмний засіб з доброзичливим інтерфейсом. Простий у використанні, що не вимагає спеціальних знань для впровадження. Завдяки інтуїтивності дозволяє розпочати роботу без тривалого навчання працівників. Оптимальний для невеликих компаній через доступні тарифи.

Головне вікно Znedesk представлено на рисунку 2.1.

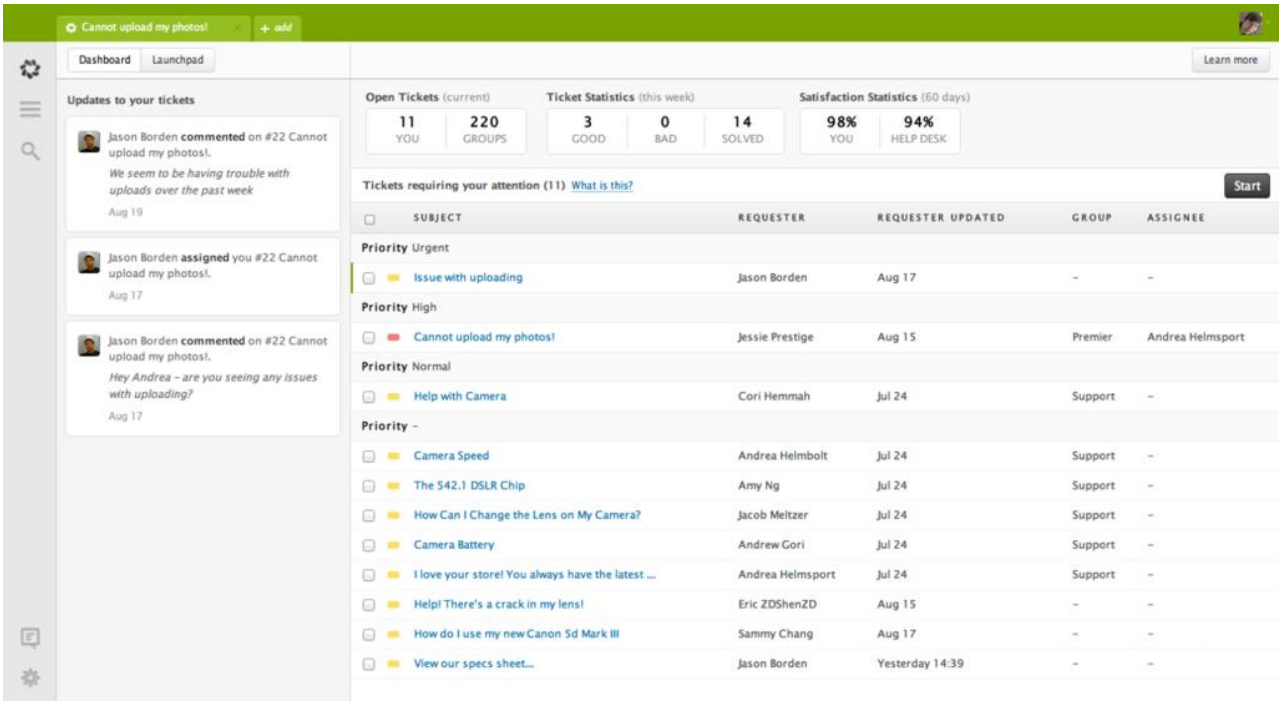

Рис. 2.1. Головне вікно системи Zendesk

Zendesk може похвалитися опрацьованим та послідовним UX (user experience). Адміністратору насамперед пропонується налаштувати систему, а співробітнику підтримки – заповнити профіль та налаштувати зручний спосіб відображення завдань. У панелі відображаються заявки, що вимагають уваги співробітника, а також замовлення, з якими співробітник взаємодіяв. Є

можливість створювати групи співробітників та окремо відображати групові задачі.

Незважаючи на видиму простоту та широке використання в середовищі малого бізнесу, Zendesk має всі можливості для організації роботи з задачами в масштабах великих корпорацій.

До переваг відносять:

- зручний інтуїтивний інтерфейс;
- найширші можливості інтеграції зі сторонніми системами та сервісами;
- можливість приймати заявки з будь-яких каналів – email, Twitter, Facebook, живий чат на сайті, а при зверненні телефоном Zendesk автоматично створює завдання;
- IP-телефонія, готова система колл-центру;
- ефективна автоматизація, що налаштовується;
- кросплатформеність, наявність програм для Android та iOS;
- наявність демократичних тарифів – $5 за одного працівника на місяць.

До недоліків відносять:

- відсутність безкоштовних версій та тарифів;
- налаштовуються правила є лише у дорожчих тарифах;
- складний інтерфейс, не завжди ефективна робота фільтрів у списку завдань.

2.2.2. Програмний засіб Okdesk

В Okdesk функціонал системи замовлень поєднаний із CRM – продукт призначений для організації підтримки клієнтів у сервісних компаніях. Okdesk має характерні можливості, необхідні для сервісних компаній – ведеться облік договорів, періодів та об'єктів обслуговування, абонентських платежів тощо.

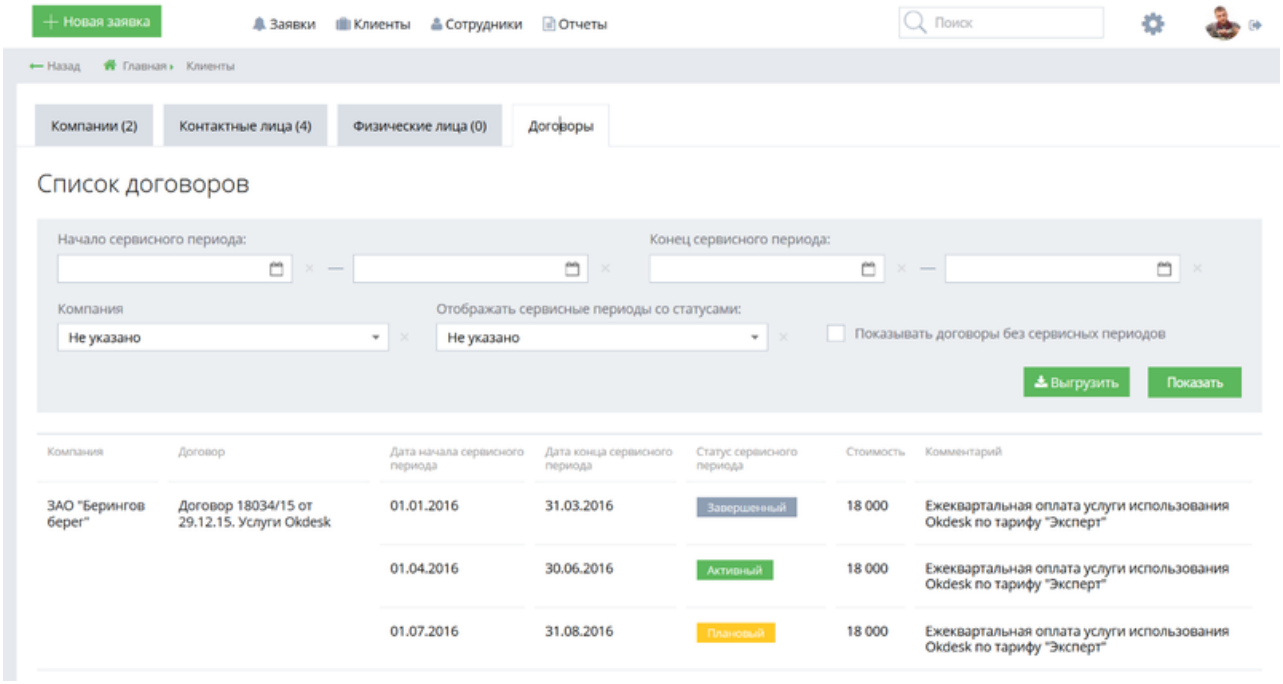

Рис. 2.2. Головне вікно системи Okdesk

Інтерфейс Okdesk простий та інтуїтивний програмний продукт, що забезпечує прийом та створення задач через електронну пошту, веб-форму, яку можна розмістити на сайті, особистий кабінет клієнта, а також через телефонні звернення. За допомогою модуля «Прайс-лист» є можливість миттєво виставляти рахунок послуг компанії з урахуванням усіх нюансів конкретного випадку.

Існує чотири тарифи з щомісячною платою від 2500 до 19000 грн, в залежності від розмірів компанії. Надається тестовий 14-денний період.

До переваг відносять:

- готове рішення з CRM для сервісних компаній;
- багатий набір функцій, у якому враховується сервісна специфіка;
- інтуїтивний інтерфейс;
- зручна робота зі списком заявок;
- API та гнучкі правила для автоматизації;
- Інтеграція з більш ніж 20 АТС;

- SMS-повідомлення;
- мобільний додаток для Android та iOS.

До недоліків відносять:

- спірний принцип фільтрації замовлень;
- налаштовуються правила, які є лише у дорожчих тарифах.

2.2.3. Програмний засіб Kayako

Система Kayako преміум-рівня, що використовується багатьма великими корпораціями (Peugeot, Toshiba). За функціональністю Kayako перевершує більшість конкурентів - у системі є все, що може знадобитися для організації служби підтримки будь-якого масштабу. У Kayako можна використовувати власні SIP-номери, завдяки чому за допомогою системи можна організувати повноцінну та всебічну підтримку клієнтів.

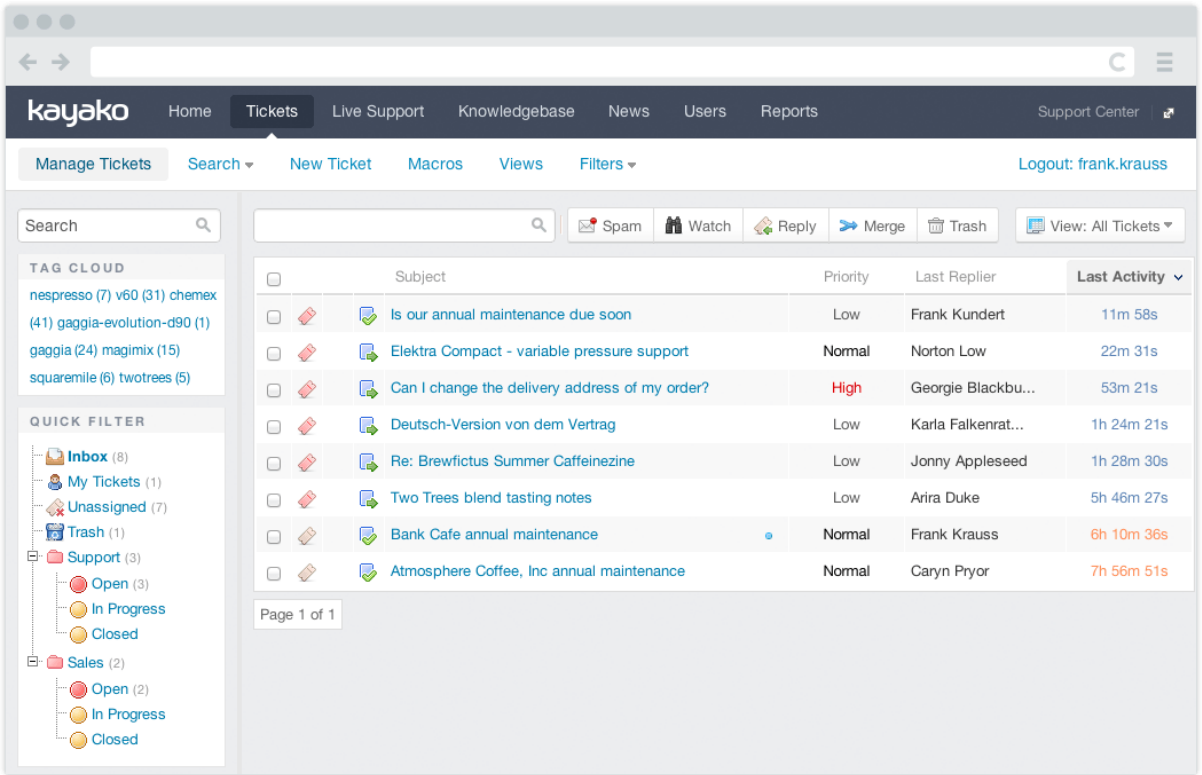

Рис. 2.3. Головне вікно системи Kayako

Через велику кількість опцій інтерфейс системи виглядає перевантаженим - спочатку після початку використання Kayako співробітникам доводиться розбиратися в принципах його роботи.

Код продукту є відкритим, до нього можна вносити зміни.

До переваг відносять:

- наявність всіх функцій, які можуть знадобитися для організації роботи з завданнями;
- можливість використання SIP-номера;
- наявність десктоп-версії для обробки дзвінків та повідомлень в live-чаті;
- SMS-повідомлення;
- створення будь-яких звітів за допомогою Kayako Query Language;
- активний розвиток системи та просунута служба підтримки.

До недоліків відносять:

- складний інтерфейс, не завжди ефективна робота фільтрів у списку завдань;
- необхідність самостійної адаптувати інтерфейс веб-системи для співробітників;
- висока вартість – від $20 доларів за працівника на місяць.

2.2.4. Програмний засіб vsDesk.

Проста і відносно молода система управління завданнями. Інтерфейс відрізняється лаконічністю - незважаючи на безліч опцій, він не виглядає перевантаженим. Завдання формуються або клієнтами через веб-додаток або співробітниками, що приймають дзвінки. При цьому є можливість створювати завдання у повній та спрощеній формі, що економить час клієнта при виникненні несуттєвої проблеми або питання.

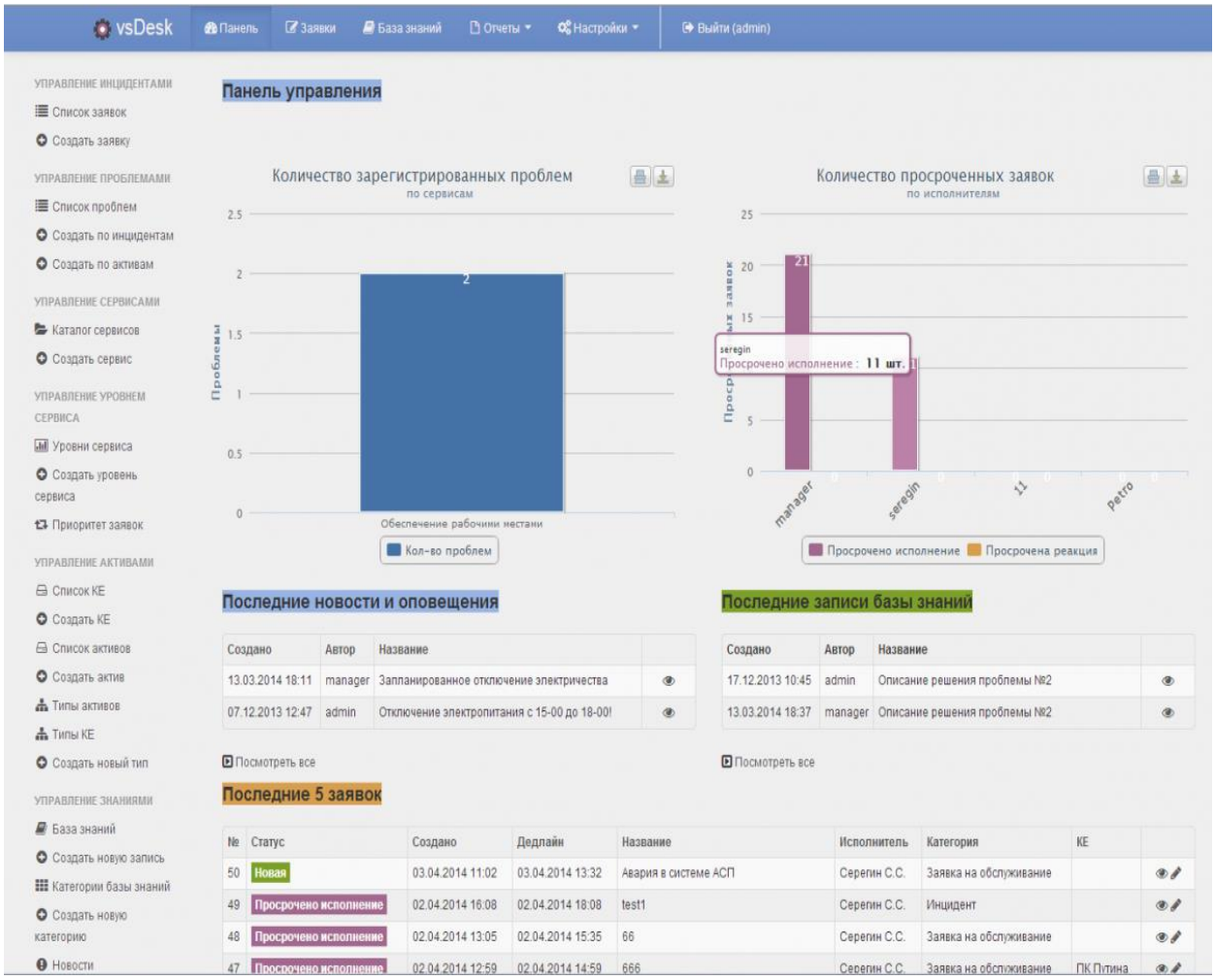

Рис. 2.4. Головне вікно системи vsDesk.

Існує три платні версії системи: «Старт», «Про» та «Корп». Відповідно вони призначені для невеликих, середніх та великих компаній. Для кожної версії встановлено разовий платіж – 30, 60 та 90 тисяч грн. Розрізняються версії з постійною технічно ю підтримкою (яка за потреби може бути продовжена) та наявністю додаткових опцій – SMS-повідомлення, внутрішній чат, звіти тощо.

До переваг відносять:

- можливість завантажити демо-версію з лімітом в 100 завдань;
- відсутність обмежень щодо кількості користувачів та співробітників;
- короткий інтуїтивний інтерфейс;

- зручний процес роботи з завданнями;
- можливість доопрацювання під час придбання версії з вихідним кодом.

До недоліків відносять:

- Не можна змінювати та гнучко налаштовувати ролі користувачів у версії «Старт».
- У процесі роботи можна зіткнутися з недоліками (які досить швидко виправляються розробниками).
- Самостійне доопрацювання можливе лише при придбанні версії «Корп» із вихідними джерелами в комплекті.

2.2.5. Програмний засіб OTRS.

Система завдань з відкритим кодом (*Open-source Ticket Request System*). Користується популярністю серед великих корпорацій (Sony, HP, Яндекс і навіть НАСА) багато в чому завдяки можливості вносити серйозні зміни в код і створювати на основі OTRS фактично власну систему завдань, адаптовану під специфічні завдання.

Інтерфейс помітно відрізняється від чисто комерційних систем. Вважається, що його можна назвати менш інтуїтивним, але зручним після належного вивчення. Список завдань відображається як черги, так і окремі з поточним статусом. Між двома варіантами відображення необхідно перемикатися. Доступно групове редагування завдань, а також блокування завдань працівником, який взяв його в роботу. При цьому можна вказати час обробки заблокованого завдання, після якого заявка знову стає доступною для інших співробітників (якщо взяв не встиг її закрити).

OTRS безкоштовна, проте є хмарні бізнес-версії, які включають підтримку фахівців.

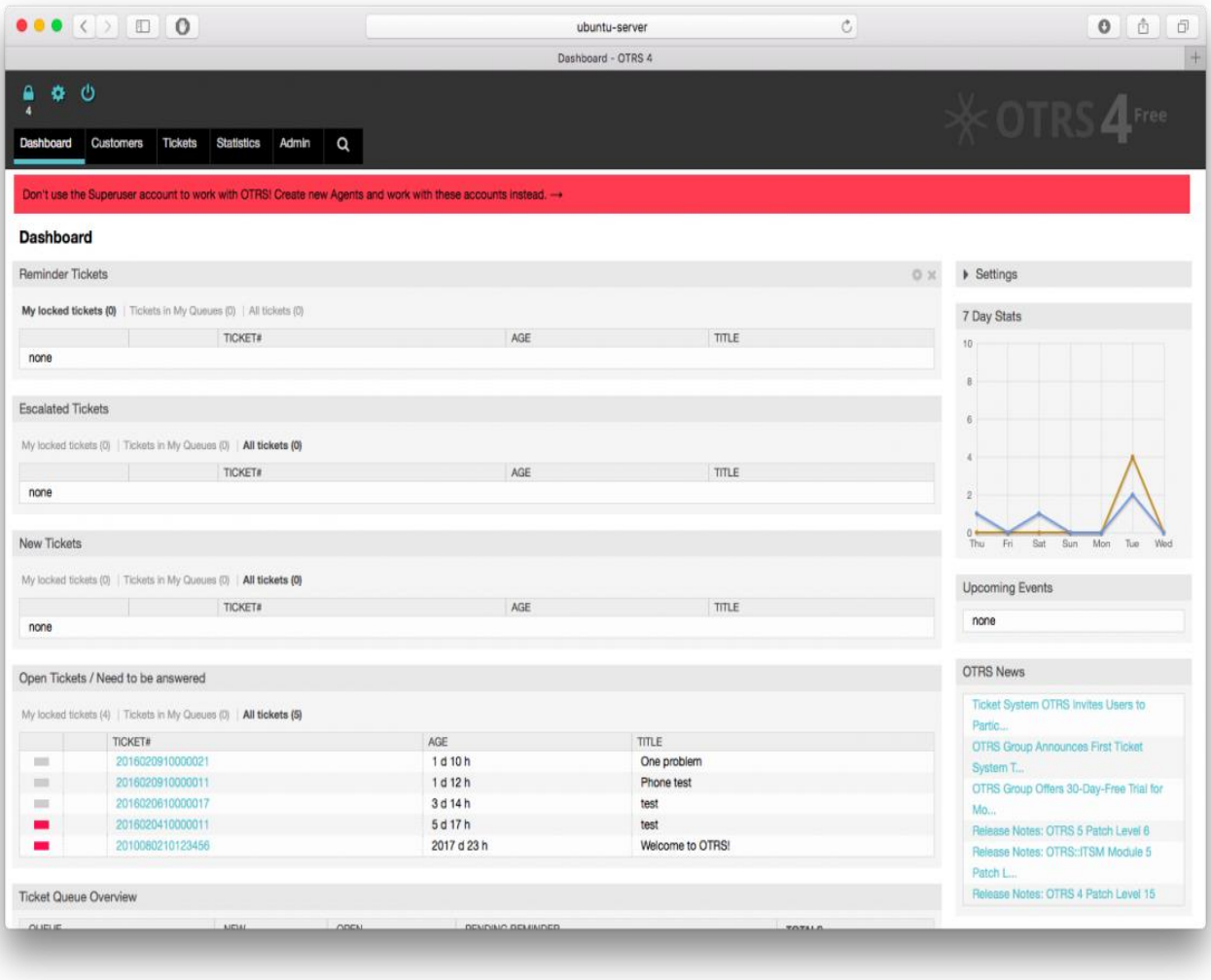

Рис. 2.5. Головне вікно системи OTRS

До переваг відносять:

- базова версія повністю безкоштовна;
- наявність версій для Windows та Linux;
- безкоштовне доповнення дозволяє керувати системою з iOS-клієнтом;
- вкрай широкі можливості з налаштування та доопрацювання системи;
- система працює з протоколом LDAP та бібліотекою ITIL;
- детальна документація, основна частина якої перекладена російською мовою.

До недоліків відносять:

- оплата за гнучкість та можливість модифікувати систему під власні завдання – складність в інтеграції, налаштуванні та обслуговуванні. Необхідно мати спеціаліста, який займатиметься системою. Альтернативний варіант – придбання платної версії з повноцінною підтримкою розробників;
- інтерфейс, до особливостей якого необхідно адаптуватись.

2.2.6. Програмний засіб osTiket.

Ще одна система з відкритим кодом, яка користується широкою популярністю завдяки простоті інтеграції та можливості швидко організувати роботу відділу підтримки. Має простий, інтуїтивний інтерфейс і багатий набір функцій. Клієнти можуть створювати завдання через відправлення на пошту, через онлайн-запит або звертаючись до служби підтримки в телефонному режимі.

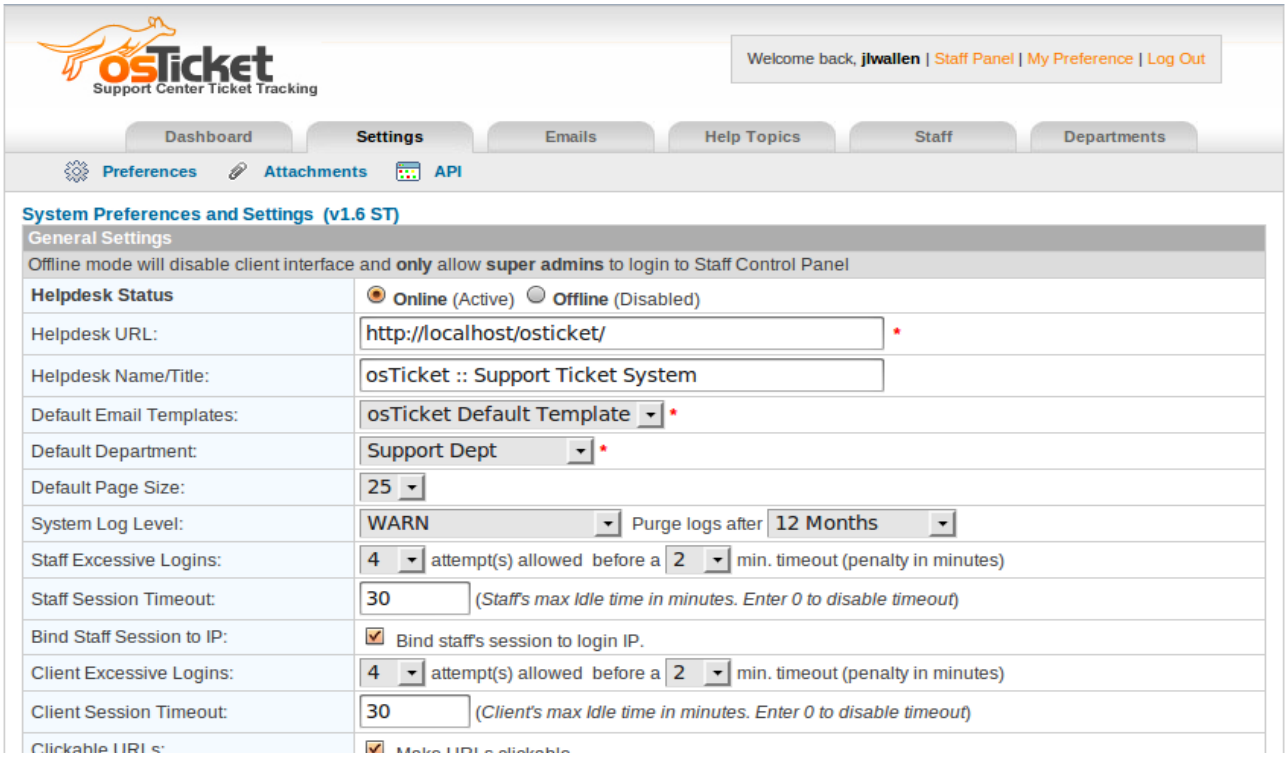

Рис. 2.6. Головне вікно системи osTiket

Завдання, що надходять до системи, зручно структуруються за темою та приналежністю до певного відділу або співробітника служби підтримки.

У процесі роботи з завданнями співробітники можуть використовувати шаблони та готові відповіді, додавати коментарі до внутрішнього користування.

Доступ до завдань може мати кілька рівнів, що є зручним при вибудовуванні ієрархії роботи із заявками.

Версія для розміщення на власному сервері безкоштовна, так само є можливість використовувати на платній основі хмарну версію із підтримкою від фахівців osTicket.

До переваг відносять:

- наявність безкоштовної версії;
- встановлення на Windows, Mac OS X та Linux;
- гнучке налаштування практичних всіх основних функцій;
- міжнародний статус продукту – підтримується багато мов;
- клієнту не потрібно проходити процес реєстрації для створення замовлення.

До недоліків відносять:

- завдання вважається закритим одразу після відповіді співробітника – без можливості для користувача будь-що уточнити чи прокоментувати.
- клієнт може бачити лише власні заявки, немає можливості вивчити чужі завдання зі схожою темою.

2.3. Використання методології яка основана на теорії модифікованого генетичного алгоритму

У випадку, коли задача має великий обсяг, доцільно використати генетичний алгоритм та його модифікації, адже вони не зводяться до безладного блукання в пошуковому просторі допустимих рішень завдяки можливості ефективного використання досвіду, набутого кожною популяцією у визначенні нової області пошуку рішень, в якій передбачається поліпшення значення цільової функції [12-14]. Механізм кожного генетичного алгоритму завжди складається із трьох основних операторів [12-14]: репродукція – процес, в якому хромосоми обираються із кращим значенням цільової функції; кроссовер – схрещування батьківських пар, генерація нащадків; мутація – дія випадкових чинників.

Задача кодується таким чином, щоб її вирішення могло бути представлено в вигляді масиву подібного до інформації складу хромосоми. Цей масив часто називають саме так «хромосома» [12]. Випадковим чином в масиві створюється деяка кількість початкових елементів «осіб», або початкова популяція. Особи оцінюються з використанням функції пристосування, в результаті якої кожній особі присвоюється певне значення пристосування, яке визначає можливість виживання особи [13]. Після цього з використанням отриманих значень пристосування обирають особи, допущені до схрещення (селекція). До осіб застосовується «генетичні оператори» (в більшості випадків це оператор схрещення (crossover) і оператор мутації (mutation)), створюючи таким чином наступне покоління осіб [14]. Особи наступного покоління також оцінюються застосуванням генетичних операторів і виконується селекція і мутація. Так моделюється еволюційний процес, що продовжується декілька життєвих циклів (поколінь), поки не буде виконано критерій зупинки алгоритму. Таким критерієм може бути [12-14]:

- знаходження глобального, або наближеного до оптимального вирішення;
- вичерпання числа поколінь, що відпущені на еволюцію;
- вичерпання часу, відпущеного на еволюцію.

Генетичні алгоритми можуть використовувати для пошуку рішень в дуже великих і важких просторах пошуку [15].

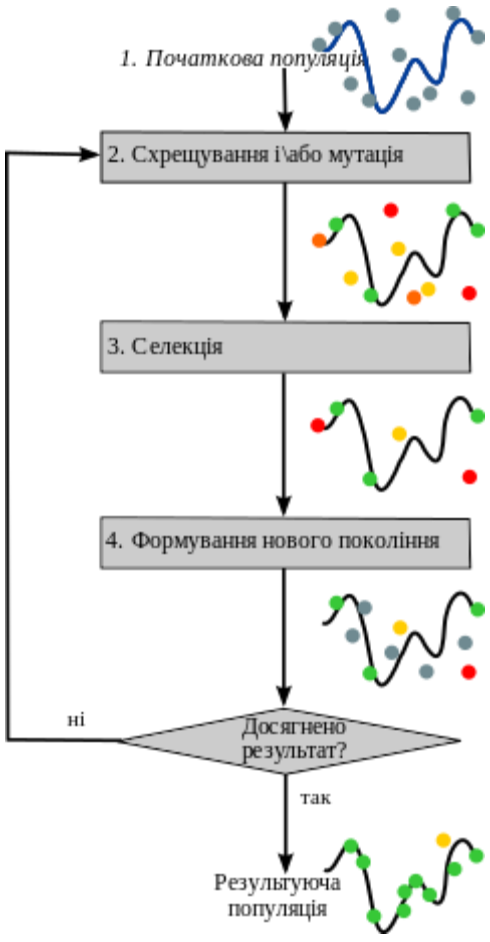

Рис 1. Схема роботи генетичного алгоритму

Етапи генетичного алгоритму наступні [15]:

- Створення початкової популяції
- Обчислення функції пристосування для осіб популяції (оцінювання)

- Повторювання до виконання критерію зупинки алгоритму
- Вибір індивідів із поточної популяції (селекція)
- Схрещення або/та мутація
- Обчислення функції пристосування для всіх осіб
- Формування нового покоління
- Створення початкової популяції

Перед першим кроком необхідно випадковим чином створити деяку початкову популяцію. Навіть якщо популяція виявиться абсолютно неконкурентоздатною, генетичний алгоритм все одно достатньо швидко переведе її в придатну для життя популяцію. Таким чином, на першому кроці можна не старатися зробити надто допасованих осіб, достатньо, щоб вони відповідали формату осіб популяції, і на них можна було порахувати функцію застосування. Наслідком першого кроку є популяція H, що налічує N осіб [16].

На етапі відбору необхідно із всієї популяції вибрати її певну долю, яка залишиться в «живих» на цьому етапі популяції. Є декілька способів провести відбір. Ймовірність виживання особи h повинна залежати від значення її застосування Fitness(h). Сама ж доля відібраних s зазвичай є параметром генетичного алгоритму, і її просто задають заздалегідь. Внаслідок відбору із N осіб популяції H повинні залишитись sN осіб, які ввійдуть в наступну популяцію H'. Решта осіб «загине» [16].

Розмноження в генетичних алгоритмах зазвичай статеве — щоб «народити» нащадку, необхідно декілька батьків, зазвичай потрібна участь двох. Розмноження в різних алгоритмах описується по різному — воно, звісно, залежить від формату осіб. Головна вимога до розмноження — щоб нащадок чи нащадки мали можливість успадкувати риси всіх батьків, «змішавши» їх якимось достатньо розумним чином [16].

Розмноження або оператор рекомбінації застосовують відразу ж після оператора відбору батьків для отримання нових особин-нащадків. Сенс рекомбінації полягає в тому, що створені нащадки повинні наслідувати генну інформацію від обох батьків. Розрізняють дискретну рекомбінацію і кросинговер.

Приклад операції розмноження: Вибрати (1-s)p/2 пар гіпотез із H і провести з ними розмноження, отримавши по два нащадки від кожної пари (якщо розмноження описано так, щоб давати одного нащадка, необхідно вибрати (1 — s)p пар), і додати цих нащадків в H'. В результаті H' буде складатися з N осіб [16].

Особи для розмноження зазвичай вибираються із всієї популяції H, а не із тих, що вижили на першому кроці (хоча останній варіант теж має право на існування). Справа в тому, що головна проблема генетичних алгоритмів — недостача різноманітності (diversity) в особах [16]. Достатньо швидко виділяється єдиний генотип, який являє собою локальний максимум і згодом всі елементи популяції програють йому в відборі, і вся популяція «забивається» копіями цієї особи. Існують різні способи боротьби із таким небажаним ефектом; один з них — вибір для розмноження не з самих «допасованих», а взагалі зі всіх осіб [16].

До мутацій відноситься все те ж, що і до розмноження: є деяка доля мутантів m, що є параметром генетичного алгоритму, і на кроці мутацій необхідно вибрати mN осіб, а згодом змінити їх згідно з заздалегідь заданими операціями мутації [16].

Висновок до розділу 2

Вибір певної системи управління замовлень цілком залежить від специфіки поставленої задачі. Вимоги, що пред'являються до систем, можуть значно відрізнятися в залежності від розмірів компанії та характеру послуг. Більшість

систем, представлені на сучасному ринку, активно розвиваються та здатні задовольнити запити будь-якої компанії. Але головною проблемою при використанні готових систем є ціна та межі функціоналу які встановлюються виробником. Крім відсутність оцінки набраних завдань для конкретного виконавця. Тому виникає доцільність розробки власного системи з можливістю обрання завдання для конкретного виконавця, з чітко визначеним функціоналом за який не доведеться переплачувати і можливість повного адміністрування створеного продукту і можливість його модернізації власними силами.

РОЗДІЛ 3. ПРОЦЕС РОЗРОБКИ ПРОЕКТУ «СТВОРЕННЯ СИСТЕМИ ПІДТРИМКИ ОБРАННЯ ЗАМОВЛЕНЬ НА ВИКОНАННЯ ПОСЛУГ СИСТЕМНОГО АДМІНІСТРАТОР КОМП'ЮТЕРНИХ МЕРЕЖ»

3.1 Обрання та обґрунтування засобів розробки інформаційної системи

3.1.1 Мова програмування Pyton

Python надзвичайно легкий і простий у навчанні. Він дуже нагадує англійську мову, тому входить до однієї з найбільших його переваг. Це дуже потужна мова, і є безкоштовною. Це мова високого рівня, і ми можемо просто писати код англійською мовою, а python перетворює його на мову низького рівня. Він також інтерпретує мову, якою машина читає та інтерпретує код, який несе всі помилки. Якщо у користувача є якісь сумніви, вони легко вирішують це, і спільнота python прогресивно працює разом, щоб покращити основні функціональні можливості пітона. Крім того, спільнота python постійно забезпечує вдосконалення та оновлення з плином часу.

Оскільки python є портативним, його підтримують усі платформи таких галузей, як Windows, Linux, Macintosh, а також ігрові станції. Завдяки розширюваності python ми можемо повністю інтегрувати java, а також точкові компоненти мережі, навіть ми можемо викликати бібліотеки C і C ++, і це всі переваги python. Крім того, ми можемо виконувати міжмовні операції з python, тому python надзвичайно розширюється з більшістю мов програмування.

Python підтримує орієнтоване програмування, воно дозволяє поліморфізм та успадкування. Користувачі Python отримують можливість користуватися спільними категоріями, таким чином код може бути багаторазовим і додатково забезпечувати механізм захисту шляхом абстрагування знань. Крім того, він звик

розробляти прототипи, які модифікують користувача комп'ютера для прямого сканування та запису.

Штучний інтелект означає, що машинна програма, яка діє або відповідає як інтелект людського мозку, це робиться за допомогою багатьох алгоритмів або програм. Він поєднується з такими бібліотеками, як scikit-learn python, має можливість робити складний обчислення лише за допомогою одного твердження, і, крім того, такі бібліотеки, як Keras і tensorflow ping-функція машинного навчання, поєднуються в суміш. У Python також є такі бібліотеки, як відкрите резюме, яке допомагає розпізнавати зображення, такі як комп'ютерний зір, і ще одна особливість python полягає в тому, що ми можемо виявити розпізнавання обличчя чи мови. Штучний інтелект - це більш широка концепція машин, здатних виконувати завдання таким чином, які ми вважаємо розумними, а отже, ми також можемо підвищити загальну продуктивність компанії, і все це є пізнаваними рисами python.

Python в основному використовується в невеликих і великих онлайн-або офлайн-проектах, і він використовується для побудови графічного інтерфейсу, який розшифровується як графічний інтерфейс користувача, він також використовується для настільних додатків, тоді у нас є розробка ігор, тому Tkinter є стандартною бібліотекою графічного інтерфейсу для python, тому python у поєднанні Завдяки Tkinter це швидкий та простий спосіб створити графічний додаток та програми. Python також пропонує спеціальну основу для розробки ігор, яка є PYGAME.

Python приємний для хороших ідей або продукту для запуску в компанії, яка зараз робить код - це не просте завдання, оскільки містить декілька методів, починаючи зі свого стилю, його код додатково перевірені випадки та тестування коду, безперечно, найважливіший життєвий процес життя коду цикл, це також складне завдання для тестера коду, щоб розібратися на найпростішій мові програмування для тестування автоматизації, а також python полягає в тому, що

найкраще дозвіл для цього недоліку, таким чином, це кілька інтегрувати рамки тестування, що охоплюють налагодження та швидкі робочі процеси в даний час в області блоку кілька інструментів і модулі для створення предметів набагато простіше, як хімічний елемент, який є стилем в інструменті автоматизації, і тому ми отримали осколок, тому він підтримує тестування з кросплатформою та крос-браузером з рамками, на кшталт кадрів Pytest та Генрі М. Роберт В даний час це кілька приємних підтримок тестування коду.

Python може обробляти велику кількість даних. Він підтримує паралельні обчислення або метод квадратної міри, готовий використовувати Python для цього, тому в python у нас в бібліотеці говориться як PYDOOP, де ми будемо квадратно вимірювати, щоб написати програму MapReduce в python та інформацію про техніку, яка подарується з інтервалом в кластері HDFS. багато бібліотек люблять час доби та PySpark для гігантських процесів.

Він використовується як сценарій мови, тому при написанні коду пишеться в межах сценаріїв і стає мертвим, тому машина в першу чергу запускається та інтерпретує код, тоді як усі ваші перевірки помилок закінчуються під час виконання пропускної здатності, і таким чином за допомогою автоматизації ми можемо автоматизувати певну задачу в Програму ми можемо фактично відкрити браузер і розмістити вміст на динамічних веб-сайтах.

Python - це провідна мова для декількох науковців з інформатики, що існують протягом багатьох років. Студенти академії та дослідники галузі PY експлуатували мову MATLAB для дослідницького проекту, в даний час кожен з них почав модифікацію з розрядкою боротьби чисельних двигунів, таких як NumPy і Pandas python, додатково розглядає табличну матрицю так само, як застосовується математична інформація, і вона додатково візуалізує це разом із загальними бібліотеками, такими як matplotlib та Seabourn, тому, якщо ви отримаєте великий обсяг інформації, це може бути T або блок пам'яті комп'ютера, і тому пітфон експлуатації ви зможете просто отримати уявлення про нього.

Python стає дуже популярним в останні два роки, і програмісти python отримують найвищу зарплату. Google, який насправді використовує python для веб-пошуку, тоді у нас є YouTube, який багато в чому написаний на python, і багато інших компаній, таких як Instagram, Drobox, Facebook та багато інших компаній використовують Python. Це дозволяє високопродуктивне середовище, ніж інша мова на зразок C ++ та Java, тому пітон є першим вибором для досвідчених програмістів [17].

3.1.2. Django

Django – це фреймворк допоможе розробити CRUD додаток під ключ. З Django не доведеться винаходити велосипед. Він працює з коробки і дозволить зосередитися на бізнес-логіці і продуктах для звичайних людей.

Принцип «Все включено» («Batteries included») означає, що більшість інструментів для створення програми – частина фреймворка, а не поставляються у вигляді окремих бібліотек.

Django містить величезну кількість функціональності для вирішення більшості завдань веб-розробки. Ось деякі з високорівневих можливостей Django, які вам доведеться шукати окремо, якщо ви віддасте перевагу мікро-фреймворку:

- ORM
- Міграції бази даних
- Аутентифікація користувача
- Панель адміністратора
- Форми

Django як фреймворк задає структуру проекту. Вона допомагає розробникам розуміти, де і як додавати нову функціональність.

Завдяки однакової для всіх проектів структурі набагато простіше знайти вже готові рішення або отримати допомогу від спільноти. Величезна кількість захоплених розробників допоможе впоратися з будь-яким завданням набагато швидше.

Додатки в Django дозволяють розділити проект на декілька частин. Додатки встановлюються шляхом додавання в settings.INSTALLED_APPS. Цей підхід дозволяє легко інтегрувати готові рішення.

Сотні універсальних модулів і додатків дуже сильно прискорять розробку. Погляньте на їх список на сайті djangopackages.org.

Django безпечний з коробки і включає механізми запобігання поширених атак на зразок SQL-ін'єкцій (XSS) і підробки міжсайтових запитів (CSRF).

Django REST Framework, який часто скорочують до «DRF», є бібліотекою для побудови API. Він має модульну архітектуру, що настроюється, яка добре працює для створення як простих, так і складних API.

У DRF політики аутентифікації і дозволів доступні з коробки. Він поставляється з базовими класами для CRUD операцій і вбудованою утилітою для тестування розробляється API.

Великі REST API часто вимагають великої кількості запитів для отримання всіх необхідних даних. GraphQL – це мова запитів, яка дозволяє обмінюватися пов'язаними даними набагато простіше. Детальніше почитати про основні концепції GraphQL можна в офіційній документації.

Graphene-Django дозволить легко додати відповідну функціональність в ваш проект. Моделі, форми, аутентифікація, політики дозволів і інші функціональні можливості Django можна використовувати для створення GraphQL API. Бібліотека так само поставляється з утилітою для тестування результату [18].

3.1.3. CSS

Кожен веб-сайт має вміст і має заздалегідь задану структуру чи макет. Цей макет використовується для представлення вмісту. Це головна мета CSS; він відокремлює структуру документа від представлення документа. HTML - це все про структуру і нічого іншого, оскільки більшість із нас знають про це основне визначення HTML. За допомогою CSS користувач може вносити швидкі зміни за потреби, але це непросте завдання з HTML. Раніше колір тла моєї веб-сторінки був червоним (код буде # ff000). Якщо розробник хоче змінити будь-який колір дизайну, він просто повинен виправити єдиний рядок коду, який відображатиметься в будь-якій кількості веб-сторінок. Він (код #eeeeee для світло-сірого) змінить це, і його робота зроблена [19].

Менше використання кодування підвищить ефективність сторінки та скоротить час завантаження. Чим більше часу завантаження, тим більше розчарування користувачів. Що насправді відбувається в кодуванні, цікаво знати і як CSS є рятівником. Зображення поля таблиці вбудовано в код; це робить читання коду двічі. По-перше, щоб зрозуміти основну структуру таблиці, а другий раз фактично відобразити вміст у таблиці. Ця подвійна робота може уповільнити завантаження веб-сайту. Тепер переваги CSS, хоча зображення завжди більше, ніж найбільший код, який використовується або обговорюється вище, оскільки ми використовували CSS в окремому файлі, вбудований у формат CSS код буде кешований після першого початкового запиту, і немає необхідності. щоб знову завантажити сторінку. Це одна з важливих поведінки CSS над HTML.

CSS пропонує своїм користувачам, вносячи невеликі зміни на ваш веб-сайт, такі ж зміни відображаються і в інших частинах веб-сайту. Чим більший вміст вашого веб-сайту та макетів, тим більше часу заощаджує CSS для змін. Він також перевіряє та забезпечує кожну сторінку послідовною, що також є основними перевагами CSS[20].

Ваша присутність на цифровому ринку ака Інтернет є дуже важливою формою бізнес-точки зору. CSS вважається однією з найбезпечніших методологій кодування веб-розробок, це також означає, що ваш веб-сайт має більше вмісту, ніж кодів. Однією з переваг CSS є те, що CSS дуже корисний для цілей цифрового маркетингу, коли кожна просування вашої веб-сторінки дає шанс забити бізнес. Отже, більше вмісту, ніж формула коду за допомогою та використання методів CSS, краще для цифрового маркетингу та вашого бізнесу.

Доступність - один із найважливіших факторів розгляду переваг CSS над іншими методами. В Індії не так сильно наголошується на веб-сайтах з точки зору стандартів доступності (норми W3C). W3C працює на принципі, щоб зробити кожен веб-сайт однаково доступним для інвалідів, як те, як звичайні користувачі ними користуються. По-перше, CSS надає перевагу доступності, створюючи структуру та документ, що полегшує читачам екрану чітко читати вміст. Це корисно для глухих користувачів, оскільки вони покладаються на екран зчитування екрана для використання веб-додатків.

Сказане - лише вершина айсберга, оскільки історія нескінченна. Зона доступності дуже обширна і використовує CSS повною мірою. Крім вищезгаданих фактів, CSS може бути корисним для lоступності багатьма способами.

Пробіл, вирівнювання та позиціонування: означає контроль над CSS та його вищою версією надзвичайний над візуальними місцями розташування, такими як поля, поплавці, відступ тексту. Це збільшує кількість зображень і корисно для користувачів із слабким зором.

Користувачі перевизначать стилі формату

Створений вміст - це підказки, такі як смуги прокрутки, кадри та зміст[21]

3.2 Розрахунок техніко-економічного ефекту для проекту виконаного в магістерській роботі

Після завершення проекту було проведено аналіз розробки і її ефективності в цілому.

Економічний ефект – відображає результат реалізації економічних програм та заходів, що характеризується відношенням отриманого економічного ефекту (результату) до сумарних витрат для отримання цього результату.

Техніко-економічний ефект від впровадження створюваної системи підтримки обрання замовлень на виконання послуг системного адміністратор комп'ютерних мереж визначається за співвідношенням витрат та розробку системи.

3.2.1 Вихідні дані для розрахунку

Система підтримки обрання замовлень на виконання послуг системного адміністратора комп'ютерних мереж. Узагальнені дані вхідної та вихідної інформації для системи підтримки обрання замовлень на виконання послуг системного адміністратора комп'ютерних мереж за видами вхідної та вихідної інформації таблиця 3.1

Таблиця 3.1. *Узагальнені дані для вхідної та вихідної ІС.*

Вид інформації	Позначення	К-сть наборів даних
Змінна інформація	ЗІ	m=4
Нормативно-довідкова інформація	НДІ	n=3
База даних	БД	p=1
Обробка в режимі реального часу	РЧ	Так
Забезпечення комунікації між клієнтом та сервером	ТОУ	ні

Витрати часу на систему підтримки обрання замовлень на виконання послуг системного адміністратора комп'ютерних мереж, а саме на розробку ескізного проекту Т1 і технічного завдання Т2, будуть наступні (табл. 3.2.).

Таблиця 3.2. *Визначення витрат часу*

Вид системи	Стадія розробки системи	
	Передпроектне дослідження	Беклог (SCRUM)
	B	B
Система підтримки обрання замовлень на виконання послуг системного адміністратора комп'ютерних мереж	T_1= 53	T_2=42

Визначається базове значення витрат часу для стадій планування проекту розробки і впровадження системи. Для цього використовуються дані для системи підтримки обрання замовлень на виконання послуг системного адміністратора комп'ютерних мереж.

Вхідними даними для визначення є:

- кількість форм вхідної інформації B1 = 2,
- кількість форм вихідної інформації B2 = 2,
- базове значення витрат часу для стадій "Планування спринту": 50;
- базове значення витрат часу для стадій "Спринти": 81;
- базове значення витрат часу для стадій "Впровадження": 31.

Базове значення витрат часу ТБ коригується за допомогою поправочних коефіцієнтів для всіх стадій розробки автоматизованої системи.

Розрахунок витрат часу для стадії планування проекту (ТЗ).

Коефіцієнт трудомісткості робіт kп визначається

$$k_{П} = \frac{k_1 * m + k_2 * n + k_3 * p}{m + n + p}$$

$$k_{П} = \frac{1.0 * 4 + 0.72 * 3 + 2.08 * 1}{4 + 3 + 1} = 0{,}824$$

і з врахуванням коефіцієнтів табл. 3.3.:

Таблиця 3.3. *Коефіцієнти k1, k2, k3 для стадії планування проекту.*

Вид використаної інформації	Ступінь новизни
	В
k_1 (ЗІ)	1.0
k_2 (НДІ)	0.72
k_3 (БД)	2.08

Коефіцієнт ступеню новизни даного проекту, kO, що враховує вид обробки інформації для трьох стадій розробки системи наведено в табл. 3.4.

Таблиця 3.4. *Коефіцієнт ступеню новизни проекту, kO.*

Стадія розробки системи	Вид обробки	Ступінь новизни
		В
Планування системи	РЧ	1.26
Розрока системи	РЧ	1.32
Впровадження системи	РЧ	1.21

Витрати часу для стадії планування проекту Т3 розраховуються за формулою:

$$T_3 = \mathrm{T_{Б3}} * \mathrm{k_П} * \mathrm{k_О} = 50*0,98*1,26 = 61.74$$

Розрахунок витрат часу для стадії розробка проекту (Т4) і впровадження системи (Т5).

Для визначення витрат часу на стадії розробка проекту використовують формулу:

$$\mathrm{T_4} = T_{Б4} * k_П * k_О * k_С$$

де kП – коефіцієнт, що враховує вид використаної інформації і визначається за формулою:

$$\mathrm{k_П} = \frac{1.1*4 + 0.58*3 + 0.48*1}{4+3+1} = 0,827$$

Таблиця 3.5. *Коефіцієнти k1, k2, k3 для стадії розробка проекту.*

Вид використаної інформації	Група складності алгоритму	Ступінь новизни
		В
k_1 (ЗІ)	2	1.1
k_2 (НДІ)	2	0.58
k_3 (БД)	2	0.48

Коефіцієнт, що враховує вид обробки інформації на стадії розробка проекту

$$\mathrm{k_С} = 1.16$$

Витрати часу Т4 вимірюються в робочих днях працівника, розраховується:

$$Т_4 = T_{Б4} * k_П * k_О * k_С = 81*0,827*1,21*1,16 = 94.02$$

Для визначення загальних витрат часу на стадії впровадження проекту Т5 (робочих днях працівника) використовують формулу:

$$Т_5 = T_{Б5} * k_П * k_О * k_С = 31*0,827*1,21*1,16 = 35.9$$

Таким чином, загальні витрати ресурсів людських зусиль на проектування системи складають:

$$Т_Σ = 53+42+123,5+94.02+35.9 = 348$$ (робочі дні працівника).

Для магістерського проекту кількість робочих годин складає 348 із 7-годинним робочим днем, тому на розробку проекту виділено Ф, днів:

$$Ф = \frac{348}{7} = 50$$

Для магістерського проекту Ф = 50 днів. Тоді визначаємо кількість місяців із розрахунку 25 робочих днів.

Кількість місяців на розробку, М:

$$М = \frac{Ф}{25} = \frac{50}{25} = 2$$

Отже, для виконання такого проекту потрібно таку чисельність виконавців Ч, виконавців, обраховується за формулою: $Ч = \frac{Т_Σ}{Ф} = \frac{348}{75} \approx 17.4$

Якщо прийняти, що оплата програміста здійснюється в розмірі 15000 грн, то оплата праці всіх виконавців складе:

$$V'_1 = × * М * ÇЇ_{їЬ} = 7*2*15000 = 21000$$ грн.

3.2.2 Витрати, пов’язані з розробкою програми на ПК

Розрахунок річного фонду часу роботи ПК.

Дійсний річний фонд часу ПК у годинах дорівнює числу робочих годин, які провів за технікою співробітник, час на технічне обслуговування та ремонт

техніки не враховується (в середньому 5год/міс + 6 роб. днів/рік).

$$T_{ПК} = 2000 - (6*8 + 5*12) = 1892$$ год.

Оскільки під час виконання магістерського проекту студент в середньому зазвичай витрачає 450 год. Часу роботи нам ним, то величина фонду часу ПК дорівнює

$$T'_{ПК} = 1892 * \frac{450}{2000} = 425.7$$ год

Поточні витрати на експлуатацію V1" .

Балансова вартість ПК вираховується

$$Ц_{ПК} = Ц_{Р} * (1 + k_{УН}) = 8000 * (1 + 0.12) = 8960$$ грн

де ЦР – ринкова вартість ПК, орієнтовно складає 8000 грн.,

kУН – коефіцієнт, який враховує витрати на установку і налаштування ПК і дорівнює 0.12.

Амортизаційні відрахування використання ПК, ЗАМ, обчислюються

$$З_{АМ} = \frac{8960}{5} = 1792$$ грн.

норма амортизаційних відрахувань для дорівнює НА = 5:

Витрати на електроенергію яку використовує робоча техніка

$$З_{ЕЛ} = Р_{ПК} * Т'_{ПК} * Ц_{ЕЛ} * А = 1.4 * 425.7 * 0.74 * 0.9 = 397$$ грн

де потужність ПК, РПК = 1.4 кВт,

фонд корисного часу роботи ПК, ТПК = 425.7 год,

вартість 1 кВт електроенергії для підприємств, ЦЕЛ = 1.74 грн/кВт,

коефіцієнт інтенсивного використання ПК, А = 0.9.

ЗР – витрати на ремонт і технічне обслуговування ПК визначаються як 6% від балансової вартості ПК, ЦПК.

$$З_{Р} = Ц_{ПК} * 0.06 = 8960 * 0.06 = 537.6$$ грн.

ЗМАТ – непрямі витрати, пов’язані з експлуатацією робочої техніки,

визначаються як 5% від балансової вартості ПК ЦПК.

$$З_{МАТ} = Ц_{ПК} * 0.05 = 8960 * 0.05 = 448 \text{ грн.}$$

Таким чином, отримаємо:

заробітна плата персоналу (якщо роботи виконуються не на власному ПК); ЗОП = 1680 грн, ЗАМ = 896 грн, ЗЕЛ = 113.41 грн,

Поточні витрати на експлуатацію V1", грн, рахуються:

$$V_1" = З_{ОП} + З_{АМ} + З_{ЕЛ} + З_{Р} + З_{МАТ} = 1680 + 896 + 113.41 + 537 + 448 = 3674.41$$

Отримуємо, що загальні витрати на розробку системи розраховуються наступною формулою і складуть:

$$V_1 = V'_1 + V_1" = 84000 + 3674.41 = 87674 \text{ грн.}$$

Витрати на закупівлю і налаштування ПК V2.

На підприємстві є ПК, на якому буде працюватиме конкретна система, тому V2=0.

Витрати на підготовку приміщення V3.

Дані витрати залежать від стану приміщення, де буде встановлений ПК. Так як необхідне приміщення є, тому:

$$V_3 = 0 \text{ грн.}$$

Витрати на навчання персоналу V4.

Зазвичай навчання персоналу триватиме 1 місяць, тому можна вважати, що:

$$V_3 = 2000 \text{ грн.}$$

3.2.3 Загальна вартість розробки і впровадження системи

Загальна вартість розробки і впровадження розробленої системи V∑, вираховується за наступною формулою:

$$V_Σ = V_1 + V_2 + V_3 + V_4 = 87674 + 0 + 0 + 2000 = 89674 \text{ грн.}$$

Оскільки норма амортизаційних втрат для комп’ютерних систем НА = 5, то

для обрахування річного економічного ефекту слід брати до уваги величину, формулою

$$V_Р = \frac{V_\Sigma}{H_А} = \frac{89674}{5} = 17934,8$$ грн.

Річний прибуток ПР від впровадження системи буде досягнуто за рахунок впровадження додаткової інформаційної системи на прикладі різниці as-is і to-be і складає 20000 та 12500 грн за тиждень. Коефіцієнт економічної ефективності розробки вираховується:

Прибуток від впровадження ІС за рік:

54* 7500 = 405.000грн

$$К_{ЕФ} = \frac{П_Р}{V_Р} = \frac{12500}{17934.8} = 0,696$$

Термін окупності розробки визначається за формулою[22].

$$T_{ОК} = \frac{1}{К_{ЕФ}} = \frac{1}{0,696} = 1,43$$

Таким чином, термін окупності інформаційної системи буде півтора роки.

3.3. Створення інформаційної системи підтримки обрання замовлень на виконання послуг системного адміністратор комп'ютерних мереж

3.3.1. Створення концепції продукту

Концепція продукту складається із усіх визначених задач та проблем на момент часу. Створюється мною, на основі вимог та побажань до створюваного додатку.

У нашій компанії велика кілька відділів, кожен відділ має свою специфіку. Заявки можуть переміщуватися між відділами у процесі роботи. Крім того, у нас

є можливість встановити продукт на власному сервері, і ми зацікавлені в його тонкому підстроюванні під себе — розробники для подібних завдань у нашій компанії також є. Виходячи з цих передумов, було складено перелік вимог.

Вимоги:

- Активність розробників та спільноти – продукт має розвиватися та підтримуватися.
- Відносно не велика ціна готового продукту
- Можливість встановлення на web хостингу
- Реалізація власними силами внутрішніх працівників – для можливості модифікації системи в зручний для себе час.
- Вихідний код має бути зрозумілим та структурованим.
- Продукт використовує бази даних SQLite.
- Наявність веб-інтерфейсу для роботи із заявками.
- Можливість бачити який саме співробітник відповідав на заявку.
- Підтримка кількох відділів. Листи автоматично потрапляють до потрібного відділу.
- Можливість залишати внутрішні коментарі у завданнях.
- Наявність механізму ескалації завдання (якщо на звернення довго не відповідають, про це одразу дізнається менеджер).
- Можливість зворотного зв'язку: оцінка користувачем відповіді на співробітника підтримки [23].

3.3.2. Проектування бази даних

Проектування бази даних починається з формування критеріїв до функціоналу майбутнього додатку.

Основна таблиця отримала назву «ticket». Вона буде містити інформацію

про створення користувачем, оператором тікети. Сутність буде мати наступні поля:

- «title»
- «full_text»
- «customer»
- «status»
- «date»

Як базу даних було обрано «SQLite»

Код створення сутнотсі описаний наступним фрагментом коду, який відображений в Додатку 1.

3.3.3. Створення елементів системи

Для створення додатку було обрано мову програмування Python. Для зручності буду використовувати фреймворк Django.

Етапи створення:

- Опис бази даних
- Основні налаштування Додаток 2
- Створення HTML шаблонів
- Базова HTML сторінка Додаток 3
- Створення форм Додаток 4
- Представлення Додаток 5
- Логіка роботи додатку Додаток 6

Створюючи HTML, використовуючи шаблон, є ризик, що змінна може містити символи, які вплинуть на структуру отриманого HTML.

Очевидно, дані користувача не можна сліпо довіряти і вставляти безпосередньо у вміст сторінки, оскільки зловмисники можуть використовувати

це з поганими намірами. Такий тип уразливості називається Cross Site Scripting (XSS) атакою.

Щоб уникнути цієї проблеми, у вас є два варіанти:

1) можете застосовувати до всіх сумнівних змінних фільтр escape, який перетворює потенційно небезпечні HTML символи на безпечні. Таке рішення було прийнято в перших версіях Django, але проблема в тому, що воно покладає тягар відповідальності за безпеку на вас, розробника / автора шаблону. Легко забути екранувати змінну.

2) Друге, ми дописуємо бібліотеку, щоб відбувалось автоматичне екранування змінних.

Зокрема, виконуються такі заміни:

< замінюється на <

> замінюється на >

' (одинарна лапка) замінюється на '

" (подвійна лапка) замінюється на "

& замінюється на &amp.

3.3.4. Результат виконаної роботи. Перша версія готового продукту

Після успішного розміщення додатку на web хостингу ми отримали наступний результат.

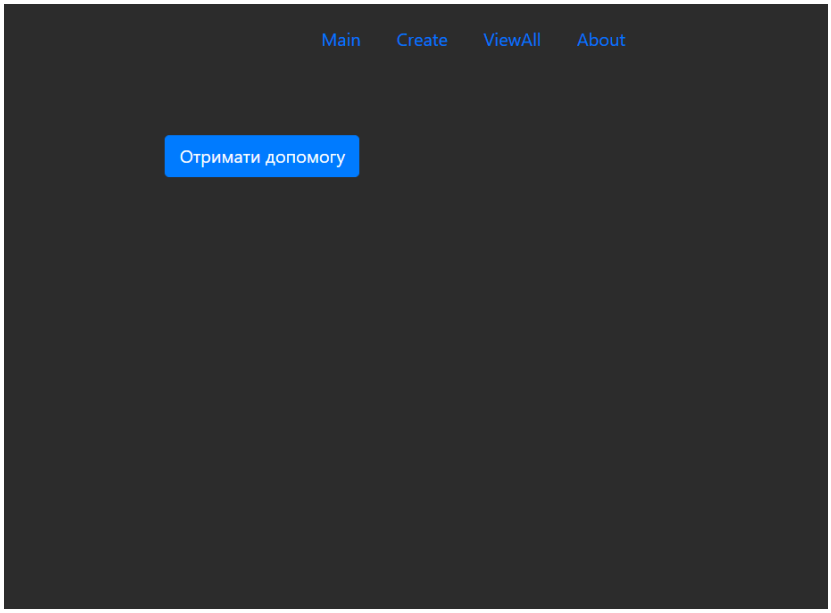

Рис. 3.1. вікно головної сторінка

На малюнку вище відображений вигляд головної сторінки яка містить кнопку «Отримати допомогу» при натисканні якої користувач зможе оформити заявку на отримання допомогу від кваліфікованого співробітника, також сторінка містить навігацію по сайту.

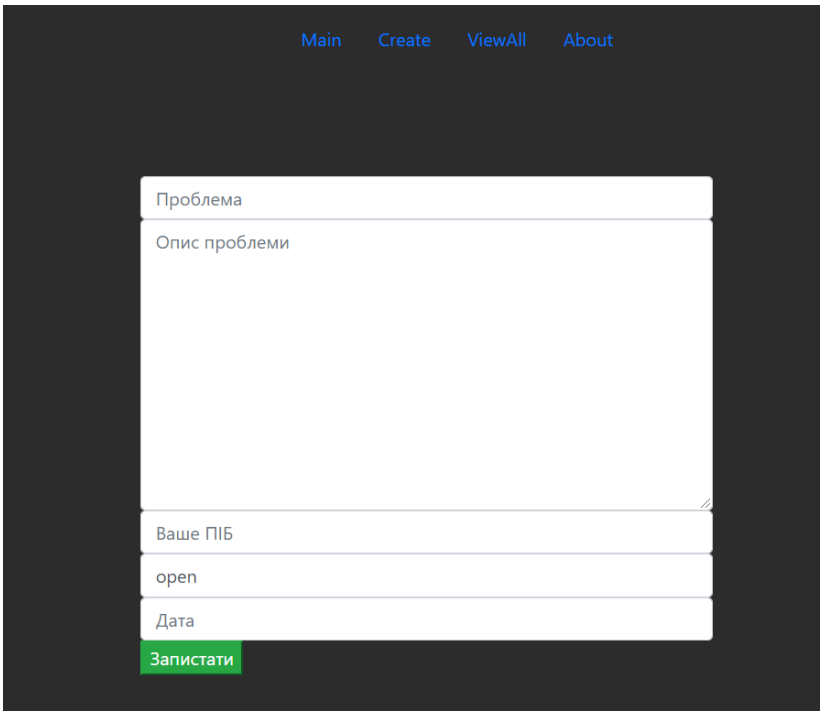

Рис. 3.2. Вікно створення запиту

На малюнку вище відображений вигляд процесу створення запиту, дана сторінка містить форму для створення запиту. в якій описується проблема, опис даної проблеми, ПІБ власника запиту та дата, у відповідних полях «Проблема», «Опис проблеми», «Ваше ПІБ», «Дата»

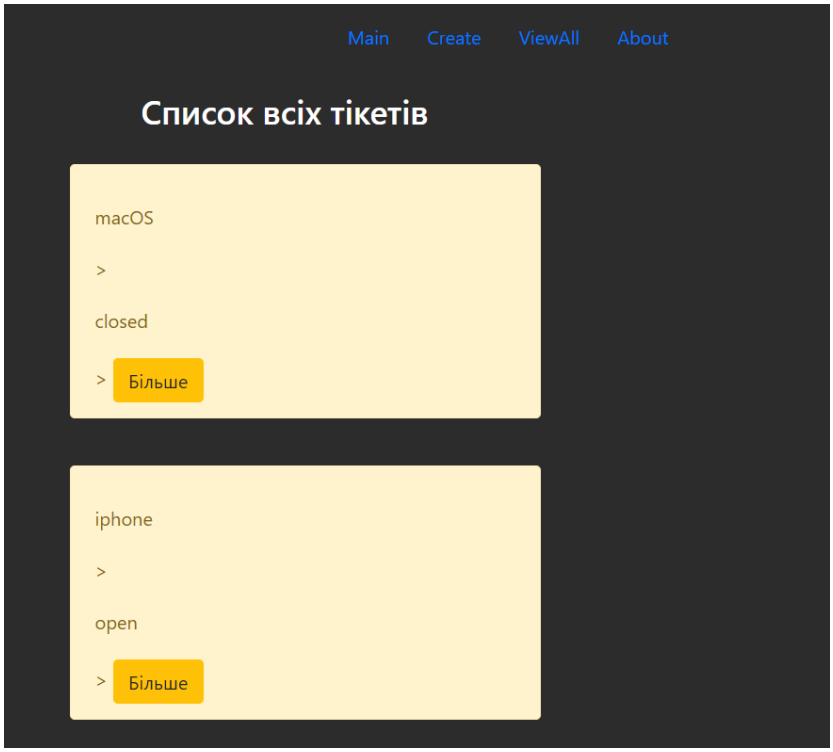

Рис. 3.3. Вікно списку завдань

На малюнку вище відображений вигляд процес групування оформлених заявок на виконання поставлених проблем, перейшовши натисканням по кнопці «Більше» можна побачити всю детальну інформацію про створений тікет.

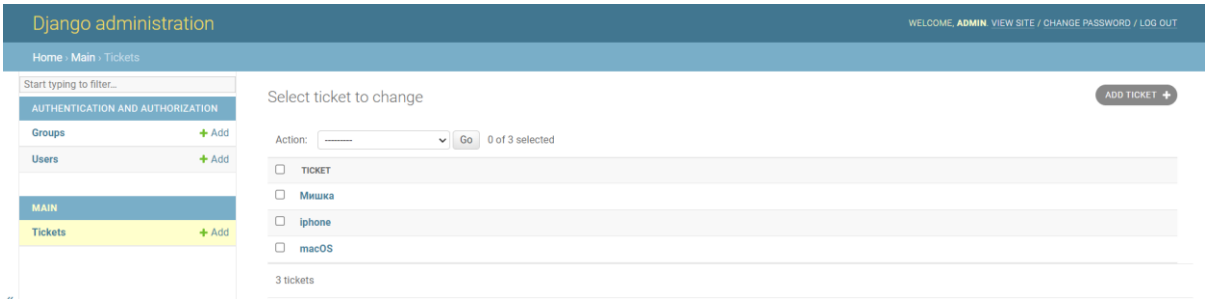

Рис. 3.3. Вікно адміністрування

На малюнку вище відображений вигляд вікна адміністрування, сторінка адміністратора додатку в якій можна вносити зміни до бази даних та налаштувань проекту.

Висновок до розділу 3

У третьому розділі проведено обґрунтування обрання Python, Django та CSS для розробки інформаційної системи. Проведено опис функціоналу системи.

Створена система має надзвичайно малий термін окупності, так як враховуються лише трудові затрати годин кваліфікованого співробітника, який вже працює на підприємстві, та вартість за оренду хост сервера.

ОСНОВНІ ВИСНОВКИ ТА РЕЗУЛЬТАТИ

Кваліфікаційна робота на здобуття освітнього ступеню магістра присвячена розв’язанню науково-технічного завдання покращенню якості управління процесом обрання замовлень на виконання послуг системним адміністратором, на прикладі ТОВ «Джи Ес Естейт». Метою роботи є пошук алгоритмів та підходів для підтримки прийняття рішень для ефективного обрання замовлень на послуги системного адміністратора комп'ютерних мереж, а також створення інформаційної системи для підтримки розв’язання поставленої задачі. Досліджено алгоритми та підходи для підтримки прийняття рішень для ефективного обрання замовлень на послуги системного адміністратора комп'ютерних мереж. Досліджено сучасні підходи та технології по створенню програмних продуктів. Визначено оптимальний набір інструментів для реалізації інформаційної системи підтримки обрання замовлень на виконання послуг системним адміністратором комп'ютерних мереж. Розроблено інформаційна система підтримки обрання замовлень на виконання послуг системним адміністратором комп'ютерних мереж на основі обраних алгоритмів та методів.

Створена система дозволяє обирати замовлення для їх виконання адміністратору комп’ютерної мережі чи цілій команді. На етапі обрання набору замовлень на виконання відбувається обрання тільки тих замовлень, що дозволить отримати найкращий ефект для виконавця у вигляді кінцевої оплати, але це не означає, що інші замовлення будуть не виконанні. Спочатку обираються критичні та більш важливі замовлення, а потім менше критичні та вартісні. За рахунок людино-машинного інтерфейсу є можливість скорегувати будь-який набір замовлень з власних міркувань адміністратору чи менеджеру, який розподіляє роботи між виконавцями. Використання створеної системи забезпечить системному адміністратору швидко приймати рішення при обранні

та плануванні своїх робіт на заданий проміжок часу. Практичне значення роботи полягає в тому, що створена система та її модулі можуть бути використані для створення web-орієнтованих систем підтримки прийняття ефективних рішень при управлінні підприємств із надання різних послуг, що дасть змогу вивести весь процес управління на якісно новий рівень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. IDEF0: функціональне моделювання ділових процесів, CASE-засоби (моделювання), Програмування, статті [Електронний ресурс]. Режим доступу: http://easy-code.com.ua/2012/08/idef0-funkcionalne-modelyuvannya-dilovix-procesiv-case-zasobi-modelyuvannya-programuvannya-statti/
2. Методологія функціонально-вартісного аналізу АВС (ФВА) [Елетронний ресурс]. Режим доступу: http://easy-code.com.ua/2011/04/metodologiya-funkcionalno-vartisnogo-analizu-abc-fva/
3. Левитин А. В. (2006) Метод грубой силы: Задача о рюкзаке, *Алгоритмы: введение в разработку и анализ* = Introduction to The Design and Analysis of Aigorithms. М.: Вильямс, С. 160-163.
4. Романовская А. М., Мендзив М. В. (2010) Динамическое программирование: Учебное пособие. Омск. 58 с. ISBN 978-5-91892-030-5
5. Пападимитриу Х., Стайглиц К. (1985) Комбинаторная оптимизация: Алгоритмы и сложность. М.: Мир.
6. Gendereau M., Potvin J.-I. (2019) Handbook of Metaheuristic, 3rd ed, Springer International Publishing AG. 604 pp.
7. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. (2005) Алгоритмы: построение и анализ. М.: Вильямс.
8. Каскадна модель життєвого циклу розробки ПО - Вершиніна Е.В., Гонченко М.С. http://ukrdoc.com.ua/text/7830/index-1.html?page=2.
9. ДОДАТОК А - Галузева система управління якістю [Елетронний ресурс]. Режим доступу: http://ua.convdocs.org/docs/index-131570.html?page=7.
10. Автоматизована система управління відділом технічної підтримки на підприємсті [Елетронний ресурс]. Режим доступу: http://revolution.allbest.ru/pramming/00554173_0.html.

11. Скрам [Елетронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/%D0%A1%D0%BA%D1%80%D0%B0%D0%BC
12. Poli R., Langdon W. B., McPhee N. F. (2008). *A Field Guide to Genetic Programming*. Lulu.com, freely available from the internet.
13. Gendereau M., Potvin J.-I. (2019) Handbook of Metaheuristic, 3rd ed, Springer International Publishing AG. 604 pp.
14. Hrybkov S., Kharkianen O., Ovcharuk V., Ovcharuk I. (2020) Development of Information Technology for Planning Order Fulfillment at a Food Enterprise. *Eastern-European Journal of Enterprise Technologies*, Vol. 1 (103), pp. 62-73.
15. Poli, R., Langdon, W. B., McPhee, N. F. (2008). *A Field Guide to Genetic Programming*. Lulu.com, freely available from the internet. ISBN 978-1-4092-0073-4.
16. Глосарій термінів з хімії / уклад. Й. Опейда, О. Швайка ; Ін-т фізико-органічної хімії та вуглехімії ім. Л. М. Литвиненка НАН України, Донецький національний університет. — Дон. : Вебер, 2008. — 738 с. — ISBN 978-966-335-206-0.
17. Donovan, Ryan (2020). The final Python 2 release marks the end of an era. *Stackoverflow Blog*. Stack Overflow. Процитовано 9 жовтня 2020.
18. The django Open Source Project on Open Hub: Languages Page — 2006.
19. *Дэвид Сойер Макфарланд.* Новая большая книга CSS = CSS: The Missing Manual. — Санкт-Петербург: Питер, 2017. — 720 с. — 1000 экз. — ISBN 978-5-496-02080-0.
20. *Эд Титтел, Джефф Ноубл.* HTML, XHTML и CSS для чайников, 7-е издание = HTML, XHTML & CSS For Dummies, 7th Edition. — М.: «Диалектика», 2011. — 400 с. — ISBN 978-5-8459-1752-2.
21. *Стивен Шафер.* HTML, XHTML и CSS. Библия пользователя, 5-е издание = HTML, XHTML, and CSS Bible, 5th Edition. — М.: «Диалектика», 2011. — 656 с. — ISBN 978-5-8459-1676-1.

22. Типове положення з планування, обліку і калькулювання собівартості продукції (робіт, послуг) у промисловості. Постанова КМ України від 26.04.1996 № 473. («Закон і бізнес» № 41; 42; 43 за 1996р. «Галицькі контракти», «Бізнес» та ін. періодичні видання за вересень - жовтень 1996р).
23. HELPDESK — КАКУЮ ВЫБРАТЬ СИСТЕМУ ПОДДЕРЖКИ ПОЛЬЗОВАТЕЛЕЙ? [Елетронний ресурс]. Режим доступу: https://blog.sprinthost.ru/2011/07/28/helpdesk-how-to-choose/

ДОДАТКИ

Додаток 1.

Створення форм, даний файл створює форми для введення інформації в нашу базу даних. Реалізований створенням класу «class ticketForm» який наслідує стандартний клас «django»

```
from django.forms import ModelForm, TextInput, Textarea, DateInput
from .models import ticket


class ticketForm(ModelForm):
    class Meta:
        model = ticket
        fields = ['title', 'full_text', 'customer', 'status', 'date']

        widgets = {
            'title': TextInput(attrs={
                'class': 'form-control',
                'placeholder': 'Проблема'
            }),
            'full_text': Textarea(attrs={
                'class': 'form-control',
                'placeholder': 'Опис проблеми'
            }),
            'customer': TextInput(attrs={
                'class': 'form-control',
                'placeholder': 'Ваше ПІБ'
            }),
            'status': TextInput(attrs={
                'class': 'form-control',
                'placeholder': 'status'
            }),
            'date': DateInput(attrs={
                'class': 'form-control',
                'placeholder': 'Дата'
            }),
        }
```

Додаток 2.

Файл налаштування проекту який створений на Django. В даному файлі описується базова безпека проекту та налаштування роботи сайту (debug mode, Admin panel etc.)

```
Django settings for helpdesk project.

Generated by 'django-admin startproject' using Django 4.0.

For more information on this file, see
https://docs.djangoproject.com/en/4.0/topics/settings/

"""

from pathlib import Path

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


# Q_u_i_c_k_-st_a_rt deve_lop_ment sett_ings – unsu_itable for production
# See https://docs.djangoproject.com/en/4.0/howto/deployment/checklist/

# SEC WARN keep the sec!
SECRET_KEY = 'django-insecure-f@7%4_0w72)4av6%^q0l1r#*$!l993=@-
^nyckpwwef266(u(l'

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

ALLOWED_HOSTS = []


# Appl_icat-ion defin_ition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
```

```
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'main',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'helpdesk.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'helpdesk.wsgi.application'

# Database
# https://docs.djangoproject.com/en/4.0/ref/settings/#databases

DATABASES = {
    'default': {
```

```
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}


# Password validation
# https://docs.djangoproject.com/en/4.0/ref/settings/#auth-password-validators

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

# Internationalization
# https://docs.djangoproject.com/en/4.0/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.0/howto/static-files/

STATIC_URL = '/static/'
```

```
# Default primary key field type
# https://docs.djangoproject.com/en/4.0/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

STATICFILES_DIRS = [BASE_DIR / "static"]
```

Додаток 3.

Базова сторінка HTML створена як основна для всіх майбутніх web-сторінок додатку.

```
{% load static %}
<!doctype html>
<html lang="ru">
<head>
   <meta charset="UTF-8">
   <meta name="viewport"
         content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
   <meta http-equiv="X-UA-Compatible" content="ie=edge">
   <title>{% block title %}{% endblock %}</title>
   <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
   <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.8.2/css/all.css">
    <link rel="stylesheet" href="{% static 'main/css/main.css' %}">
   <header class="d-flex justify-content-center py-3">
     <ul class="nav nav-pills">
        <li class="nav-item"><a href="/index" class="nav-link" aria-current="page">Main</a></li>
       <li class="nav-item"><a href="/create" class="nav-link" aria-current="page">Create</a></li>
       <li class="nav-item"><a href="/views" class="nav-link">ViewAll</a></li>
       <li class="nav-item"><a href="/about" class="nav-link">About</a></li>
     </ul>
   </header>
</head>
<body>
   <main>
      {% block content %}
      {% endblock %}
   </main>
</body>
</html>
```

Додаток 4.

Файл створений для відображення логіки додатку. Кожен клас відповідає за окрему веб сторінку

```
from django.shortcuts import render
from .models import ticket
from .forms import ticketForm
from django.views.generic import DetailView, UpdateView, DeleteView

class TicketDetaliView(DetailView):
    model = ticket
    template_name = "main/details_view.html"
    context_object_name = 'ticket'

class TicketUpdateView(UpdateView):
    model = ticket
    template_name = "main/create.html"

    form_class = ticketForm


class TicketDelete(DeleteView):
    model = ticket
    success_url = '/views'
    template_name = "main/ticket-delete.html"


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

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


def views(request):
    data = ticket.objects.all()
    return render(request, 'main/views.html', {'data': data})

def create(request):
```

```
err = ''

if request.method == 'POST':
    form = ticketForm(request.POST)
    if form.is_valid():
        form.save()
    else:
        err = 'Error'

form = ticketForm()

date = {
    'form': form,
    'err': err
}
return render(request, 'main/create.html', date)
```

Додаток 5.

Файл описує навігацію по додатку, обробляє веб запити користувача та вказує який клас відображення виконувати.

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

urlpatterns = [
    path('', views.index),
    path('create', views.create),
    path('views', views.views),
    path('about', views.about),
    path('<int:pk>', views.TicketDetaliView.as_view(), name='ticket-detail'),
    path('<int:pk>/update', views.TicketUpdateView.as_view(), name='ticket-update'),
    path('<int:pk>/delete', views.TicketDelete.as_view(), name='ticket-delete'),
```

Додаток 6

Відображення модифікованої бібліотеки шаблонізатора «Jinja». В силу того, що стандартна бібліотека шаблонізатора «Jinja» не влаштовувала створювану систему була виконана її модифікація.

```
import enum
import json
import os
import re
import typing as t
import warnings
from collections import abc
from collections import deque
from random import choice
from random import randrange
from threading import Lock
from types import CodeType
from urllib.parse import quote_from_bytes

import markupsafe

if t.TYPE_CHECKING:
```

```
    import typing_extensions as te

F = t.TypeVar("F", bound=t.Callable[..., t.Any])

# special singleton representing missing values for the runtime
missing: t.Any = type("MissingType", (), {"__repr__": lambda x: "missing"})()

internal_code: t.MutableSet[CodeType] = set()

concat = "".join


def pass_context(f: F) -> F:
    """Pass the :class:`~jinja2.runtime.Context` as the first argument
    to the decorated function when called while rendering a template.

    Can be used on functions, filters, and tests.

    If only ``Context.eval_context`` is needed, use
    :func:`pass_eval_context`. If only ``Context.environment`` is
    needed, use :func:`pass_environment`.
```

```
    .. versionadded:: 3.0.0
       Replaces ``contextfunction`` and ``contextfilter``.
    """
    f.jinja_pass_arg = _PassArg.context  # type: ignore
    return f


def pass_eval_context(f: F) -> F:
    """Pass the :class:`~jinja2.nodes.EvalContext` as the first argument
    to the decorated function when called while rendering a template.
    See :ref:`eval-context`.

    Can be used on functions, filters, and tests.

    If only ``EvalContext.environment`` is needed, use
    :func:`pass_environment`.

    .. versionadded:: 3.0.0
       Replaces ``evalcontextfunction`` and ``evalcontextfilter``.
    """
```

```
    f.jinja_pass_arg = _PassArg.eval_context  # type: ignore
    return f


def pass_environment(f: F) -> F:
    """Pass the :class:`~jinja2.Environment` as the first argument to
    the decorated function when called while rendering a template.

    Can be used on functions, filters, and tests.

    .. versionadded:: 3.0.0
       Replaces ``environmentfunction`` and ``environmentfilter``.
    """
    f.jinja_pass_arg = _PassArg.environment  # type: ignore
    return f


class _PassArg(enum.Enum):
    context = enum.auto()
    eval_context = enum.auto()
    environment = enum.auto()
```

```
@classmethod
def from_obj(cls, obj: F) -> t.Optional["_PassArg"]:
    if hasattr(obj, "jinja_pass_arg"):
        return obj.jinja_pass_arg  # type: ignore

    for prefix in "context", "eval_context", "environment":
        squashed = prefix.replace("_", "")

        for name in f"{squashed}function", f"{squashed}filter":
            if getattr(obj, name, False) is True:
                warnings.warn(
                    f"{name!r} is deprecated and will stop working"
                    f" in Jinja 3.1. Use 'pass_{prefix}' instead.",
                    DeprecationWarning,
                    stacklevel=2,
                )
                return cls[prefix]

    return None
```

```
def contextfunction(f: F) -> F:
    """Pass the context as the first argument to the decorated function.

    .. deprecated:: 3.0
        Will be removed in Jinja 3.1. Use :func:`~jinja2.pass_context`
        instead.
    """
    warnings.warn(
        "'contextfunction' is renamed to 'pass_context', the old name"
        " will be removed in Jinja 3.1.",
        DeprecationWarning,
        stacklevel=2,
    )
    return pass_context(f)


def evalcontextfunction(f: F) -> F:
    """Pass the eval context as the first argument to the decorated
    function.

    .. deprecated:: 3.0
```

```
        Will be removed in Jinja 3.1. Use
        :func:`~jinja2.pass_eval_context` instead.

    .. versionadded:: 2.4
    """
    warnings.warn(
        "'evalcontextfunction' is renamed to 'pass_eval_context', the"
        " old name will be removed in Jinja 3.1.",
        DeprecationWarning,
        stacklevel=2,
    )
    return pass_eval_context(f)


def environmentfunction(f: F) -> F:
    """Pass the environment as the first argument to the decorated
    function.

    .. deprecated:: 3.0
        Will be removed in Jinja 3.1. Use
        :func:`~jinja2.pass_environment` instead.
```

```
    """
    warnings.warn(
        "'environmentfunction' is renamed to 'pass_environment', the"
        " old name will be removed in Jinja 3.1.",
        DeprecationWarning,
        stacklevel=2,
    )
    return pass_environment(f)


def internalcode(f: F) -> F:
    """Marks the function as internally used"""
    internal_code.add(f.__code__)
    return f


def is_undefined(obj: t.Any) -> bool:
    """Check if the object passed is undefined.  This does nothing more than
    performing an instance check against :class:`Undefined` but looks nicer.
    This can be used for custom filters or tests that want to react to
    undefined variables.  For example a custom default filter can look like
```

```
    this::

        def default(var, default=''):
            if is_undefined(var):
                return default
            return var
    """
    from .runtime import Undefined

    return isinstance(obj, Undefined)


def consume(iterable: t.Iterable[t.Any]) -> None:
    """Consumes an iterable without doing anything with it."""
    for _ in iterable:
        pass


def clear_caches() -> None:
    """Jinja keeps internal caches for environments and lexers.  These are
    used so that Jinja doesn't have to recreate environments and lexers all
```

```
    the time.  Normally you don't have to care about that but if you are
    measuring memory consumption you may want to clean the caches.
    """
    from .environment import get_spontaneous_environment
    from .lexer import _lexer_cache

    get_spontaneous_environment.cache_clear()
    _lexer_cache.clear()



def import_string(import_name: str, silent: bool = False) -> t.Any:
    """Imports an object based on a string.  This is useful if you want to
    use import paths as endpoints or something similar.  An import path can
    be specified either in dotted notation (``xml.sax.saxutils.escape``)
    or with a colon as object delimiter (``xml.sax.saxutils:escape``).

    If the `silent` is True the return value will be `None` if the import
    fails.

    :return: imported object
    """
```

```
    try:
        if ":" in import_name:
            module, obj = import_name.split(":", 1)
        elif "." in import_name:
            module, _, obj = import_name.rpartition(".")
        else:
            return __import__(import_name)
        return getattr(__import__(module, None, None, [obj]), obj)
    except (ImportError, AttributeError):
        if not silent:
            raise



def open_if_exists(filename: str, mode: str = "rb") -> t.Optional[t.IO]:
    """Returns a file descriptor for the filename if that file exists,
    otherwise ``None``.
    """
    if not os.path.isfile(filename):
        return None

    return open(filename, mode)
```

```
def object_type_repr(obj: t.Any) -> str:
    """Returns the name of the object's type.  For some recognized
    singletons the name of the object is returned instead. (For
    example for `None` and `Ellipsis`).
    """
    if obj is None:
        return "None"
    elif obj is Ellipsis:
        return "Ellipsis"

    cls = type(obj)

    if cls.__module__ == "builtins":
        return f"{cls.__name__} object"

    return f"{cls.__module__}.{cls.__name__} object"


def pformat(obj: t.Any) -> str:
```

```
    """Format an object using :func:`pprint.pformat`."""
    from pprint import pformat  # type: ignore

    return pformat(obj)



_http_re = re.compile(
    r"""
    ^
    (
        (https?://|www\.)  # scheme or www
        (([\w%-]+\.)+)?  # subdomain
        (
            [a-z]{2,63}  # basic tld
        |
            xn--[\w%]{2,59}  # idna tld
        )
    |
        ([\w%-]{2,63}\.)+  # basic domain
        (com|net|int|edu|gov|org|info|mil)  # basic tld
    |
```

```
        (https?://)  # scheme
        (
            (([\d]{1,3})(\.[\d]{1,3}){3})  # IPv4
        |
            (\[([\da-f]{0,4}:){2}([\da-f]{0,4}:?){1,6}])  # IPv6
        )
    )
    (?::[\d]{1,5})?  # port
    (?:[/?#]\S*)?  # path, query, and fragment
    $
    """,
    re.IGNORECASE | re.VERBOSE,
)
_email_re = re.compile(r"^\S+@\w[\w.-]*\.\w+$")


def urlize(
    text: str,
    trim_url_limit: t.Optional[int] = None,
    rel: t.Optional[str] = None,
    target: t.Optional[str] = None,
```

```
    extra_schemes: t.Optional[t.Iterable[str]] = None,
) -> str:
    """Convert URLs in text into clickable links.

    This may not recognize links in some situations. Usually, a more
    comprehensive formatter, such as a Markdown library, is a better
    choice.

    Works on ``http://``, ``https://``, ``www.``, ``mailto:``, and email
    addresses. Links with trailing punctuation (periods, commas, closing
    parentheses) and leading punctuation (opening parentheses) are
    recognized excluding the punctuation. Email addresses that include
    header fields are not recognized (for example,
    ``mailto:address@example.com?cc=copy@example.com``).

    :param text: Original text containing URLs to link.
    :param trim_url_limit: Shorten displayed URL values to this length.
    :param target: Add the ``target`` attribute to links.
    :param rel: Add the ``rel`` attribute to links.
    :param extra_schemes: Recognize URLs that start with these schemes
        in addition to the default behavior.
```

```
.. versionchanged:: 3.0
    The ``extra_schemes`` parameter was added.

.. versionchanged:: 3.0
    Generate ``https://`` links for URLs without a scheme.

.. versionchanged:: 3.0
    The parsing rules were updated. Recognize email addresses with
    or without the ``mailto:`` scheme. Validate IP addresses. Ignore
    parentheses and brackets in more cases.
"""
if trim_url_limit is not None:

    def trim_url(x: str) -> str:
        if len(x) > trim_url_limit:  # type: ignore
            return f"{x[:trim_url_limit]}..."

        return x

else:
```

```
    def trim_url(x: str) -> str:
        return x


words = re.split(r"(\s+)", str(markupsafe.escape(text)))
rel_attr = f' rel="{markupsafe.escape(rel)}"' if rel else ""
target_attr = f' target="{markupsafe.escape(target)}"' if target else ""

for i, word in enumerate(words):
    head, middle, tail = "", word, ""
    match = re.match(r"^([(<]|&lt;)+", middle)

    if match:
        head = match.group()
        middle = middle[match.end() :]

    # Unlike lead, which is anchored to the start of the string,
    # need to check that the string ends with any of the characters
    # before trying to match all of them, to avoid backtracking.
    if middle.endswith((")", ">", ".", ",", "\n", "&gt;")):
        match = re.search(r"([)>.,\n]|&gt;)+$", middle)
```

```
    if match:
        tail = match.group()
        middle = middle[: match.start()]

# Prefer balancing parentheses in URLs instead of ignoring a
# trailing character.
for start_char, end_char in ("(", ")"), ("<", ">"), ("&lt;", "&gt;"):
    start_count = middle.count(start_char)

    if start_count <= middle.count(end_char):
        # Balanced, or lighter on the left
        continue

    # Move as many as possible from the tail to balance
    for _ in range(min(start_count, tail.count(end_char))):
        end_index = tail.index(end_char) + len(end_char)
        # Move anything in the tail before the end char too
        middle += tail[:end_index]
        tail = tail[end_index:]
```

```
if _http_re.match(middle):
    if middle.startswith("https://") or middle.startswith("http://"):
        middle = (
            f'<a href="{middle}"{rel_attr}{target_attr}>{trim_url(middle)}</a>'
        )
    else:
        middle = (
            f'<a href="https://{middle}"{rel_attr}{target_attr}>'
            f"{trim_url(middle)}</a>"
        )

elif middle.startswith("mailto:") and _email_re.match(middle[7:]):
    middle = f'<a href="{middle}">{middle[7:]}</a>'

elif (
    "@" in middle
    and not middle.startswith("www.")
    and ":" not in middle
    and _email_re.match(middle)
):
    middle = f'<a href="mailto:{middle}">{middle}</a>'
```

```
        elif extra_schemes is not None:
            for scheme in extra_schemes:
                if middle != scheme and middle.startswith(scheme):
                    middle = f'<a href="{middle}"{rel_attr}{target_attr}>{middle}</a>'

        words[i] = f"{head}{middle}{tail}"

    return "".join(words)



def generate_lorem_ipsum(
    n: int = 5, html: bool = True, min: int = 20, max: int = 100
) -> str:
    """Generate some lorem ipsum for the template."""
    from .constants import LOREM_IPSUM_WORDS

    words = LOREM_IPSUM_WORDS.split()
    result = []

    for _ in range(n):
```

```
next_capitalized = True
last_comma = last_fullstop = 0
word = None
last = None
p = []

# each paragraph contains out of 20 to 100 words.
for idx, _ in enumerate(range(randrange(min, max))):
    while True:
        word = choice(words)
        if word != last:
            last = word
            break
    if next_capitalized:
        word = word.capitalize()
        next_capitalized = False
    # add commas
    if idx - randrange(3, 8) > last_comma:
        last_comma = idx
        last_fullstop += 2
        word += ","
```

```
            # add end of sentences
            if idx - randrange(10, 20) > last_fullstop:
                last_comma = last_fullstop = idx
                word += "."
                next_capitalized = True
            p.append(word)

        # ensure that the paragraph ends with a dot.
        p_str = " ".join(p)

        if p_str.endswith(","):
            p_str = p_str[:-1] + "."
        elif not p_str.endswith("."):
            p_str += "."

        result.append(p_str)

    if not html:
        return "\n\n".join(result)
    return markupsafe.Markup(
        "\n".join(f"<p>{markupsafe.escape(x)}</p>" for x in result)
```

```
    )



def url_quote(obj: t.Any, charset: str = "utf-8", for_qs: bool = False) -> str:
    """Quote a string for use in a URL using the given charset.

    :param obj: String or bytes to quote. Other types are converted to
        string then encoded to bytes using the given charset.
    :param charset: Encode text to bytes using this charset.
    :param for_qs: Quote "/" and use "+" for spaces.
    """
    if not isinstance(obj, bytes):
        if not isinstance(obj, str):
            obj = str(obj)

        obj = obj.encode(charset)

    safe = b"" if for_qs else b"/"
    rv = quote_from_bytes(obj, safe)

    if for_qs:
```

```
        rv = rv.replace("%20", "+")

    return rv



def unicode_urlencode(obj: t.Any, charset: str = "utf-8", for_qs: bool = False) -> str:
    import warnings

    warnings.warn(
        "'unicode_urlencode' has been renamed to 'url_quote'. The old"
        " name will be removed in Jinja 3.1.",
        DeprecationWarning,
        stacklevel=2,
    )
    return url_quote(obj, charset=charset, for_qs=for_qs)



@abc.MutableMapping.register
class LRUCache:
    """A simple LRU Cache implementation."""
```

```
# this is fast for small capacities (something below 1000) but doesn't
# scale.  But as long as it's only used as storage for templates this
# won't do any harm.

def __init__(self, capacity: int) -> None:
    self.capacity = capacity
    self._mapping: t.Dict[t.Any, t.Any] = {}
    self._queue: "te.Deque[t.Any]" = deque()
    self._postinit()

def _postinit(self) -> None:
    # alias all queue methods for faster lookup
    self._popleft = self._queue.popleft
    self._pop = self._queue.pop
    self._remove = self._queue.remove
    self._wlock = Lock()
    self._append = self._queue.append

def __getstate__(self) -> t.Mapping[str, t.Any]:
    return {
        "capacity": self.capacity,
```

```
        "_mapping": self._mapping,
        "_queue": self._queue,
    }

def __setstate__(self, d: t.Mapping[str, t.Any]) -> None:
    self.__dict__.update(d)
    self._postinit()

def __getnewargs__(self) -> t.Tuple:
    return (self.capacity,)

def copy(self) -> "LRUCache":
    """Return a shallow copy of the instance."""
    rv = self.__class__(self.capacity)
    rv._mapping.update(self._mapping)
    rv._queue.extend(self._queue)
    return rv

def get(self, key: t.Any, default: t.Any = None) -> t.Any:
    """Return an item from the cache dict or `default`"""
    try:
```

```
        return self[key]
    except KeyError:
        return default

def setdefault(self, key: t.Any, default: t.Any = None) -> t.Any:
    """Set `default` if the key is not in the cache otherwise
    leave unchanged. Return the value of this key.
    """
    try:
        return self[key]
    except KeyError:
        self[key] = default
        return default

def clear(self) -> None:
    """Clear the cache."""
    with self._wlock:
        self._mapping.clear()
        self._queue.clear()

def __contains__(self, key: t.Any) -> bool:
```

```
    """Check if a key exists in this cache."""
    return key in self._mapping

def __len__(self) -> int:
    """Return the current size of the cache."""
    return len(self._mapping)

def __repr__(self) -> str:
    return f"<{type(self).__name__} {self._mapping!r}>"

def __getitem__(self, key: t.Any) -> t.Any:
    """Get an item from the cache. Moves the item up so that it has the
    highest priority then.

    Raise a `KeyError` if it does not exist.
    """
    with self._wlock:
        rv = self._mapping[key]

        if self._queue[-1] != key:
            try:
```

```
                self._remove(key)
            except ValueError:
                # if something removed the key from the container
                # when we read, ignore the ValueError that we would
                # get otherwise.
                pass

            self._append(key)

        return rv

def __setitem__(self, key: t.Any, value: t.Any) -> None:
    """Sets the value for an item. Moves the item up so that it
    has the highest priority then.
    """
    with self._wlock:
        if key in self._mapping:
            self._remove(key)
        elif len(self._mapping) == self.capacity:
            del self._mapping[self._popleft()]
```

```
        self._append(key)
        self._mapping[key] = value


def __delitem__(self, key: t.Any) -> None:
    """Remove an item from the cache dict.
    Raise a `KeyError` if it does not exist.
    """
    with self._wlock:
        del self._mapping[key]

        try:
            self._remove(key)
        except ValueError:
            pass


def items(self) -> t.Iterable[t.Tuple[t.Any, t.Any]]:
    """Return a list of items."""
    result = [(key, self._mapping[key]) for key in list(self._queue)]
    result.reverse()
    return result
```

```
    def values(self) -> t.Iterable[t.Any]:
        """Return a list of all values."""
        return [x[1] for x in self.items()]

    def keys(self) -> t.Iterable[t.Any]:
        """Return a list of all keys ordered by most recent usage."""
        return list(self)

    def __iter__(self) -> t.Iterator[t.Any]:
        return reversed(tuple(self._queue))

    def __reversed__(self) -> t.Iterator[t.Any]:
        """Iterate over the keys in the cache dict, oldest items
        coming first.
        """
        return iter(tuple(self._queue))
    __copy__ = copy
def select_autoescape(
    enabled_extensions: t.Collection[str] = ("html", "htm", "xml"),
    disabled_extensions: t.Collection[str] = (),
    default_for_string: bool = True,
```

```
    default: bool = False,
) -> t.Callable[[t.Optional[str]], bool]:
    """Intelligently sets the initial value of autoescaping based on the
    filename of the template.  This is the recommended way to configure
    autoescaping if you do not want to write a custom function yourself.

    If you want to enable it for all templates created from strings or
    for all templates with `.html` and `.xml` extensions::

        from jinja2 import Environment, select_autoescape
        env = Environment(autoescape=select_autoescape(
            enabled_extensions=('html', 'xml'),
            default_for_string=True,
        ))

    Example configuration to turn it on at all times except if the template
    ends with `.txt`::

        from jinja2 import Environment, select_autoescape
        env = Environment(autoescape=select_autoescape(
            disabled_extensions=('txt',),
```

```
        default_for_string=True,
        default=True,
    ))
The `enabled_extensions` is an iterable of all the extensions that
autoescaping should be enabled for.  Likewise `disabled_extensions` is
a list of all templates it should be disabled for.  If a template is
loaded from a string then the default from `default_for_string` is used.
If nothing matches then the initial value of autoescaping is set to the
vale of `default`.
For security reasons this function operates case insensitive.
.. versionadded:: 2.9
"""
enabled_patterns = tuple(f".{x.lstrip('.').lower()}" for x in enabled_extensions)
disabled_patterns = tuple(f".{x.lstrip('.').lower()}" for x in disabled_extensions)

def autoescape(template_name: t.Optional[str]) -> bool:
    if template_name is None:
        return default_for_string
    template_name = template_name.lower()
    if template_name.endswith(enabled_patterns):
        return True
```

```
        if template_name.endswith(disabled_patterns):
            return False
        return default
    return autoescape


def htmlsafe_json_dumps(
    obj: t.Any, dumps: t.Optional[t.Callable[..., str]] = None, **kwargs: t.Any
) -> markupsafe.Markup:
    """Serialize an object to a string of JSON with :func:`json.dumps`,
    then replace HTML-unsafe characters with Unicode escapes and mark
    the result safe with :class:`~markupsafe.Markup`.

    This is available in templates as the ``|tojson`` filter.

    The following characters are escaped: ``<``, ``>``, ``&``, ``'``.

    The returned string is safe to render in HTML documents and
    ``<script>`` tags. The exception is in HTML attributes that are
    double quoted; either use single quotes or the ``|forceescape``
    filter.
```

```
:param obj: The object to serialize to JSON.
:param dumps: The ``dumps`` function to use. Defaults to
    ``env.policies["json.dumps_function"]``, which defaults to
    :func:`json.dumps`.
:param kwargs: Extra arguments to pass to ``dumps``. Merged onto
    ``env.policies["json.dumps_kwargs"]``.

.. versionchanged:: 3.0
    The ``dumper`` parameter is renamed to ``dumps``.

.. versionadded:: 2.9
"""
if dumps is None:
    dumps = json.dumps

return markupsafe.Markup(
    dumps(obj, **kwargs)
    .replace("<", "\\u003c")
    .replace(">", "\\u003e")
    .replace("&", "\\u0026")
    .replace("'", "\\u0027")
```

```
    )


class Cycler:
    """Cycle through values by yield them one at a time, then restarting
    once the end is reached. Available as ``cycler`` in templates.

    Similar to ``loop.cycle``, but can be used outside loops or across
    multiple loops. For example, render a list of folders and files in a
    list, alternating giving them "odd" and "even" classes.

    .. code-block:: html+jinja

        {% set row_class = cycler("odd", "even") %}
        <ul class="browser">
        {% for folder in folders %}
          <li class="folder {{ row_class.next() }}">{{ folder }}
        {% endfor %}
        {% for file in files %}
          <li class="file {{ row_class.next() }}">{{ file }}
        {% endfor %}
        </ul>
```

```
:param items: Each positional argument will be yielded in the order
    given for each cycle.

.. versionadded:: 2.1
"""

def __init__(self, *items: t.Any) -> None:
    if not items:
        raise RuntimeError("at least one item has to be provided")
    self.items = items
    self.pos = 0

def reset(self) -> None:
    """Resets the current item to the first item."""
    self.pos = 0

@property
def current(self) -> t.Any:
    """Return the current item. Equivalent to the item that will be
    returned next time :meth:`next` is called.
```

```
        """
        return self.items[self.pos]

    def next(self) -> t.Any:
        """Return the current item, then advance :attr:`current` to the
        next item.
        """
        rv = self.current
        self.pos = (self.pos + 1) % len(self.items)
        return rv
    __next__ = next

class Joiner:
    """A joining helper for templates."""
    def __init__(self, sep: str = ", ") -> None:
        self.sep = sep
        self.used = False
    def __call__(self) -> str:
        if not self.used:
            self.used = True
            return ""
```

```
        return self.sep
    class Namespace:
        """A namespace object that can hold arbitrary attributes.  It may be
        initialized from a dictionary or with keyword arguments."""

        def __init__(*args: t.Any, **kwargs: t.Any) -> None:  # noqa: B902
            self, args = args[0], args[1:]
            self.__attrs = dict(*args, **kwargs)

        def __getattribute__(self, name: str) -> t.Any:
            # __class__ is needed for the awaitable check in async mode
            if name in {"_Namespace__attrs", "__class__"}:
                return object.__getattribute__(self, name)
            try:
                return self.__attrs[name]
            except KeyError:
                raise AttributeError(name) from None

        def __setitem__(self, name: str, value: t.Any) -> None:
            self.__attrs[name] = value
```

```
    def __repr__(self) -> str:
        return f"<Namespace {self.__attrs!r}>"
class Markup(markupsafe.Markup):
    def __new__(cls, base="", encoding=None, errors="strict"):  # type: ignore
        warnings.warn(
            "'jinja2.Markup' is deprecated and will be removed in Jinja"
            " 3.1. Import 'markupsafe.Markup' instead.",
            DeprecationWarning,
            stacklevel=2,
        )
        return super().__new__(cls, base, encoding, errors)
def escape(s: t.Any) -> str:
    warnings.warn(
        "'jinja2.escape' is deprecated and will be removed in Jinja"
        " 3.1. Import 'markupsafe.escape' instead.",
        DeprecationWarning,
        stacklevel=2,
    )
    return markupsafe.escape(s)
```

Додаток 7

База даних створеної системи. У зв’язку з тим що використовується код для створення бази даних SQLite відобразити базу даних можливо в наступному вигляді.

Поле	Тип даних	Призначення
ID	int	Ключове поле яке створене для індексації даних
Title	char	Поле яке містить заголовок заявки
Full text	char	Поле яке містить опис проблеми
Customer	char	Поле яке містить дані автора заявки
Status	char	Поле яке містить статус заявки
Date	date	Поле яке містить дату створення заявки