

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) Автоматизації і комп'ютерних систем
Кафедра Інформаційних технологій, штучного інтелекту і кібербезпеки

«До захисту в ЕК»
Директор інституту(декан
факультету)

Андрій ФОРСЮК
(ім'я та Прізвище)

02 2025 р.

«До захисту допущено»
Завідувач кафедри

Сергій ГРИБКОВ
(ім'я та Прізвище)

02 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

зі спеціальності 122 «Комп'ютерні науки»
(код та назва спеціальності)
освітньо-професійної програми Комп'ютерні науки

на тему: «Розроблення кроссплатформенного месенджера для внутрішньої комунікації співробітників підприємства "Dnipro-M"»

Виконав: здобувач 5 курсу, групи ЗКН-5-2

Кайстриюков Ігор Віталійович
(прізвище, ім'я, по батькові повністю)

(підпис)

Керівник Грама Михайло Петрович
(прізвище, ім'я та по батькові повністю)

(підпис)

Консультанти Грама М.П
(ім'я та прізвище)

(підпис)

(ім'я та прізвище)

(підпис)

(ім'я та прізвище)

(підпис)

Рецензент

Кайстриюков І.В.
(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 – 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) Автоматизації і комп'ютерних систем
Кафедра Інформаційних технологій, штучного інтелекту і кібербезпеки

«До захисту в ЕК»
 Директор інституту(декан
 факультету)

_____ Андрій ФОРСЮК _____
 (підпис) (ім'я та Прізвище)

« ____ » _____ 2025 р.

«До захисту допущено»
 Завідувач кафедри

_____ Сергій ГРИБКОВ _____
 (підпис) (ім'я та Прізвище)

« ____ » _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

зі спеціальності 122 «Комп'ютерні науки»
 (код та назва спеціальності)
 освітньо-професійної програми Комп'ютерні науки

на тему: «Розроблення кросплатформенного месенджера для внутрішньої комунікації співробітників підприємства "Dnipro-M"»

Виконав: здобувач 5 курсу, групи ЗКН-5-2

_____ Кайстриюков Ігор Віталійович _____
 (прізвище, ім'я, по батькові повністю) (підпис)

Керівник _____ Грама Михайло Петрович _____
 (прізвище, ім'я та по батькові повністю) (підпис)

Консультанти _____ Грама М.П _____
 (ім'я та прізвище) (підпис)

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

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

Рецензент _____
 (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 – 2025 р.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ
 Інститут (факультет) Автоматизації і комп'ютерних систем
 Кафедра Інформаційних технологій, штучного інтелекту і кібербезпеки
 Освітній ступінь Бакалавр
 Спеціальність 122 «Комп'ютерні науки»
 (код і назва)
 Освітньо-професійна програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
 Завідувач кафедри
Грибков С.В.

“ 07 ” жовтня 2024 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

Кайстриюкова Ігора Віталійовича

(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення кросплатформенного месенджера для внутрішньої комунікації співробітників підприємства "Дніпро-М"
 керівник роботи старший викладач Грама Михайло Петрович

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “07” жовтня 2024 року №886-кв

2. Строк подання здобувачем роботи 03 лютого 2025 року

3. Вихідні дані до роботи

1. Дані про компанію «Dnipro-M»

2. Дані про співробітників компанії

3. Посадові інструкції

4. Положення

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

1. Системний аналіз проектування додатків корпоративного спілкування

2. Моделі бази даних додатку

3. Інтерфейс програми для комунікації працівників компанії «Dnipro-M»

4. Охорона праці та техніка безпеки

5. Перелік графічного матеріалу

1. Організаційна структура магазину компанії «Dnipro-M»

2. Фізична модель бази даних

3. UML-діаграма можливостей співробітника в системі

4. Алгоритм роботи модуля обробки нового користувача

5. Командна мапа користувача

6. Приклади роботи програми

7. Фрагменти коду

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Грама М.П., старший викладач	07.10.24	26.11.24
2	Грама М.П., старший викладач	27.11.24	16.12.24
3	Грама М.П., старший викладач	16.12.24	10.01.25
4	Грама М.П., старший викладач	13.01.25	20.01.25

7. Дата видачі завдання 07.10.2024**КАЛЕНДАРНИЙ ПЛАН**

№	Назва етапів виконання кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Системний аналіз підприємства та постановка задачі на проектування	08.10.2024 - 28.11.2024	Виконано
2	Проектування БД	27.11.2024 - 16.12.2024	Виконано
3	Створення ІС та написання інструкцій користувача	16.12.2024 - 11.01.2025	Виконано
4	Оформлення пояснювальної записки та презентації	12.01.2025 - 25.01.2025	Виконано

Здобувач

(підпис)

Кайстриюков І.В.

(прізвище та ініціали)

Керівник роботи

(підпис)

Грама М.П.

(прізвище та ініціали)

АНОТАЦІЯ

Дипломна робота за темою: Розроблення кроссплатформенного месенджера для внутрішньої комунікації співробітників підприємства «Dnipro-M».

Виконав студент: Кайстриюков Ігор Віталійович

Дипломна робота містить: 99 сторінок друкованого тексту, 37 рисунків, 5 таблиць та нараховує 33 джерела використаної інформації.

Об'єкт дослідження: розробка корпоративного месенджера для внутрішньої комунікації. Предмет дослідження: набір технологічних та архітектурних рішень, що забезпечують ефективне створення кроссплатформенного месенджера.

Мета дослідження: розробка корпоративного месенджера для підприємства «Dnipro-M», що забезпечує швидкий, безпечний та зручний спосіб комунікації між співробітниками.

Розглянуто, що веб-застосунки є інтегрованими системами, які поєднують різні методи візуалізації та взаємодії з інформацією. У процесі дослідження технологій створення веб-застосунків було сформовано сценарій, який описує етапи розробки корпоративного месенджера для внутрішньої комунікації співробітників компанії «Dnipro-M».

Особливу увагу приділено дослідженню популярних платформ розробки мобільних та веб-застосунків. Також розглянуто ключові особливості мов програмування та їхніх фреймворків, що забезпечують створення надійних мультимедійних застосунків.

У межах роботи було описано етапи розробки веб-застосунку для інтерактивного спілкування співробітників, який реалізовано за допомогою фреймворку Django. Крім того, було проведено тестування працездатності системи, що підтвердило її відповідність вимогам користувачів.

Ключові слова: YII, DJANGO FRAMEWORK, LARAVEL, SPRING TA RUBY ON RAILS, ФРЕЙМВОРК, PYTHON, PHP, JAVA, ВЕБ-ЗАСТОСУНОК, СПІЛКУВАННЯ.

ANNOTATION

Thesis on the topic: Development of a cross-platform messenger for internal communication of employees at «Dnipro-M».

Executed by student: Igor Vitaliyovych Kaistriukov

The thesis consists of: 99 pages of printed text, 37 figures, 5 tables, and includes 33 sources of information used.

Object of study: development of a corporate messenger for internal communication. Subject of study: a set of technological and architectural solutions that ensure the efficient creation of a cross-platform messenger.

Purpose of the study: development of a corporate messenger for the «Dnipro-M» enterprise that provides a fast, secure, and convenient way of communication between employees.

It is considered that web applications are integrated systems that combine various methods of visualization and interaction with information. During the study of web application development technologies, a scenario was formed describing the stages of developing a corporate messenger for internal communication of «Dnipro-M» company employees.

Particular attention was paid to the study of popular platforms for mobile and web application development. The key features of programming languages and their frameworks, which ensure the creation of reliable multimedia applications, are also considered.

Within the framework of the work, the stages of developing a web application for interactive communication of employees, implemented using the Django framework, were described. In addition, the system's performance was tested, confirming its compliance with user requirements.

Key words: YII, DJANGO FRAMEWORK, LARAVEL, SPRING AND RUBY ON RAILS, FRAMEWORK, PYTHON, PHP, JAVA, WEB APPLICATION, COMMUNICATION.

ЗМІСТ

ВСТУП.....	8
РОЗДІЛ 1. СИСТЕМНИЙ АНАЛІЗ ПРОЄКТУВАННЯ ДОДАТКІВ КОРПОРАТИВНОГО СПІЛКУВАННЯ	11
1.1. Класифікація додатків для корпоративного спілкування	11
1.2. Найпоширеніші області застосування додатків корпоративного спілкування	14
1.3. Аналіз платформ для створення додатків корпоративного спілкування	15
РОЗДІЛ 2. ТЕХНІЧНЕ ЗАВДАННЯ СТВОРЕННЯ ДОДАТКУ ДЛЯ ВНУТРІШНЬОЇ КОМУНІКАЦІЇ СПІВРОБІТНИКІВ ПІДПРИЄМСТВА «DNIPRO-M»	20
2.1. Особливості мов програмування та фреймворків для створення додатків корпоративного спілкування (Python, PHP, Java, Ruby).....	20
2.2. Функціонал фреймворків.....	26
РОЗДІЛ 3. ЕТАПИ РОЗРОБКИ ДОДАТКУ КОРПОРАТИВНОГО СПІЛКУВАННЯ ЗА ДОПОМОГОЮ ФРЕЙМВОРКУ DJANGO	42
3.1. Опис функціональності розробки.....	42
3.2. Логіка роботи додатку	46
3.3. Розробка програмних модулів додатку	58
3.4. Візуалізація роботи створеного застосунку	74
РОЗДІЛ 4. ОХОРОНА ПРАЦІ	84
4.1. Загальний стан охорони праці на підприємстві «Dnipro-M».....	84
4.2. Навчання з користування засобами пожежогасіння та першої медичної допомоги.....	86
4.3. Питання навчання та інструктажу персоналу «Dnipro-M».....	87
4.4. Психофізіологічний захист працівників є ключовим аспектом політики охорони праці в компанії «Dnipro-M»	88

4.5. Запобігання професійному травматизму компанії «Dnipro-M»	90
4.6. Правила поводження з електроінструментом в торгових точках компанії Dnipro-M	91
ВИСНОВКИ	94
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	97
ДОДАТКИ.....	100

ВСТУП

У сучасному світі розробка програмних рішень для корпоративної комунікації стає все більш актуальною, оскільки підприємства прагнуть підвищити ефективність внутрішньої взаємодії співробітників. Зокрема, розвиток месенджерів як ключових засобів комунікації суттєво змінив підхід до організації робочого процесу. Швидке розповсюдження цифрових технологій сприяє зростанню кількості програмних рішень, орієнтованих на корпоративне спілкування, що дозволяє компаніям забезпечувати гнучкість та оперативність у прийнятті рішень.

Основною перевагою сучасних хмарних месенджерів є їхня доступність для користувачів незалежно від пристрою чи операційної системи. Вони дозволяють здійснювати комунікацію як у режимі реального часу, так і в автономному режимі, що суттєво підвищує їхню функціональність. Такі месенджери стають важливими інструментами не лише для обміну текстовими повідомленнями, а й для спільної роботи над проєктами, передачі файлів, відеозв'язку та інтеграції з іншими корпоративними сервісами.

Актуальність дослідження. У сучасних умовах цифрової трансформації підприємств комунікаційні технології відіграють ключову роль у підвищенні продуктивності співробітників та оптимізації внутрішніх процесів. Використання веб-технологій у корпоративних месенджерах дозволяє поєднувати різні типи даних, такі як текст, аудіо, відео та документи, забезпечуючи ефективну взаємодію між співробітниками. Завдяки інтеграції з корпоративними системами (наприклад, CRM, ERP), такі рішення стають невід'ємною частиною цифрової інфраструктури підприємства.

Попри наявність великої кількості месенджерів на ринку, більшість із них не адаптовані до специфічних потреб підприємств, зокрема до безпечної та ефективної інтеграції з внутрішніми системами компанії. Тому розробка корпоративного месенджера, орієнтованого на потреби підприємства «Dnipro-M», є актуальним завданням, що дозволяє підвищити рівень інформаційної

безпеки, контролю за комунікаціями та покращити загальну продуктивність співробітників.

Об'єкт дослідження: розробка корпоративного месенджера для внутрішньої комунікації.

Предмет дослідження: набір технологічних та архітектурних рішень, що забезпечують ефективне створення кросплатформенного месенджера.

Мета дослідження: розробка корпоративного месенджера для підприємства «Dnipro-M», що забезпечує швидкий, безпечний та зручний спосіб комунікації між співробітниками.

Завдання дослідження:

1. Проаналізувати існуючі рішення для корпоративної комунікації та визначити їхні переваги й недоліки.
2. Дослідити сучасні технології розробки кросплатформенних застосунків, зокрема для створення месенджерів.
3. Розглянути протоколи обміну повідомленнями та технології шифрування, що забезпечують безпеку корпоративної комунікації.
4. Вибрати оптимальну архітектуру для розробки месенджера, включаючи серверну частину та клієнтські застосунки.
5. Розробити прототип месенджера та протестувати його функціональність у реальних умовах.
6. Проаналізувати продуктивність та безпеку розробленого застосунку.

У ході дослідження застосовуються такі методи:

1. Аналіз літератури та наукових публікацій – дослідження теоретичних основ розробки кросплатформенних застосунків.
2. Порівняльний аналіз – оцінка існуючих месенджерів та визначення їхніх функціональних можливостей.
3. Метод проектування – розробка архітектури месенджера, вибір відповідних технологій та методів шифрування.
4. Експериментальний метод – створення та тестування прототипу месенджера.

5. Методи безпеки інформації – оцінка захищеності переданих даних.

Наукова новизна дослідження полягає у створенні кросплатформного корпоративного месенджера з підвищеним рівнем безпеки, що оптимізований для потреб підприємства «Dnipro-M». Унікальність рішення полягає в інтеграції з внутрішніми сервісами компанії та використанні сучасних методів шифрування та зберігання даних.

Практична цінність дослідження полягає у можливості використання розробленого месенджера у реальних умовах корпоративної комунікації, що сприятиме підвищенню ефективності роботи співробітників та покращенню внутрішніх бізнес-процесів.

Джерелами інформації для дослідження є наукові праці, монографії, періодичні видання, технічна документація та аналіз сучасних програмних рішень у сфері корпоративної комунікації.

Структура роботи. Відповідно до мети і завдань дослідження структура дипломна роботи складається із вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. Зміст дослідження викладено на 96 сторінках машинописного тексту. Наукова робота містить 37 рисунків та 5 таблиць. За час роботи опрацьовано 34 літературних джерела.

РОЗДІЛ 1. СИСТЕМНИЙ АНАЛІЗ ПРОЄКТУВАННЯ ДОДАТКІВ КОРПОРАТИВНОГО СПІЛКУВАННЯ

1.1. Класифікація додатків для корпоративного спілкування

Сучасні технології корпоративної комунікації розвиваються з урахуванням потреб підприємств у швидкому, безпечному та зручному обміні інформацією між співробітниками. Особливо важливу роль у цьому відіграють корпоративні месенджери, які стали ключовим інструментом внутрішньої взаємодії. Завдяки гнучкості та широкому спектру можливостей вони забезпечують миттєвий зв'язок між працівниками, незалежно від їхнього місцезнаходження. Доступність на різних пристроях – комп'ютерах, планшетах і смартфонах – робить такі системи універсальним рішенням для компаній із розгалуженою структурою або віддаленими підрозділами.

Корпоративні месенджери відрізняються за способом розгортання, функціональністю, рівнем безпеки, можливістю інтеграції з бізнес-системами та підтримуваними платформами. Одні компанії віддають перевагу хмарним рішенням, що не потребують локальної інфраструктури та забезпечують доступ із будь-якого місця. Інші обирають локальні системи, які гарантують максимальний рівень контролю над даними та відповідають внутрішнім політикам безпеки. Окрім традиційного текстового обміну, корпоративні месенджери можуть підтримувати голосові дзвінки, відеоконференції, інтерактивні робочі простори та інтеграцію з різними бізнес-сервісами, такими як CRM, ERP та файлові сховища. Особливе значення має безпека, оскільки внутрішні комунікації часто містять конфіденційну інформацію. Деякі системи використовують наскрізне шифрування, що унеможливорює доступ сторонніх осіб до даних, інші – корпоративні механізми шифрування, які забезпечують захист у межах організації [13].

Залежно від потреб підприємства вибір корпоративного месенджера може ґрунтуватися на таких критеріях, як швидкість комунікації, рівень інтеграції з внутрішніми інформаційними системами, вимоги до безпеки та масштабованість

рішення. Для компанії Dnipro-M, що має широку мережу магазинів та сервісних центрів, ключовими вимогами є кросплатформенність, стабільність роботи навіть при великій кількості користувачів, безпечність передавання даних і можливість інтеграції з внутрішніми бізнес-процесами. Розробка власного корпоративного месенджера дозволить оптимізувати взаємодію між співробітниками, підвищити ефективність виконання завдань і забезпечити централізоване управління комунікацією.

Таким чином, впровадження спеціалізованого комунікаційного рішення для Dnipro-M не лише покращить зв'язок між співробітниками, а й сприятиме більшій узгодженості всередині компанії, що особливо важливо в умовах масштабного бізнесу з динамічними процесами.

Таблиця 1.1. Основні поняття підвидів мультимедіа-додатків [10]

Категорія	Опис
Анімація	По-кадрова анімація створюється шляхом зміни серії зображень, що відтворюються у швидкій послідовності. Програмна анімація використовує алгоритми для динамічного керування рухом об'єктів, змінюючи їхні параметри, такі як розмір, напрямок або швидкість.
Додатки для web	Банери слугують рекламним інструментом, привертаючи увагу користувачів та перенаправляючи їх на цільові сторінки. Веб-додатки для обміну інформацією, наприклад, гостьові книги, дозволяють користувачам залишати коментарі або відгуки на сайтах.
Ігри	Розважальні ігри спрямовані на дозвілля та можуть мати різні жанри, такі як головоломки, стратегії чи гонки. Навчальні ігри поєднують розваги з освітнім процесом, допомагаючи користувачам засвоювати знання у цікавій інтерактивній формі.
Мультимедіа-галереї	Кадрова зміна зображень використовується у слайд-шоу чи анімаційних ефектах. Панорамні зображення дають змогу переглядати широкоформатні сцени. Інтерактивні галереї дозволяють користувачам керувати зображеннями, переглядати їх у деталях та взаємодіяти з контентом.

Розробка корпоративного месенджера для Dnipro-M передбачає поетапне створення ключових компонентів, які визначають його функціональність, зручність використання та інтеграцію в робоче середовище компанії. Визначення концепції та загальної архітектури є одним із перших і найважливіших кроків. На цьому етапі розробляється структура застосунку, що охоплює розміщення основних елементів інтерфейсу, логіку навігації та принципи взаємодії користувачів. Використання модульної багат шарової структури сприяє розділенню функціональних блоків, що підвищує гнучкість і адаптивність системи [32, с. 10–26].

Динамічність месенджера досягається через інтеграцію змінних і розробку скриптів, що регулюють обробку повідомлень, взаємодію користувачів і доступ до додаткових функцій. Використання мультимедійних елементів, зокрема звукових повідомлень, графічних символів і текстових індикаторів статусу, підвищує інтерактивність платформи. Анімаційні ефекти та візуальні підказки сприяють інтуїтивному використанню системи, що особливо важливо в умовах швидкої корпоративної комунікації.

Функціональні можливості месенджера значно розширюються завдяки інтеграції зовнішніх бібліотек і сервісів. Це дозволяє реалізувати захищене шифрування повідомлень, оперативний обмін файлами та синхронізацію з іншими корпоративними платформами. Навігаційна система забезпечує зручний доступ до групових чатів, управління історією повідомлень і налаштування прав користувачів. Використання сучасних мов програмування та веб-технологій визначає загальну архітектуру месенджера, що гарантує його стабільність, масштабованість і безпеку корпоративного зв'язку в межах компанії Dnipro-M [32, с. 10–26].

1.2. Найпоширеніші області застосування додатків корпоративного спілкування

Корпоративні месенджери відіграють ключову роль у внутрішній комунікації сучасних підприємств, забезпечуючи ефективний обмін інформацією, координацію процесів та зручність командної роботи. Для компанії Dnipro-M, яка має широку мережу магазинів, сервісних центрів та логістичних складів, наявність централізованої системи зв'язку між співробітниками є стратегічно важливим елементом бізнес-операцій. Завдяки корпоративному месенджеру можна оперативно передавати важливі робочі повідомлення, узгоджувати дії між відділами та забезпечувати швидке реагування на запити клієнтів [6, с. 103–105].

У сфері управління персоналом такі системи застосовуються для організації внутрішніх нарад, проведення тренінгів та оперативного інформування працівників про нові корпоративні ініціативи. Зокрема, корпоративний месенджер Dnipro-M може бути інтегрований із системою навчання співробітників, що дозволить централізовано проводити інструктажі, підвищення кваліфікації та тестування знань персоналу щодо продукції та стандартів обслуговування клієнтів.

У маркетинговій діяльності месенджери слугують ефективним засобом внутрішньої координації рекламних кампаній, узгодження акцій та контролю ефективності маркетингових заходів у всіх торгових точках. Оперативний зв'язок між маркетинговим відділом та адміністраторами магазинів дає змогу швидко впроваджувати нові рекламні стратегії та адаптувати їх залежно від зворотного зв'язку від клієнтів.

Логістичний сектор компанії також виграє від впровадження корпоративного месенджера, адже він дозволяє контролювати постачання товарів, взаємодіяти зі складами та відстежувати рух продукції в режимі реального часу. Це особливо важливо для Dnipro-M, оскільки забезпечення

наявності необхідного асортименту в кожному магазині є критичним фактором успішної торгівлі.

У сфері технічного обслуговування корпоративний месенджер може використовуватися для швидкого спілкування між сервісними інженерами, координації ремонтних запитів та отримання консультацій щодо складних технічних випадків. Інтеграція з базами даних сервісного центру дозволить співробітникам отримувати доступ до історії обслуговування клієнтів, що значно підвищить якість надання послуг. Сучасні технології корпоративного спілкування включають розширений функціонал, зокрема відеоконференції, обмін цифровими документами, інтеграцію з CRM-системами та автоматизованими бізнес-процесами. Усе це сприяє підвищенню продуктивності працівників, оптимізації робочих процесів та ефективному управлінню корпоративними ресурсами [6, с. 108]. Впровадження спеціалізованого месенджера в Dnipro-M дозволить об'єднати всі рівні комунікації компанії, забезпечуючи безперервний зв'язок між співробітниками та підвищуючи загальну ефективність роботи.

1.3. Аналіз платформ для створення додатків корпоративного спілкування

При розробці корпоративного месенджера для Dnipro-M критично важливим є вибір платформи, оскільки сумісність програмного забезпечення з різними операційними системами та пристроями визначає його функціональність і зручність використання. Кожне середовище розробки пропонує власний набір інструментів для створення, тестування та розгортання застосунку, що впливає на його адаптацію до різних корпоративних потреб.

Оскільки різні операційні системи та мобільні пристрої мають взаємну несумісність, традиційний підхід до розробки нативних додатків для кожної платформи може бути ресурсозатратним. Для уникнення цього обмеження все більшої популярності набувають кросплатформені технології, які дозволяють створювати універсальні програмні рішення, сумісні з Windows, macOS, Linux,

Android та iOS. Це особливо актуально для Dnipro-M, де комунікаційна система має працювати безперебійно як у магазинах, так і в логістичних центрах, сервісних пунктах та офісах компанії.

З розвитком мобільних пристроїв та широким використанням комп'ютерів у корпоративному середовищі виникла необхідність у створенні універсальних інструментів, що спрощують розгортання програмного забезпечення на різних пристроях. Використання кросплатформених фреймворків, таких як Flutter, React Native, Electron, дозволяє реалізувати корпоративний месенджер з єдиним кодовим базисом для всіх платформ. Це суттєво знижує витрати на розробку, пришвидшує впровадження системи та спрощує її оновлення [21, с. 158–162].

Вибір підходу до розробки корпоративного месенджера визначається не лише технічними можливостями, але й реальними бізнес-потребами компанії. Для Dnipro-M важливою є інтеграція месенджера з корпоративною інфраструктурою, включаючи внутрішні ERP- та CRM-системи, системи управління складом та сервісною підтримкою клієнтів. Також враховуються вимоги до захисту конфіденційної інформації та забезпечення стабільної роботи на всіх пристроях, якими користуються співробітники.

Таким чином, корпоративний месенджер Dnipro-M має базуватися на сучасних кросплатформених рішеннях, що дозволять забезпечити високу гнучкість, масштабованість та безпеку системи, при цьому знижуючи витрати на розробку та підтримку.

Таблиця 1.2. Порівняльна характеристика платформ для розробки мобільних додатків

Платформа	Наявність емулятора	Міжплатформенне розгортання	Вартість засобів розробки	Мова програмування	Інструменти розробки	Популярність та особливості
Flutter	Вбудований у SDK	iOS, Android, Web, Windows, macOS	Безкоштовно	Dart	Android Studio, Visual Studio Code	Одна з найшвидших платформ для розробки кросплатформених застосунків

Продовження таблиці 1.2.

React Native	Вбудований у Expo, Android Emulator, Xcode	iOS, Android, Web	Безкоштовно	JavaScript, TypeScript	Visual Studio Code, Expo, React Native CLI	Популярна платформа для створення мобільних застосунків із великими можливостями інтеграції з веб-технологіями.
Swift (iOS)	Вбудований у Xcode	Лише iOS та iPadOS	Безкоштовно (тільки на macOS)	Swift	Xcode	Офіційна мова розробки Apple, забезпечує високу продуктивність та глибоку інтеграцію в екосистему Apple.
Kotlin Multiplatform	Вбудований у Android Studio	Android, iOS, Web, Desktop	Безкоштовно	Kotlin	Android Studio, IntelliJ IDEA	Використовується Google для Android, а також дозволяє розробку під iOS та Web.
Xamarin (.NET MAUI)	Вбудований у Visual Studio	iOS, Android, Windows, macOS	Платний (Visual Studio Enterprise) / безкоштовний (Community)	C#	Visual Studio	Потужна кросплатформена технологія від Microsoft для мобільних та десктопних застосунків.
Electron	Вбудований у Node.js	Windows, macOS, Linux	Безкоштовно	JavaScript, HTML, CSS	Visual Studio Code, Atom	Використовується для створення кросплатформових десктопних застосунків із використанням веб-технологій.
Unity (для мобільних ігор)	Вбудований у Unity Editor	iOS, Android, Windows, macOS	Безкоштовний базовий тариф (Pro – платний)	C#	Unity Editor, Visual Studio	Найпопулярніший движок для мобільних ігор із потужною графікою та мультиплатформним розгортанням.

Таблиця містить сучасні платформи для розробки мобільних застосунків, орієнтовані на багатоплатформені рішення. Flutter, React Native та Kotlin Multiplatform забезпечують ефективну розробку для iOS, Android, Web та десктопних ОС, що ідеально підходить для корпоративного месенджера Dnipro-M. Swift та Xamarin залишаються актуальними для нативної розробки під iOS та багатоплатформних C#-застосунків відповідно. Electron використовується для створення кросплатформових десктопних застосунків, а Unity залишається основним інструментом для мобільних ігор. Таблиця демонструє актуальні технології, їх особливості та можливості інтеграції у бізнес-середовище.

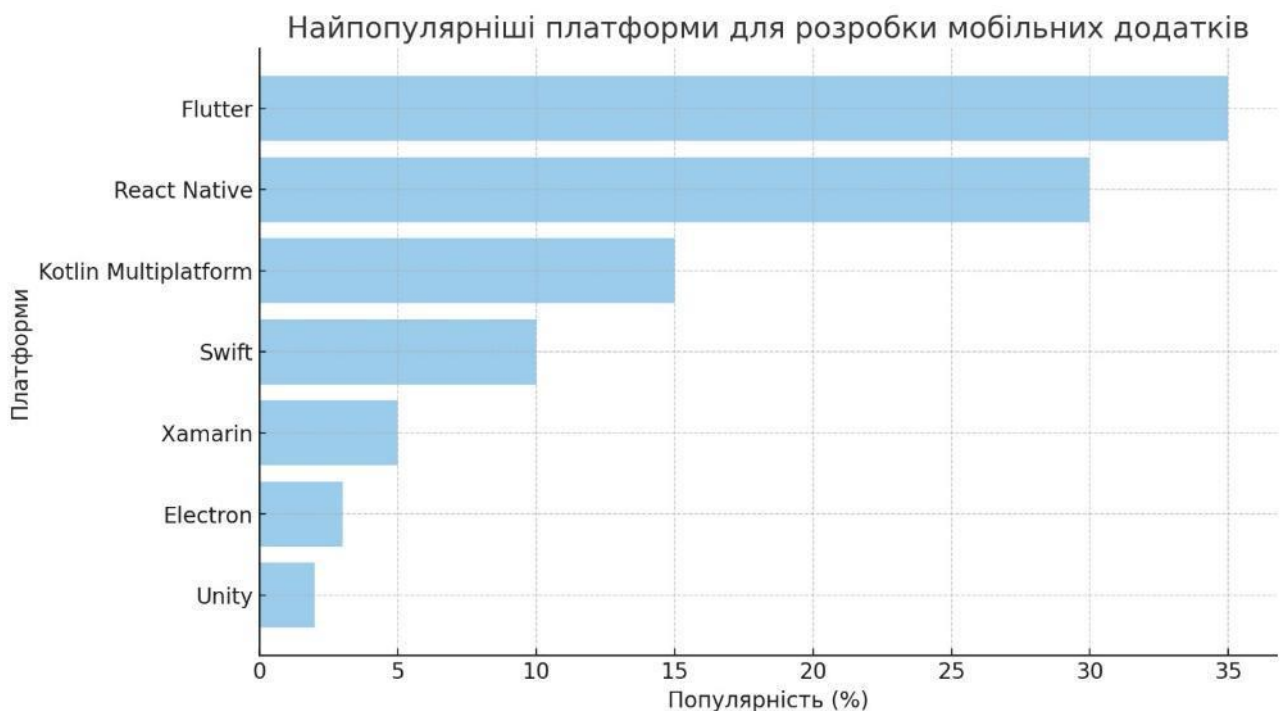


Рисунок 1.1 – Найпопулярніші платформи для розробки мобільних додатків
[31]

Додатково до таблиці, створено інфографіку (рис. 1.1), яка сприяє кращому усвідомленню дослідженої інформації та полегшує її використання у майбутньому. Діаграма відображає найпопулярніші платформи для розробки мобільних додатків, ґрунтуючись на їхній ринковій частці. Flutter (35%) займає перше місце завдяки своїй гнучкості та високій продуктивності, що робить його ідеальним вибором для створення кросплатформених застосунків. React Native

(30%) також широко використовується завдяки підтримці JavaScript та інтеграції з екосистемою веб-розробки.

Kotlin Multiplatform (15%) забезпечує ефективну розробку для Android та iOS, особливо в проєктах, що потребують нативної продуктивності. Swift (10%) залишається основною мовою для створення застосунків під iOS, забезпечуючи високу швидкість і безпеку коду. Xamarin (.NET MAUI) (5%) використовується для багатоплатформної розробки на базі C#, але поступово втрачає популярність через появу ефективніших рішень.

РОЗДІЛ 2. ТЕХНІЧНЕ ЗАВДАННЯ СТВОРЕННЯ ДОДАТКУ ДЛЯ ВНУТРІШНЬОЇ КОМУНІКАЦІЇ СПІВРОБІТНИКІВ ПІДПРИЄМСТВА «DNIPRO-M»

2.1. Особливості мов програмування та фреймворків для створення додатків корпоративного спілкування (Python, PHP, Java, Ruby)

Розробка корпоративного месенджера для внутрішньої комунікації співробітників підприємства «Dnipro-M» потребує ретельного вибору мов програмування та фреймворків, які забезпечать високу продуктивність, безпеку та можливість масштабування. Важливим аспектом є підтримка кросплатформенності, що дозволяє інтегрувати систему на мобільних пристроях, десктопних клієнтах та веб-інтерфейсі.

Python: Гнучкість і безпека серверних рішень. Однією з найпопулярніших мов програмування для розробки серверних рішень є Python, який відзначається простотою синтаксису, швидкістю розробки та потужними можливостями інтеграції. Фреймворки Django та Flask дозволяють створювати масштабовані та безпечні веб-додатки, що критично важливо для корпоративної комунікації. Використання Python також забезпечує:

1. Автоматичне управління пам'яттю, що зменшує ризик збоїв у роботі серверної частини;
2. Вбудовані бібліотеки для шифрування та захисту інформації;
3. Зручність роботи з базами даних, зокрема PostgreSQL, MySQL, SQLite;
4. Інтеграцію з AI-модулями для автоматизації комунікації, наприклад, через чат-боти.

PHP: Взаємодія з базами даних та веб-інтерфейс. Попри розвиток сучасних мов, PHP залишається затребуваним у сфері корпоративних веб-додатків завдяки фреймворкам Laravel та Symfony. Використання PHP у розробці месенджера дозволяє:

1. Забезпечити ефективну роботу з базами даних через ORM (Object-Relational Mapping);

2. Швидко реалізувати веб-інтерфейс завдяки готовим бібліотекам;
3. Оптимізувати серверні навантаження за допомогою кешування та обробки запитів.

Java: Надійність та масштабованість. Java є одним із провідних виборів для створення корпоративних систем, що потребують високої безпеки та стабільності. Завдяки Spring Framework месенджер для «Dnipro-M» може мати такі переваги:

1. Гнучка серверна архітектура, що дозволяє інтегрувати корпоративні сервіси;
2. Підтримка мікросервісної архітектури, що спрощує масштабування;
3. Захищені API для зв'язку з іншими бізнес-додатками.

Ruby: Ефективність у швидкому прототипуванні. Ruby у поєднанні з фреймворком Ruby on Rails дозволяє реалізовувати швидку розробку веб-додатків завдяки вбудованим шаблонам та автоматизації процесів. Використання Ruby у месенджері сприяє:

1. Швидкому створенню MVP-продукту завдяки простому синтаксису;
2. Автоматизованій інтеграції з API інших корпоративних сервісів;
3. Зручному управлінню базами даних через Active Record ORM.

З урахуванням особливостей кожної мови, для створення корпоративного месенджера доцільно використовувати Python (Django) або Java (Spring) для серверної частини, PHP (Laravel) для веб-інтерфейсу та Ruby on Rails для швидкого розгортання нових функцій. Використання цих технологій забезпечить захищеність даних, стабільність роботи та можливість масштабування, що є ключовими вимогами корпоративної комунікації.

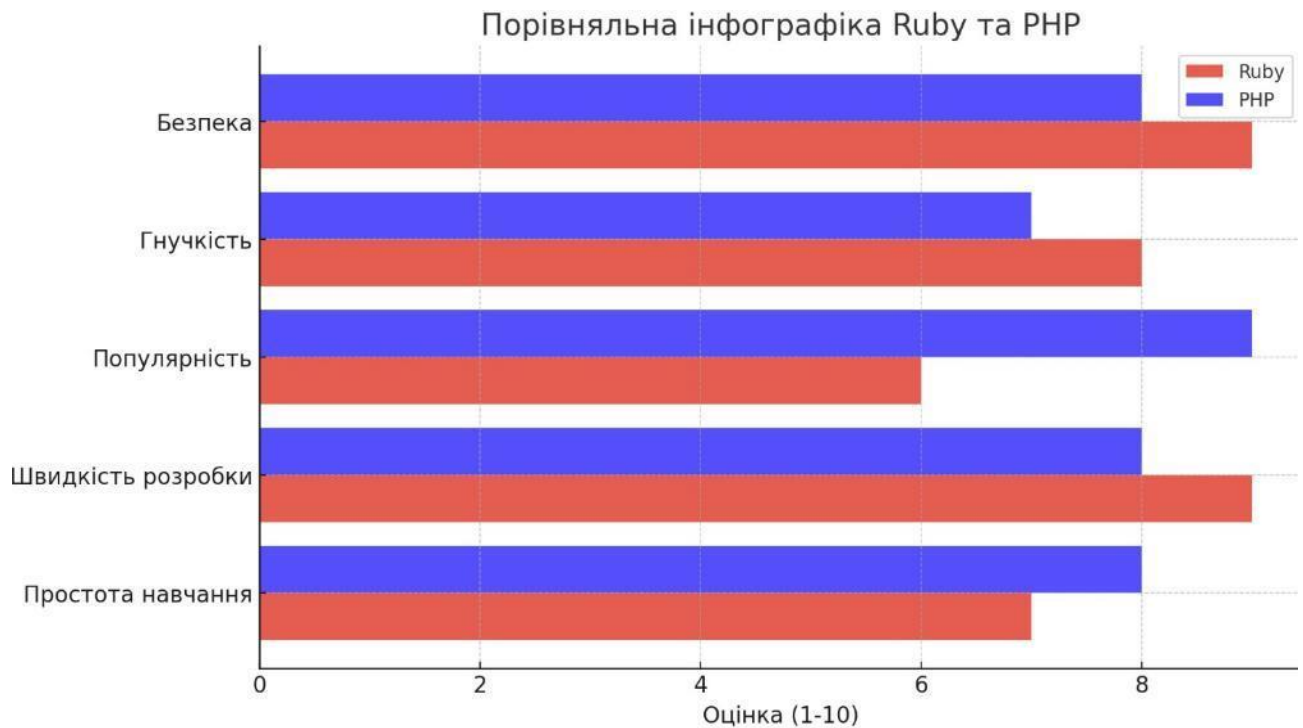


Рисунок 2.1 – Порівняння Ruby та PHP [34]

На рисунку представлено порівняльний аналіз мов програмування Ruby та PHP за п'ятьма критеріями: простота навчання, швидкість розробки, популярність, гнучкість та безпека.

1. Простота навчання: PHP має трохи вищу оцінку, оскільки є більш знайомим для початківців і широко використовується у веб-розробці.
2. Швидкість розробки: Ruby випереджає завдяки своїм фреймворкам, зокрема Ruby on Rails, які спрощують розробку додатків.
3. Популярність: PHP залишається популярнішим завдяки широкому використанню в веб-розробці та підтримці великої кількості хостингів.
4. Гнучкість: Ruby відзначається кращою гнучкістю завдяки своєму об'єктно-орієнтованому підходу.
5. Безпека: Обидві мови мають хороші механізми захисту, проте Ruby вважається більш безпечним через суворіші підходи до кодування.

Фреймворки є важливим інструментом для розробки корпоративних додатків, оскільки забезпечують стандартизовану архітектуру, спрощують розробку та зменшують ризик помилок. Вони надають набір готових модулів і

компонентів, що дозволяють розробникам зосередитися на бізнес-логіці, а не на низькорівневих аспектах програмування.

На відміну від бібліотек, які є лише наборами функцій, фреймворки визначають загальну структуру проєкту, задаючи правила та стандарти для його побудови. Вони дозволяють організувати взаємодію між компонентами програми, спрощуючи підтримку та масштабування. Це особливо важливо для корпоративних месенджерів, які мають підтримувати високу продуктивність, безпеку та інтеграцію з іншими системами.

Фреймворки поділяються на серверні та клієнтські. Серверні, такі як Django (Python) або Laravel (PHP), відповідають за обробку запитів, зберігання даних і керування користувачами. Клієнтські, такі як React або Vue.js, дозволяють створювати динамічний інтерфейс, який забезпечує зручний користувацький досвід. У випадку корпоративного месенджера для «Dnipro-M» необхідно поєднати обидва типи фреймворків для створення інтегрованого рішення.

Використання фреймворків значно скорочує час розробки, оскільки вони містять вбудовані інструменти для автентифікації, роботи з базами даних, маршрутизації запитів та обробки подій. Це дозволяє зосередитися на розробці унікального функціоналу без необхідності створення базових механізмів з нуля.

Переваги фреймворків також включають підтримку спільноти розробників, що забезпечує швидке оновлення, виправлення помилок та постійне вдосконалення. Використання стандартних рішень сприяє сумісності додатків з іншими корпоративними системами, що є важливим для інтеграції месенджера з CRM, ERP та іншими бізнес-інструментами.

Таким чином, вибір відповідного фреймворку є ключовим етапом у проєктуванні корпоративного месенджера, оскільки він визначає майбутню архітектуру системи, її гнучкість, масштабованість та безпеку.

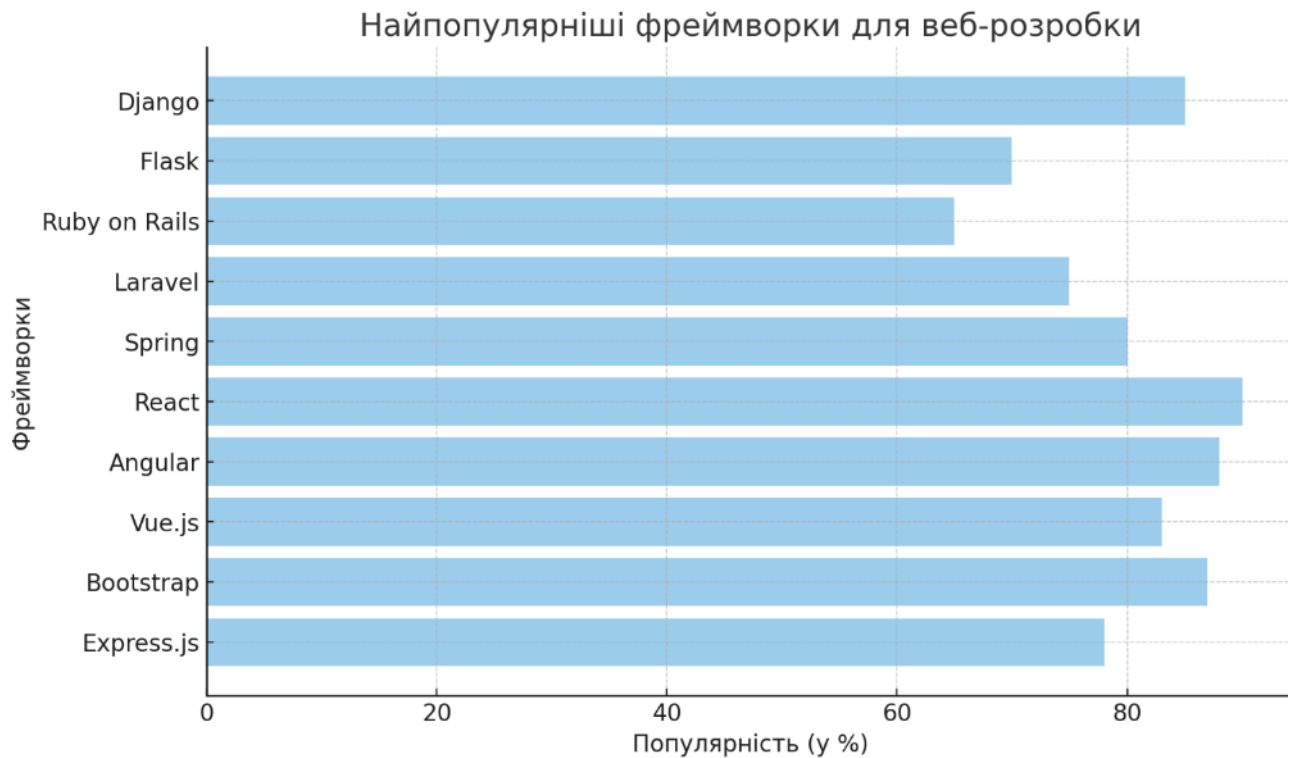


Рисунок 2.2 – Приклади найпопулярніших фреймворків [31]

Рисунок демонструє найпопулярніші фреймворки для веб-розробки. Він порівнює рівень популярності серед таких технологій, як Django, Flask, Ruby on Rails, Laravel, Spring, React, Angular, Vue.js, Bootstrap і Express.js. Діаграма показує, що React та Django є одними з найпопулярніших рішень, тоді як менш відомі фреймворки, такі як Flask або Ruby on Rails, мають нижчий рівень популярності, але все ще широко використовуються в розробці.

Розробка корпоративного месенджера для підприємства «Dnipro-M» потребує комплексного підходу до вибору технологій. Зокрема, важливим є використання фреймворків, що спрощують розробку, забезпечують ефективну організацію коду, інтеграцію з базами даних і підтримку безпеки. Фреймворки дозволяють значно скоротити час створення застосунку, спрощують співпрацю між розробниками та забезпечують повторне використання готових модулів [7].

Фреймворки поділяються за призначенням та сферою застосування. Веб-фреймворки, зокрема Django (Python), Laravel (PHP), Spring (Java) та Ruby on Rails (Ruby), є основою для розробки веб-додатків та корпоративних сервісів. Вони допомагають у створенні динамічних веб-систем, забезпечуючи інтеграцію

з базами даних, механізмами авторизації та засобами шифрування. Додатково існують фреймворки для розширення можливостей мов програмування, що застосовуються у розробці операційних систем та автоматизації робочих процесів.

Особливу роль відіграють фреймворки для роботи з асинхронним передаванням даних та інтерактивними веб-додатками. Наприклад, Flask (Python) та Express.js (JavaScript) застосовуються у поєднанні з технологією AJAX, що дозволяє обмінюватися інформацією між клієнтами без перезавантаження сторінки. Це важливо для миттєвого передавання повідомлень у месенджерах. Окремо слід виділити фреймворки для управління вмістом (CMS), які інтегруються з корпоративними порталами, та мультимедійні фреймворки, що використовуються для реалізації відеозв'язку або інтеграції медіафайлів.

Розробники часто використовують фронтенд-фреймворки (Angular, React, Vue.js), що забезпечують розробку зручного користувацького інтерфейсу для веб-додатків. Вони працюють із HTML, CSS та JavaScript і створюють адаптивні веб-інтерфейси, зручні для взаємодії користувачів. У свою чергу, бекенд-фреймворки, такі як Django (Python), Ruby on Rails (Ruby) або Laravel (PHP), забезпечують серверну логіку, управління даними та їхню безпечну обробку.

Для реалізації корпоративного месенджера важливо обрати надійну технологічну платформу. Python є одним із лідерів у цій сфері завдяки фреймворкам Django та Flask, що забезпечують високий рівень безпеки, інтеграцію з базами даних і підтримку серверних технологій. PHP, який є основою для веб-додатків, також активно використовується у корпоративному середовищі завдяки фреймворкам Laravel та Symfony. Java широко застосовується для створення масштабованих корпоративних систем та мобільних застосунків, а його основним веб-фреймворком є Spring Framework. Ruby, завдяки Ruby on Rails, забезпечує швидку розробку веб-застосунків та інтеграцію з базами даних, що робить його корисним для прототипування нових сервісів.

Фреймворки дозволяють автоматизувати процеси, спростити розробку та зменшити кількість помилок у коді. Наприклад, CherryPy (Python) використовується для створення невеликих веб-сервісів, а Pyramid (Python) – для розробки багатофункціональних додатків із розширеними можливостями безпеки. TurboGears пропонує інтегровані шаблони, ORM і засоби для швидкої реалізації корпоративних сервісів.

Таким чином, вибір фреймворку для корпоративного месенджера «Dnipro-M» визначатиметься його функціональними можливостями, рівнем безпеки та гнучкістю в інтеграції з існуючими системами підприємства. Сучасні веб-технології дозволяють створювати ефективні рішення для внутрішньої комунікації, які відповідають потребам бізнесу та гарантують надійний захист корпоративних даних.

2.2. Функціонал фреймворків

2.2.1. Особливості використання веб-фреймворку Yii та Django Framework

У сучасній розробці корпоративних веб-додатків використання фреймворків значно спрощує процес створення функціональних, масштабованих і безпечних систем. Веб-фреймворки надають базову структуру для розробки, дозволяючи зосередитися на реалізації бізнес-логіки замість написання низькорівневого коду. Вони забезпечують автоматизацію багатьох аспектів роботи з веб-запитами, управління базами даних, валідації введених даних, сесіями користувачів та механізмами безпеки.

Основна відмінність веб-фреймворку від веб-сервера полягає в тому, що сервер лише обробляє HTTP-запити та забезпечує їхнє спрямування, тоді як фреймворк надає набір інструментів для створення динамічного веб-застосунку. Це дозволяє розробникам ефективно керувати кодом, спрощує розгортання та підтримку проєкту, а також покращує безпеку веб-додатку.

Фреймворки мають широкий функціонал, який включає механізми маршрутизації запитів, обробку форм, доступ до баз даних через ORM, управління сесіями користувачів, шаблонізатори для генерації інтерфейсу та засоби для захисту від найпоширеніших кіберзагроз. Вони є невід’ємним інструментом веб-розробки, що сприяє швидкому створенню продуктивних і надійних рішень [12, с. 420–428].

Серед найпопулярніших веб-фреймворків особливу увагу варто приділити Yii та Django, які широко використовуються у розробці корпоративних додатків.

Yii є високопродуктивним PHP-фреймворком, який ідеально підходить для розробки корпоративних інформаційних систем та бізнес-платформ. Його ключовою особливістю є висока швидкість роботи, що забезпечується завдяки кешуванню, оптимізованій архітектурі та підтримці асинхронної обробки запитів.

Фреймворк надає гнучку ORM для роботи з базами даних, що дозволяє розробникам швидко виконувати операції CRUD (створення, читання, оновлення, видалення). Завдяки інтегрованим засобам авторизації та автентифікації, Yii забезпечує надійний контроль доступу. Додатково він підтримує REST API, що робить його ідеальним вибором для створення масштабованих веб-сервісів, таких як корпоративні месенджери.

Завдяки використанню генератора коду Gii, розробники можуть швидко створювати нові компоненти, що значно прискорює процес розробки. Yii підтримує шаблон MVC (Model-View-Controller), який спрощує організацію коду та підвищує його читабельність.

Django – це потужний Python-фреймворк, який дозволяє створювати масштабовані та безпечні веб-додатки. Його головною перевагою є концепція «Швидкого розгортання» (Rapid Development), яка дозволяє значно скоротити час на розробку завдяки вбудованим модулям.

Django надає автоматизоване управління базами даних через ORM, що підтримує різні СУБД (PostgreSQL, MySQL, SQLite). Система авторизації

забезпечує багаторівневий контроль доступу користувачів, а вбудовані механізми безпеки захищають від таких атак, як SQL-ін'єкції, XSS, CSRF.

Особливою перевагою Django є гнучкість у розгортанні та масштабуванні, що дозволяє використовувати його для створення корпоративних месенджерів, бізнес-аналітичних систем та сервісів управління ресурсами.

Yii та Django мають схожий набір функцій, однак їх застосування залежить від специфіки проєкту. Якщо необхідно створити продуктивний веб-додаток із мінімальними витратами ресурсів, оптимальним вибором буде Yii. Якщо ж розробник працює з Python-орієнтованими технологіями та планує реалізувати складну логіку з аналітикою та інтеграцією зі сторонніми сервісами, Django стане кращим рішенням.

Вибір фреймворку для розробки корпоративного месенджера підприємства «Dnipro-M» має базуватися на вимогах безпеки, продуктивності та інтеграції з існуючими корпоративними сервісами. Обидва фреймворки мають активну підтримку спільноти та забезпечують ефективну розробку сучасних веб-додатків.



Рисунок 2.3 – Найпопулярніші приклади PHP веб-фреймворків

Використання веб-фреймворків значно оптимізує процес створення веб-додатків та сервісів, надаючи розробникам набір готових функцій, інструментів і шаблонів. Завдяки цьому можна суттєво скоротити час розробки, покращити структуру коду та підвищити безпеку додатка. Фреймворки забезпечують зручні

засоби для роботи з базами даних, маршрутизації запитів, обробки форм, управління сесіями користувачів та інтеграції з іншими сервісами.

Вибір відповідного веб-фреймворку повинен базуватися на потребах конкретного проєкту. Важливо враховувати такі аспекти, як сумісність із мовою програмування, рівень продуктивності, безпека, наявність документації та підтримки з боку розробницької спільноти. Популярні рішення, такі як Django (Python), Ruby on Rails (Ruby), Laravel (PHP) та Spring (Java), створені для прискорення розробки веб-додатків, надання потужного функціоналу та забезпечення відповідності сучасним стандартам безпеки та продуктивності.

Залежно від типу веб-додатка, розробники можуть обирати між мікрофреймворками, які пропонують базові можливості (наприклад, Flask для Python), і повноцінними фреймворками, що містять вбудовані механізми для роботи з базами даних, аутентифікації та управління контентом. Наприклад, Django та Ruby on Rails підходять для розробки комплексних веб-додатків із розширеними можливостями адміністрування, тоді як Express.js або FastAPI орієнтовані на створення високопродуктивних API. Також важливим фактором при виборі фреймворку є його гнучкість та можливість розширення. Наприклад, Laravel має широку екосистему розширень та пакетів, що робить його ідеальним вибором для створення веб-систем, які можуть змінюватися та розширюватися відповідно до потреб бізнесу. Spring Framework у Java надає можливість розробки масштабованих корпоративних додатків із складною бізнес-логікою. Спільнота розробників відіграє ключову роль у підтримці фреймворків, забезпечуючи швидке виправлення помилок, регулярне оновлення та вдосконалення функціоналу. Наприклад, Django має велику та активну спільноту, що гарантує стабільність і безпеку фреймворку. Також популярність платформи впливає на кількість наявних навчальних матеріалів та рівень підтримки (Рис. 2.4).

Отже, використання сучасних веб-фреймворків дозволяє створювати безпечні, продуктивні та гнучкі веб-рішення, які відповідають потребам

сучасного бізнесу та забезпечують зручність у розробці та подальшій підтримці [18].

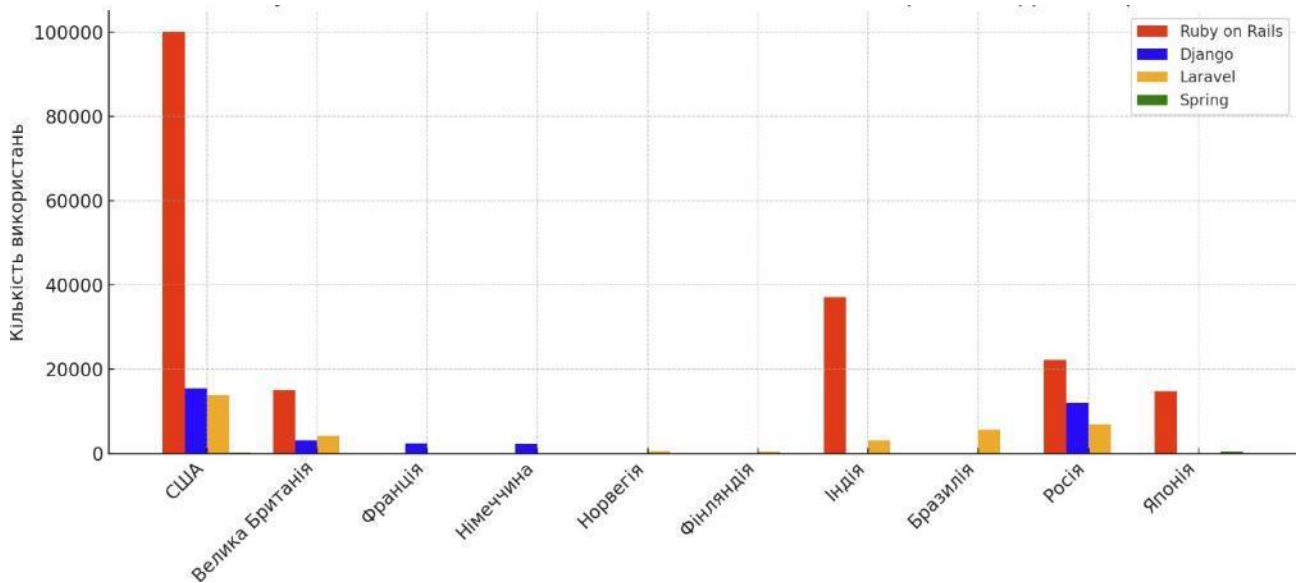


Рисунок 2.4 – Статистика від SimilarTech щодо поширеності фреймворків [18]

Рисунок ілюструє статистику від SimilarTech щодо поширеності веб-фреймворків у різних країнах. Ці дані демонструють регіональні уподобання у використанні веб-фреймворків та підкреслюють домінування Ruby on Rails у більшості країн.

Вона відображає використання чотирьох популярних фреймворків:

- Ruby on Rails (червоний) – має найбільше використання у США, Індії та Росії.
- Django (синій) – популярний у США, Великій Британії, Франції, Німеччині та Росії.
- Laravel (помаранчевий) – широко використовується в США, Великій Британії, Індії, Бразилії та Росії.
- Spring (зелений) – має незначне використання, але зустрічається в США та Японії.

Веб-фреймворки значно полегшують розробку веб-додатків, надаючи розробникам готові шаблони, структури та інструменти для створення ефективних і масштабованих рішень. Вони спрощують взаємодію з базами

даних, генерацію HTML-коду та маршрутизацію запитів, що суттєво оптимізує процес роботи.

Yii – це продуктивний PHP-фреймворк, орієнтований на швидку розробку складних веб-додатків. Він використовує компонентну структуру та забезпечує повторне використання коду, що підвищує ефективність розробки. Для його роботи потрібен сервер із підтримкою PHP 5.1.0 або вище. Головні переваги Yii – висока продуктивність, якісна документація та зручність у розширенні. Він побудований на основі об'єктно-орієнтованого підходу та містить широкий набір функціональних можливостей. Фреймворк Yii базується на шаблоні проєктування MVC, який чітко розділяє бізнес-логіку, інтерфейс і обробку запитів. Контролер управляє взаємодією між моделлю та уявленням, що дозволяє легко змінювати окремі компоненти без впливу на всю систему. У Yii також використовується фронт-контролер, який керує запитами та передає їх відповідним модулям.

Компонентна архітектура Yii дозволяє легко розширювати функціональність, підключати сторонні бібліотеки та інтегрувати систему з іншими сервісами. Його механізми кешування, аутентифікації, роботи з базами даних та підтримка AJAX забезпечують гнучкість і високу продуктивність веб-додатків. Завдяки цим можливостям Yii є ефективним інструментом для створення сучасних веб-систем (Рис. 2.5).

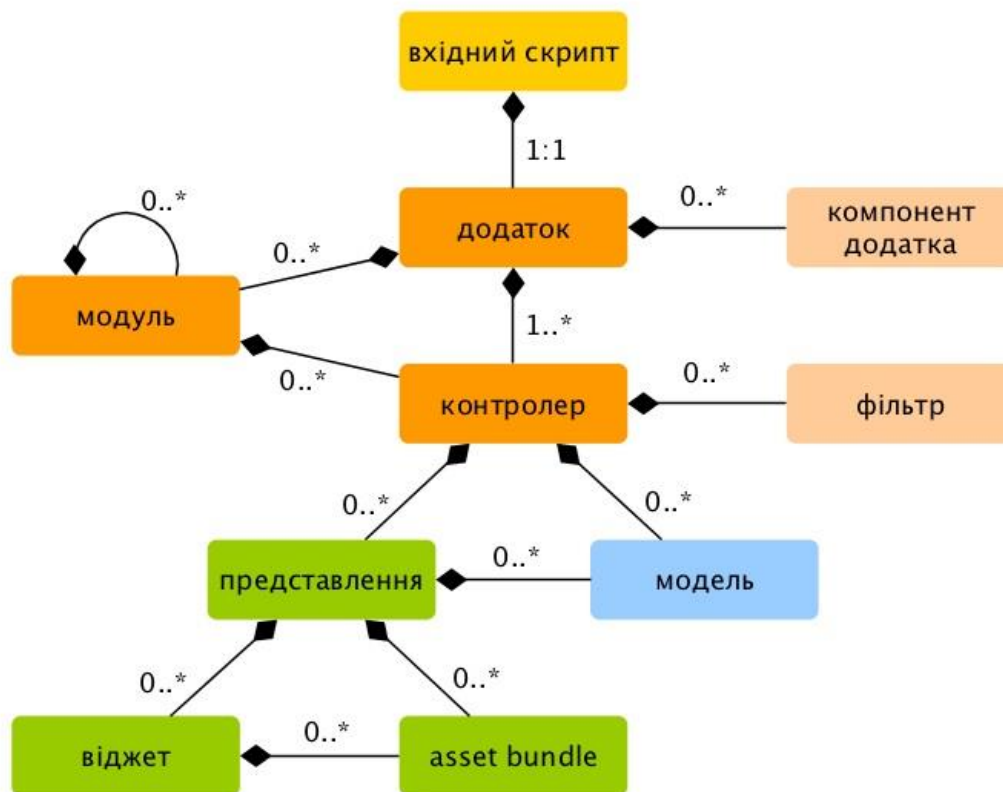


Рисунок 2.5 – Структура фреймворку Yii [7]

Yii рекомендує налаштовувати конфігурацію індивідуально для кожного члена команди, не зберігаючи конфіденційні дані, як-от паролі та API-ключі, у сховищі версій. Оскільки для розробки, тестування та продакшену використовуються різні середовища, фреймворк підтримує окремі конфігураційні файли для кожного з них. Yii дозволяє легко перемикатися між середовищами за допомогою команди, яка копіює відповідні файли з каталогу «environments» у кореневий каталог проєкту, усуваючи потребу в повторному налаштуванні параметрів [7].

Django – потужний full-stack фреймворк з відкритим кодом для веб-розробки на Python, що забезпечує швидкий перехід від прототипу до готового рішення. Він автоматизує багато процесів завдяки вбудованим функціям: панелі адміністратора, міграціям баз даних, формам і засобам аутентифікації. Django має просту структуру, високий рівень безпеки та ефективний механізм роботи з базами даних через ORM (Рис. 2.6). Використовуючи архітектуру Model-View-

Template (MVT), фреймворк дозволяє розділяти бізнес-логіку та візуальне відображення програми, забезпечуючи гнучкість у розробці [24, с. 223–228].

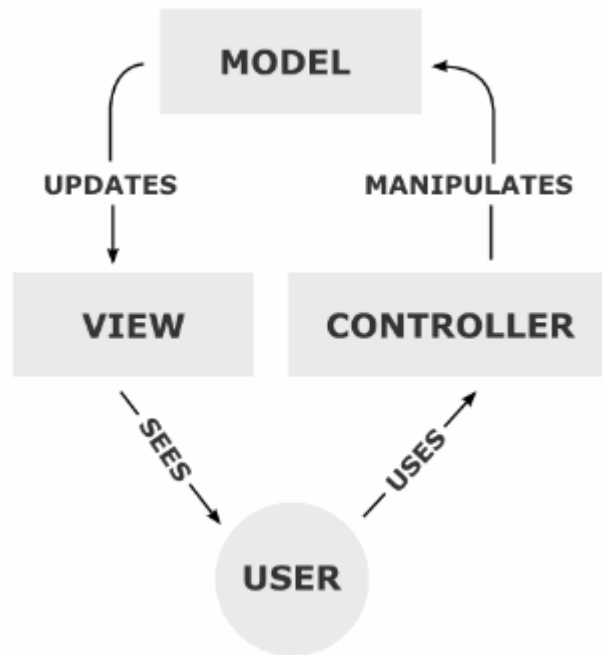


Рисунок 2.6 – Схема архітектури MVC в Django [24].

Django, як full-stack фреймворк, використовує модель MVT (Model-View-Template), що розділяє бізнес-логіку, обробку даних та відображення. Модель керує структурою даних і підтримує роботу з базами, такими як PostgreSQL, MySQL, SQLite та Oracle. Подання відповідає за обробку HTTP-запитів та взаємодію з моделлю, а шаблони забезпечують генерацію HTML-сторінок без вбудованої бізнес-логіки [24, с. 223–228].

Незважаючи на широкі можливості, Django є досить монолітним, що може сповільнювати його розвиток. Проте він чудово підходить для створення API, веб-застосунків, проєктів, що потребують частого оновлення та масштабування, а також інтеграції з сучасними технологіями, такими як машинне навчання. Завдяки ORM розробники можуть працювати з базами даних без написання складних SQL-запитів.

Серед ключових переваг Django – розвинена екосистема з численними модулями, стабільність і велика активна спільнота. Вбудована адміністративна панель спрощує керування даними, а підтримка SEO та можливість розширення

через плагіни роблять його гнучким рішенням. Django також інтегрується з популярними бібліотеками, такими як Django Rest Framework (для API), Django CMS (для управління контентом) та Django-allauth (для аутентифікації).

ORM у Django автоматизує взаємодію з базами даних, дозволяючи легко перемикаватися між ними без значних змін у коді. Це спрощує розробку та забезпечує більш читабельний і структурований підхід до роботи з даними.

2.2.2. Особливості використання веб-фреймворку Laravel

Laravel – один із найпопулярніших PHP-фреймворків, що базується на архітектурному шаблоні MVC (Model-View-Controller). Це дозволяє чітко розділяти бізнес-логіку, інтерфейс і управління даними, забезпечуючи зручну організацію коду [26]. Його головні переваги – елегантний синтаксис, потужний функціонал і велика спільнота розробників, що активно підтримує проєкт [26].

Laravel підтримує маршрутизацію, шаблонізацію, роботу з базами даних через ORM, обробку сеансів і аутентифікацію, а також має вбудовані механізми для запитів AJAX та створення API. Відкрита ліцензія MIT і доступний на GitHub код сприяють його широкому використанню [27].

Завдяки структурованому підходу Laravel (Рис. 2.7) забезпечує високу продуктивність, зручне керування макетами веб-сторінок та інтеграцію з CSS і JS-компонентами. ORM у Laravel значно спрощує роботу з базами даних, дозволяючи взаємодіяти з ними без написання складних SQL-запитів. Система безпеки забезпечує захист паролів, запобігання SQL-ін'єкціям і XSS-атакам, що робить Laravel безпечним середовищем для розробки [18].

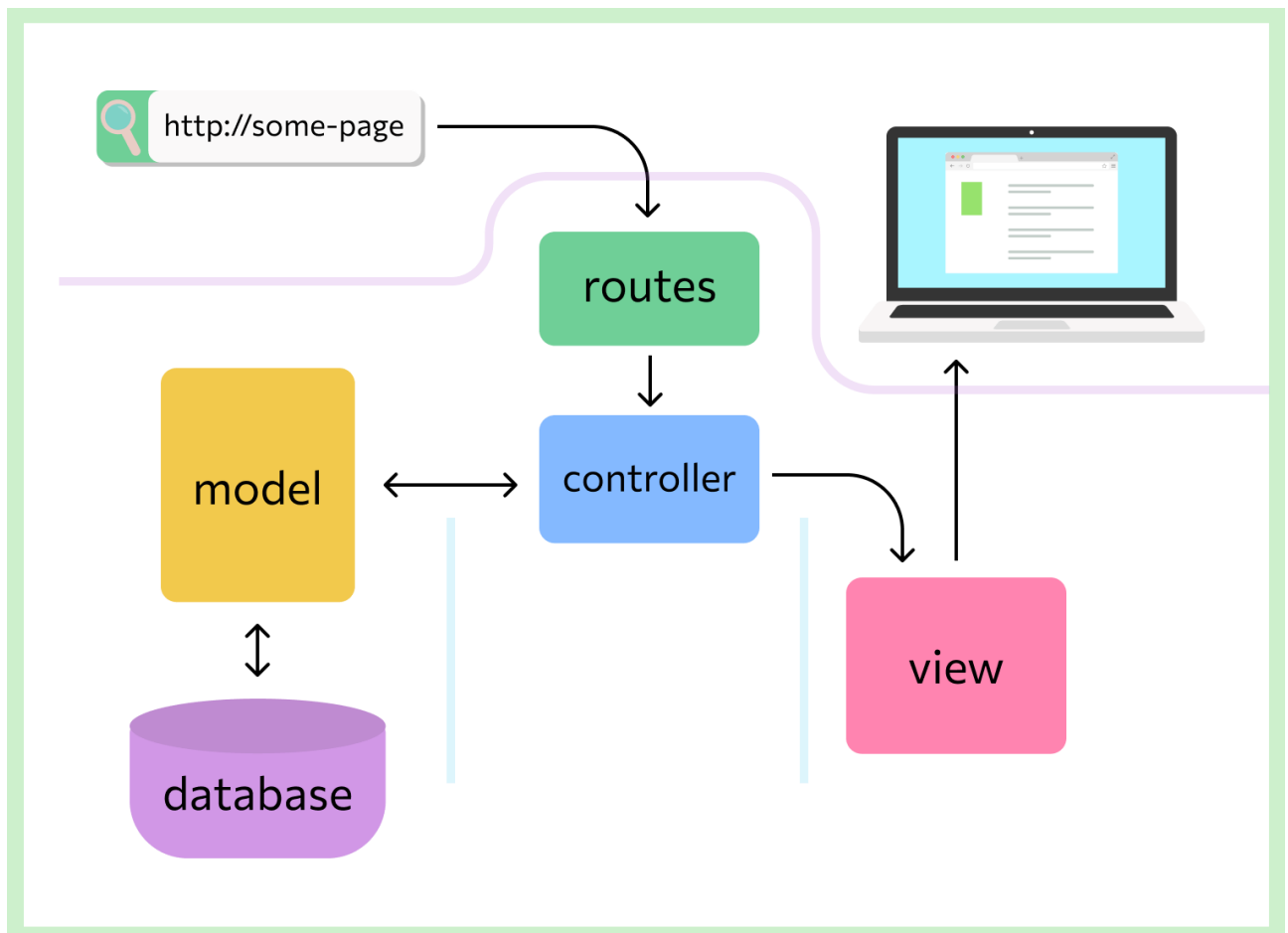


Рисунок 2.7 – Структура фреймворку Laravel [27]

Одним із ключових інструментів Laravel є Artisan – консольний інструмент для автоматизації рутинних завдань. Фреймворк також підтримує модульну структуру, дозволяючи легко додавати сторонні бібліотеки. Його механізми логування забезпечують ефективне виявлення та обробку помилок, а вбудовані засоби кешування значно покращують продуктивність веб-додатків [27].

Laravel підтримує міграції баз даних, що спрощує керування їхньою структурою без ризику втрати даних. Крім того, система тестування дозволяє проводити автоматизовані перевірки, що зменшує кількість помилок перед випуском програмного забезпечення. Завдяки цим функціям Laravel залишається одним із найкращих PHP-фреймворків для створення сучасних, безпечних і масштабованих веб-додатків [25].

2.2.3. Особливості використання веб-фреймворку Spring

Java-додатки можуть варіюватися від невеликих аплетів до складних корпоративних серверних рішень, у яких компоненти взаємодіють між собою, створюючи загальну структуру програми. Проте Java не має вбудованих засобів для організації цих компонентів, тому розробникам доводиться самостійно застосовувати патерни проєктування, такі як Factory, Builder та Decorator, щоб формувати ефективну архітектуру [23].

Таблиця 2.1 Порівняльна характеристика Spring та Yii [15], [9], [23]

Основа порівняння	Spring	Yii
Тестування	Використовує нестандартні підходи, включаючи компоненти Symfony, такі як HttpKernel, DomCrawler та інші.	Підтримує тестування за допомогою PHPUnit та Codeception, що значно спрощує процес.
Вимоги	NetBeans 8.0 або новіші версії, Java EE, JDK 8 або новіші версії.	PHP 7.4 або новіші версії, підтримка MySQL або іншої СУБД.
Кешування	Використовує механізми кешування, такі як HashMap, ConcurrentHashMap, база даних, файли.	Підтримує різні варіанти кешування: APC, база даних, файли, Memcached, WinCache, XCache.
Об'єктно-реляційне відображення (ORM)	Використовує екземпляри JDBC DataSource для взаємодії з базою даних.	Реалізує доступ до даних через DAO, шаблон ActiveRecord та підтримує Doctrine 2 через плагіни.
Міграція	Потребує класу міграції та інструментів для заповнення даних.	Використовує лише клас міграції, що спрощує процес.

Spring Framework вирішує цю проблему, пропонуючи формалізовану систему для збору компонентів у цілісний додаток. Він спрощує розробку, надаючи вбудовані шаблони, що полегшують інтеграцію компонентів. Багато компаній використовують Spring для створення масштабованих та гнучких корпоративних рішень (Рис. 2.8).

Spring підтримує існуючі технології, включаючи ORM-фреймворки та реєстраційні механізми, що спрощує тестування й забезпечує єдиний API для керування винятками. Його MVC-фреймворк є ефективною альтернативою іншим веб-рішенням, таким як Struts. Також Spring включає легкі контейнери IoC, що оптимізують використання ресурсів, та єдину систему керування транзакціями, яка може масштабуватися від локальних до глобальних операцій [8].

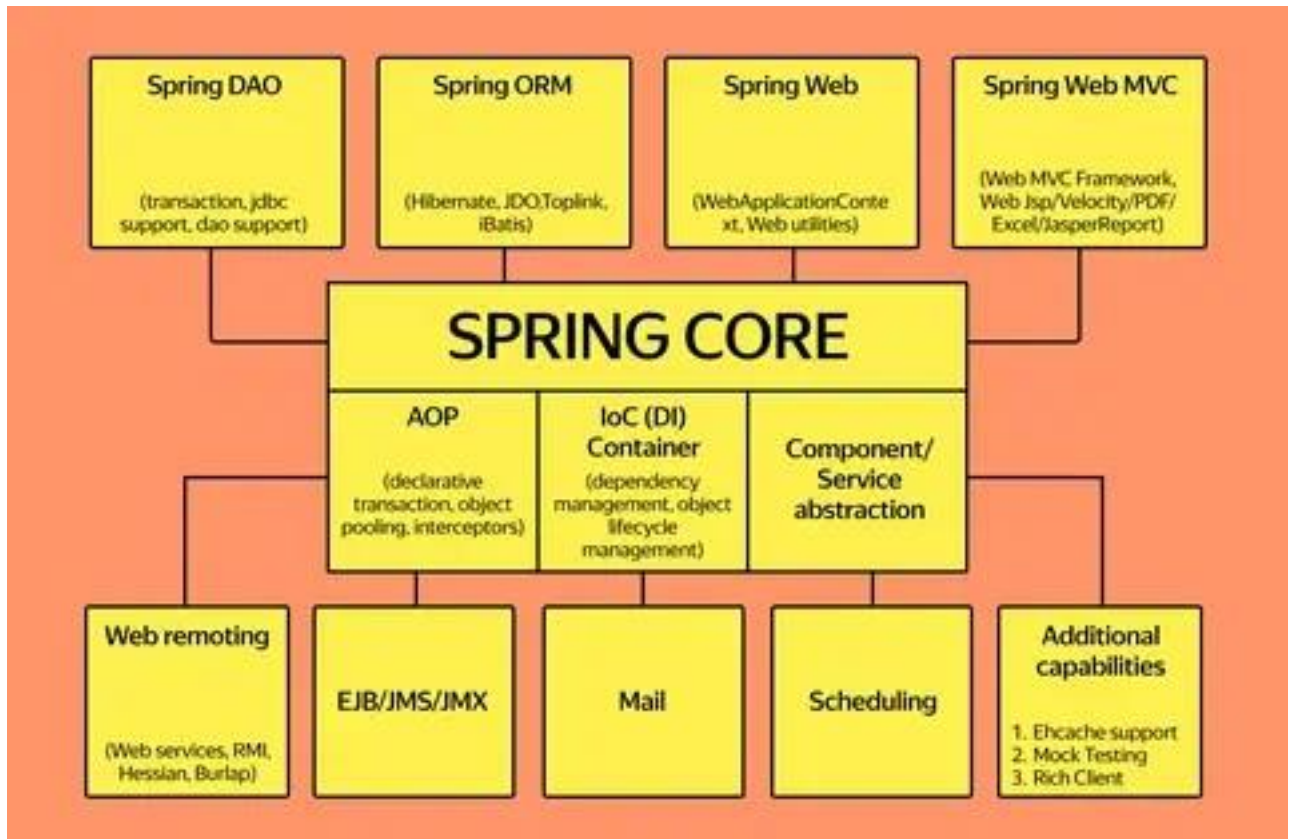


Рисунок 2.8 – Структура фреймворку Spring [22]

Важливим аспектом Spring є підтримка аспектно-орієнтованого програмування (AOP), що дозволяє ізолювати наскрізні функції, такі як безпека, кешування та логування, без зміни основного коду додатка. Крім того, механізм Dependency Injection (DI) усуває жорсткі залежності між компонентами, що робить систему більш гнучкою.

Модульна структура Spring дозволяє розробникам вибирати лише необхідні компоненти, що оптимізує продуктивність і зменшує навантаження. Це робить Spring універсальним рішенням для розробки корпоративних систем,

які потребують масштабованості, продуктивності та ефективного управління залежностями [15].

2.2.4. Особливості використання веб-фреймворку Ruby on Rails

Ruby on Rails, або просто Rails, є відкритим веб-фреймворком, розробленим мовою програмування Ruby. Його створив Девід Хайнмайер Ханссон у 2003 році, прагнучи забезпечити максимальний комфорт для розробників завдяки простоті та чистоті коду [34]. Фреймворк забезпечує ефективну структуру для розробки веб-додатків, автоматизуючи рутинні процеси, такі як створення форм, таблиць та меню. Це дозволяє значно скоротити час розробки, оскільки програмістам не потрібно реалізовувати базові компоненти з нуля [34].

Ruby on Rails (Рис. 2.9) широко використовується у створенні веб-сайтів та веб-додатків, маючи відкритий вихідний код і активну спільноту, що розробляє численні додаткові модулі. Його популярність обумовлена рядом переваг, серед яких масштабованість, безпека, висока продуктивність, тестове середовище та зручність у розробці [26].

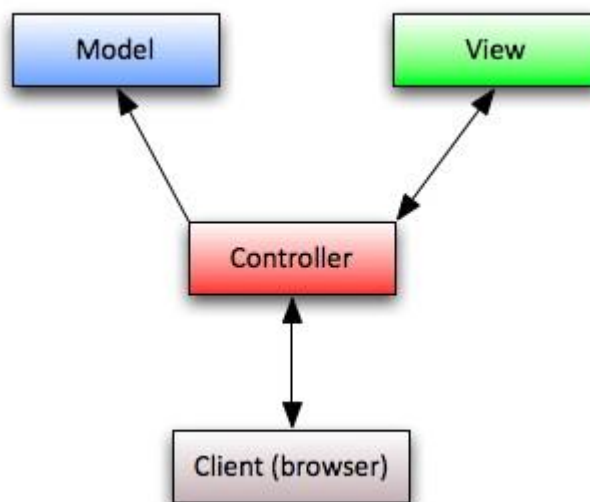


Рисунок 2.9 – Структура фреймворку Ruby on Rails [34]

Rails використовує архітектурний підхід Model-View-Controller (MVC), що розділяє додаток на три основні частини: модель, контролер і представлення, що сприяє кращій організації коду та полегшує підтримку і розширення додатків.

Таблиця 2.2. Таблиця порівняння Laravel проти Ruby on Rails [34], [26]

Основа порівняння	Laravel	Ruby on Rails
Мова програмування	PHP	Ruby
Призначення	Використовується для створення корпоративних веб-додатків, веб-сайтів та API	Оптимізований для веб-додатків із підтримкою баз даних та метапрограмування
Синтаксис	Використовує крапку з комою та фігурні дужки	Спрощений синтаксис без крапок з комою та фігурних дужок
Спільнота	Laravel має активну спільноту та платформу для навчання Laracasts	Rails має велику спільноту та ресурси, такі як RailsCasts
Рік випуску	Червень 2011 року	2004 рік
Основні особливості	Вбудований механізм автентифікації, гнучкий ORM, підтримка REST API	Простий та елегантний синтаксис, потужна підтримка MVC, великий вибір сторонніх плагінів
Об'єктно-реляційне відображення (ORM)	Використовує Eloquent ORM	Використовує ActiveRecord
Механізм шаблонів	Blade	ERB (Embedded Ruby)

Архітектура Ruby on Rails базується на чіткому розподілі логіки програми, що прискорює розробку та забезпечує ефективність управління кодом. Моделі відповідають за обробку даних і бізнес-логіку, контролери забезпечують взаємодію між користувачем і системою, а представлення відображають інформацію та керують інтерфейсом. Такий підхід дозволяє проводити паралельну розробку, що значно скорочує час створення додатків і спрощує їх подальшу підтримку [34].

Ruby on Rails сприяє створенню чистого та структурованого коду завдяки готовим компонентам, які спрощують реалізацію функціональності та роблять код більш читабельним. Це полегшує процес оновлення та розширення проєкту без зайвих змін у базовій структурі.

Один із ключових принципів фреймворку – DRY (Don't Repeat Yourself), що дозволяє уникати дублювання коду. Повторювані завдання реалізуються через шаблони та модулі, що спрощує розробку та підвищує підтримуваність коду.

Rails також вирізняється високою масштабованістю. Він може ефективно працювати навіть під значним навантаженням, що робить його ідеальним для великих веб-платформ. Наприклад, Shopify – одна з найвідоміших платформ електронної комерції, побудована на Ruby on Rails, демонструє стабільність і продуктивність навіть при великій кількості одночасних запитів.

Поєднання швидкої розробки, гнучкості та можливостей масштабування робить Ruby on Rails чудовим вибором для компаній, які прагнуть створювати продуктивні, динамічні та легко підтримувані веб-додатки.

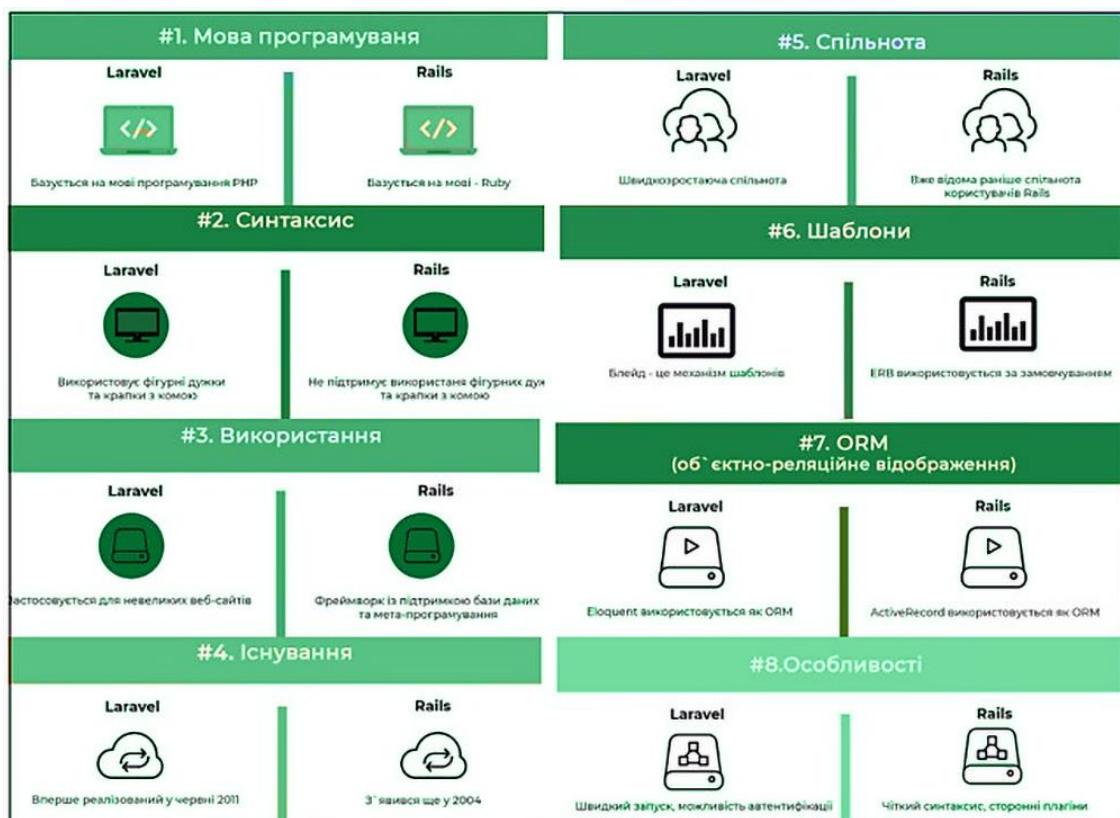


Рисунок 2.14 – Порівняльна інфографіка Laravel проти Ruby on Rails [34]

Ruby on Rails забезпечує високий рівень безпеки завдяки вбудованим механізмам захисту від **SQL-ін'єкцій** та **XSS-атак**, що значно знижує ризики уразливостей у веб-додатках. Фреймворк оптимізує витрати на розробку завдяки швидкому створенню додатків (RAD – Rapid Application Development), що дозволяє скоротити час розробки та пришвидшити вихід продукту на ринок. Rails включає тестове середовище з трьома режимами: виробничим, розробницьким і тестовим, що допомагає проводити всебічне тестування на різних етапах створення додатка. Це дає змогу виявляти та виправляти помилки ще до запуску, підвищуючи стабільність продукту. Фреймворк дотримується принципу конвенцій замість конфігурації, що спрощує початкове налаштування і дозволяє розробникам швидко адаптуватися до роботи з ним. Це сприяє збереженню чистоти коду та спрощує підтримку веб-додатків.

Недоліки Ruby on Rails, які варто враховувати при виборі фреймворку:

1. Документація може бути недостатньо деталізованою, особливо щодо рідкісних функцій або специфічних гемів, що ускладнює пошук необхідної інформації.
2. Швидкість виконання Rails поступається іншим об'єктно-орієнтованим мовам програмування, що може впливати на продуктивність додатків, особливо у високонавантажених системах.
3. Тривалий старт через велику кількість файлів та залежностей. Це може уповільнювати розгортання додатка та ускладнювати розробку.
4. Обмежені можливості хостингу – не всі провайдери підтримують Ruby on Rails через його специфічні вимоги, що може обмежувати вибір інфраструктури для розгортання проєкту [34].

Незважаючи на ці недоліки, Ruby on Rails залишається потужним інструментом для швидкої розробки веб-додатків, особливо якщо пріоритетом є швидкість розробки та зручність підтримки коду.

РОЗДІЛ 3. ЕТАПИ РОЗРОБКИ ДОДАТКУ КОРПОРАТИВНОГО СПІЛКУВАННЯ ЗА ДОПОМОГОЮ ФРЕЙМВОРКУ DJANGO

3.1. Опис функціональності розробки

Згідно мети нашого проєкту, яка полягає в створенні веб-застосунку для інтерактивного спілкування між співробітниками проаналізуємо функціональні та нефункціональні вимоги до створюваної системи.

Функціональні вимоги до створюваного веб-застосунку для інтерактивного спілкування відображають весь набір функцій розроблюваної системи, якими вона повинна бути наділена. Функціональні вимоги до веб-застосунку дозволяють виявити модель прецедентів. Прецедент описує послідовність виконуваних системою дій з метою надання конкретних результатів користувачеві. Складовою частиною моделі прецедентів є діаграма варіантів використання (use case diagram), яка включає представлення всіх можливих дійових осіб, асоційованих з усілякими варіантами використання. Діаграма випадків використання повинна повністю описувати функціональність системи на деякому рівні абстракції [16].

Всі параметри якості веб-застосунку – атрибути якості, описуються нефункціональними вимогами. Атрибути якості є характеристики програмного продукту, що відображають призначені для користувача характеристики: доступність, зручність установки, продуктивність, надійність, умови експлуатації і т.д. до найбільш важливих атрибутів якості відносяться: надійність, продуктивність, зручність використання програмного продукту.

Створюваний веб-застосунок повинен задовольняти наступним функціональним вимогам. Система повинна:

- надавати можливість створити акаунт користувачеві;
- надавати можливість проводити налаштування свого кабінету;
- надавати можливість налаштовувати шляхи комунікації з іншими користувачами (користувачами чату) згідно запитів користувача;
- надавати можливість заблокувати певного користувача;

- могли проводити особисті спілкування з іншим користувачем (щоб інші люди не бачили сутність переписки);
- створити повідомлення для інформування користувача тоді коли інший користувач хоче поспілкуватись (звукове повідомлення або звуковий сигнал);
- могли завантажувати в текст переписки певні аудіо- або відеофайли для поширення серед користувачів чату;
- надавати можливість створювати нотатки та працювати з ними;
- надавати можливість дивитися статистику по проведеному спілкуванню;
- дозволяти вносити зміни про дані користувача.

Під час аналізу предметної області було виділено основні функції та підфункції системи, що відповідають потребам користувача:

1. Нефункціональні вимоги до створюваного веб-застосунку не менш важливі при розробці. Вони дозволяють зробити роботу з програмним продуктом більш комфортним для користувача.
2. Вимоги до надійності. Одним з основних атрибутів якості програмного забезпечення є його надійність – здатність безвідмовно працювати і зберігати характеристики при дотриманні певних умов використання.

Серед вимог до надійності веб-застосунку виділяють:

1. Відновлення після збоїв. При виникненні апаратних збоїв (при відключенні електроживлення мережі), система повинна автоматично відновлюватися після усунення неполадок і / або причин збоїв. Виняток становлять ситуації, пов'язані з технічним пошкодженням носіїв інформації з виконуваним програмним кодом. А також, ситуації, пов'язані з доступом до інформації на сторонньому ресурсі.
2. Обробка помилкових ситуацій. В системі повинні бути передбачені засоби інформування користувача про помилковість його дій. При виникненні винятків система повинна проаналізувати і спробувати ідентифікувати помилку. При успішній ідентифікації помилки система повинна вивести користувачеві відповідне повідомлення.

3. Вимоги до продуктивності. Найважливішими аспектами розробки програмного продукту є продуктивність і продуктивність роботи, тобто здатність програми виконувати передбачені алгоритмом функції швидко і якісно.

Виділяють наступні вимоги до продуктивності веб-застосунку:

1. Час відгуку системи має становити не більше 3 секунд, а час відповіді стороннього ресурсу на запит відповідних даних – не більше 5 секунд.
2. Система повинна бути здатна виводити імпортовані дані за прийнятний час. Виняток становить випадок помилки імпорту даних.
3. Вимоги до ергономіки і зручності експлуатації. Користувальницький інтерфейс забезпечує взаємодію користувача і програмного забезпечення, отже, від його характеристик залежить ефективність використання і задоволення користувача від роботи з системою.

Виділяються наступні вимоги до зручності експлуатації та ергономіки до створюваного веб-застосунку:

- інтерфейс системи повинен бути інтуїтивно зрозумілий користувачеві;
- повідомлення про помилки повинні бути короткими, але інформативними.
- зовнішня поведінка (відгук на дії користувача) однотипних елементів інтерфейсу повинні бути однаково реалізовані.

Також веб-застосунок для спілкування повинен задовольняти наступним нефункціональним вимогам: сучасна версія програмного забезпечення; українська та англійська версія користувальницького інтерфейсу; додаток повинен мати зручний і зрозумілий інтерфейс, бути безкоштовним і без реклами.

Архітектура веб-застосунку для інтерактивного спілкування між співробітниками визначає найбільш загальне уявлення про структуру і логіку функціонування, вказує основні складові і зв'язки між ними, визначає подальшу реалізацію програмного продукту. Для опису архітектури веб-застосунку для інтерактивного спілкування використовується уніфікована мова моделювання (UML) відповідно до рекомендацій методології Rational Unified Process. Пропонована нами архітектура поєднує та автоматизує рівні прийому,

зберігання, обробки та спеціального аналізу. Прототип складається з двох основних робочих процесів для ефективної обробки різних потоків даних, а саме:

- 1) Конвеєр великих даних;
- 2) Конвеєр обробки даних.

Конвеєр великих даних. Основна роль конвеєра великих даних полягає в автоматизації переміщення величезних даних з різними типами даних із зовнішніх джерел потоків даних до бази даних для подальшого аналізу.

Крім переміщення даних, також використовуємо дані з традиційних реляційних баз даних та статичних сховищ даних, таких як файли: jpg, mp3, mp4, json, щоб їх можна було ефективніше аналізувати наступними процесами в базі даних.

Виділяються наступні компоненти UML-специфікації веб-застосунку для інтерактивного спілкування між співробітниками:

1. Концептуальна модель, що включає опис основних сценаріїв роботи системи, варіантів використання, учасників системи.
2. Логічна модель, що включає опис класів і функціональних модулів веб-застосунку.
3. Модель розміщення, що включає вимоги до апаратного забезпечення користувача, опис розміщення компонентів системи.
4. Модель реалізації, що включає опис фізичних артефактів веб-застосунку для інтерактивного спілкування між співробітниками: файли, виконувані файли, використовувані бібліотеки, а також зв'язки між ними.

Концептуальна модель.

В рамках концептуальної моделі веб-застосунку виявляються учасники системи-актори, основні сценарії роботи системи і варіанти використання – прецеденти. Зв'язки акторів і прецедентів представляються на діаграмі варіантів використання.

3.2. Логіка роботи додатку

У межах логічної моделі веб-застосунку для інтерактивного спілкування між співробітниками описуються ключові функціональні модулі та класи системи. Структура проєктованої системи відображена у вигляді діаграми класів (рис. 3.1).

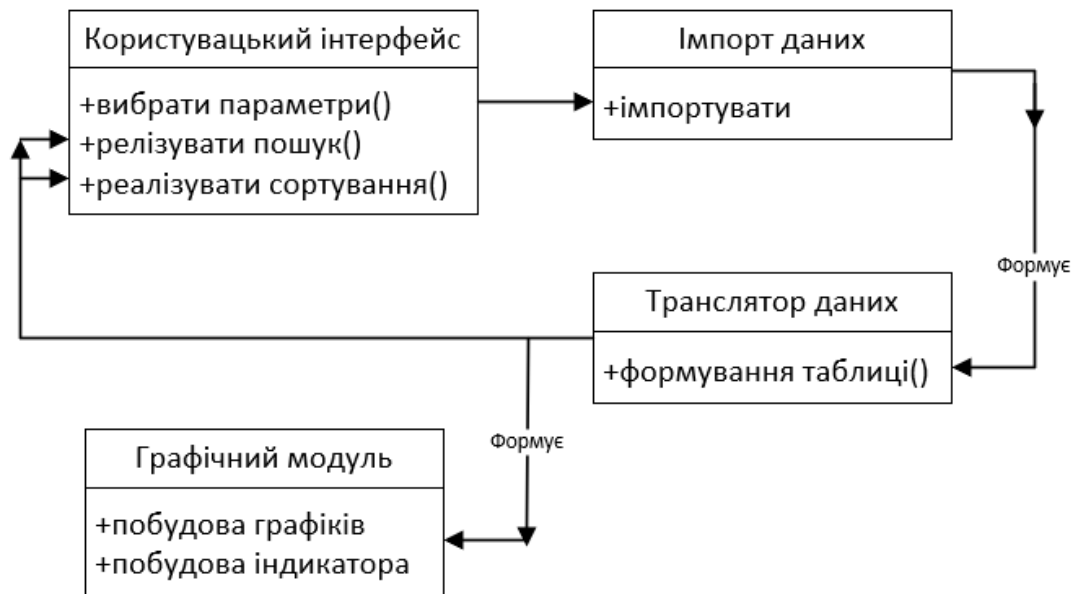


Рисунок 3.1 – Діаграма класів веб-застосунку для інтерактивного спілкування між співробітниками

Рисунок відображає архітектуру веб-застосунку, поділену на основні функціональні модулі, які взаємодіють між собою для забезпечення роботи системи. У центрі схеми розташований користувацький інтерфейс, що відповідає за вибір параметрів, пошук та сортування даних. Він безпосередньо взаємодіє з модулем імпорту даних, який забезпечує введення інформації в систему. Після імпорту дані передаються до транслятора, де здійснюється їх обробка та формування таблиць для подальшого використання. Ці таблиці, у свою чергу, передаються до графічного модуля, який відповідає за візуалізацію інформації у вигляді графіків та індикаторів. Таким чином, схема демонструє потік даних у системі, починаючи від введення користувачем параметрів до їхньої обробки та представлення у зручному графічному форматі.

У таблиці 3.1 наведено опис основних модулів системи. Програмне забезпечення створює веб-сторінку, де інформація про тему спілкування, а також дані щодо активності користувачів оновлюються з незначною затримкою. Імпортовані дані структуруються у вигляді таблиці, після чого система виконує їхню візуалізацію у вигляді графіків та індикаторів.

Таблиця 3.1. Опис модулів веб-застосунку для інтерактивного спілкування

Найменування	Опис
Користувацький інтерфейс	Відповідає за взаємодію користувача із системою, надаючи можливість вибору параметрів, здійснення пошуку та сортування даних.
Імпорт даних	Забезпечує завантаження інформації із сайту спілкування, включаючи додавання нових контактів та блокування небажаних користувачів.
Транслятор даних	Форматує отримані дані та структурує їх у вигляді таблиці для подальшого аналізу.
Графічний модуль	Відповідає за візуалізацію даних, зокрема створення графіків та індикаторів на їх основі.

Ці модулі забезпечують комплексну роботу веб-застосунку, оптимізуючи процес збору, обробки та представлення інформації користувачам.

Модель розміщення. В рамках моделі розміщення веб-застосунку формується діаграма розгортання, що містить опис фізичного простору для розміщення компонентів. Діаграма розгортання представлена на рис. 3.2.

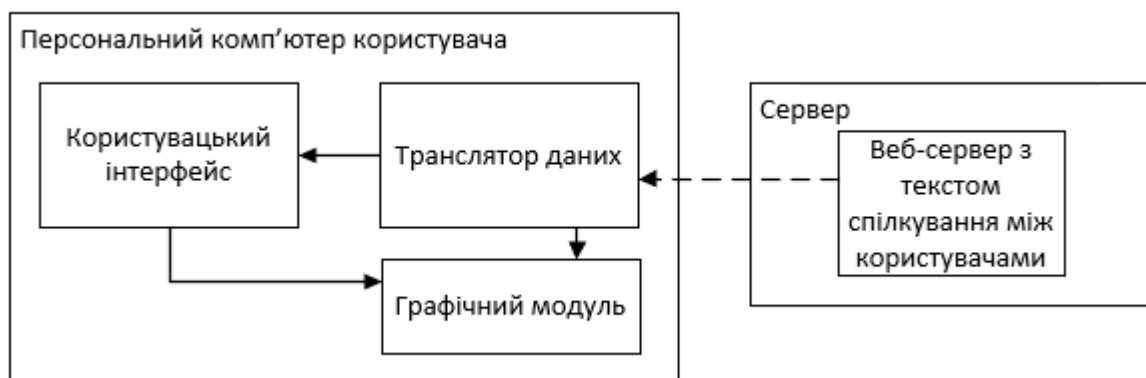


Рисунок 3.2 – Діаграма розгортання веб-застосунку для інтерактивного спілкування працівників підприємства

Для ефективної реалізації веб-застосунку для інтерактивного спілкування необхідно визначити ключові логічні та функціональні структури. Опис основних компонентів системи здійснюється за допомогою діаграми компонентів. Формування дерева цілей для розробки застосунку базується на трансформації проблем у конкретні завдання, встановленні взаємозв'язків між ними за принципом «засоби – результати» та визначенні основних напрямків розвитку. Алгоритм прийняття рішень у межах веб-застосунку належить до класу алгоритмів навчання з наглядом, що дозволяє застосовувати його як для класифікації, так і для прогнозування даних. Основним завданням цього дослідження є впровадження веб-застосунку для корпоративного спілкування. Для досягнення поставленої мети необхідно реалізувати такі ключові етапи: дослідження предметної області, розробка системної архітектури, програмна реалізація, оптимізація та подальша підтримка продукту.

Детальне відображення взаємозв'язків між цілями та засобами їх досягнення представлено у вигляді дерева цілей (рис. 3.3).



Рисунок 3.3 – Приклад, як виглядає дерево цілей згідно досліджуваної проблематики в роботі

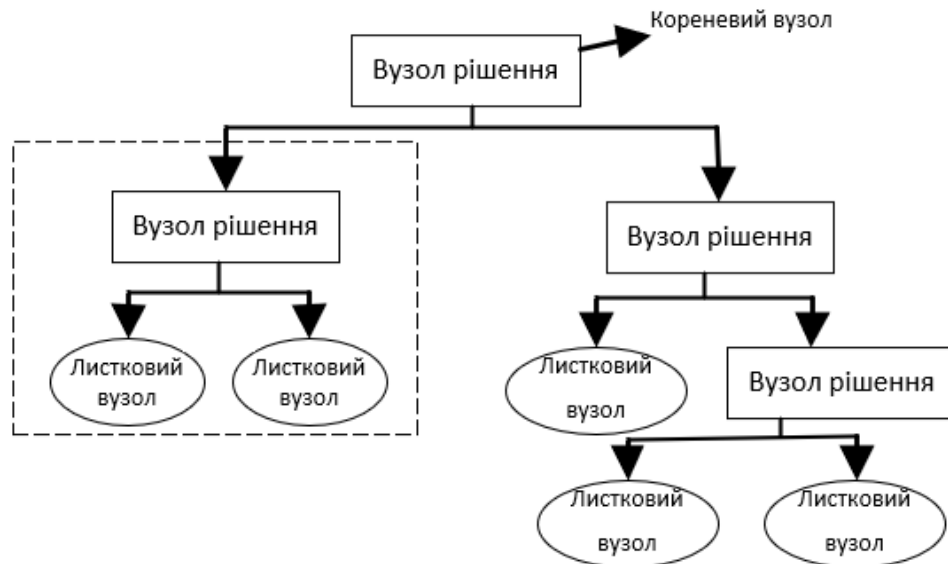


Рисунок 3.4 – Приклад роботи алгоритму дерева цілей для веб-застосунку для інтерактивного спілкування між співробітниками

Контекстна діаграма або діаграма нульового рівня, представлена на рис. 3.5, відображає загальну структуру системи спілкування, включаючи її основні взаємодії з користувачами та потоками даних. Ця діаграма є фундаментальною для визначення меж системи та основних вхідних і вихідних даних.



Рисунок 3.5 – Діаграма потоків даних нульового рівня

Діаграма демонструє, як користувачі можуть взаємодіяти з системою спілкування та як дані передаються між різними процесами, забезпечуючи належну функціональність та логіку системи. У моделі представлено дві ключові сутності: «Користувач системи із спілкування між користувачами» та «Надати доступ до системи спілкування». Користувач взаємодіє із системою через низку запитів та відповідей, що передбачають обробку різних інформаційних потоків.

Основні інформаційні потоки, що циркулюють між користувачем та системою, включають:

- Авторизаційні дані – забезпечують перевірку облікових записів під час входу в систему.
- Відгук про реєстрацію акаунта – підтвердження успішного створення нового користувацького профілю.
- Запити від системи – надсилання системних сповіщень або запитів до користувача.
- Запити від користувача – ініційовані користувачем дії, такі як запити на зміну налаштувань або пошук співрозмовників.
- Дані про знайдені варіанти – результати пошуку контактів або можливих зв'язків у системі.
- Запит додавання / видалення – запити користувачів щодо керування списком контактів, таких як додавання нових співрозмовників або блокування небажаних користувачів.
- Встановлення налаштувань – персоналізація профілю та параметрів комунікації [16].

Процеси які були сформовані, бази даних та потоки зображено на рис.3.6,.



Рисунок 3.6 – Декомпозиція системи «Прогнозування зміни» першого рівня

Наступним кроком потрібно зображено декомпозицію другого рівня процесу «Створити обліковий запис», який розбито на такі процеси: зареєструвати клієнта; авторизувати клієнта.



Рисунок 3.7 – Декомпозиція процесу «Створення облікового запису» для застосунку

Діаграма на рисунку представляє декомпозицію процесу «Створити обліковий запис» у веб-застосунку для інтерактивного спілкування. Вона деталізує ключові етапи реєстрації користувача та їхню взаємодію із системою. Процес починається з авторизаційних даних, які надходять до підпроцесу «Вибір кімнати спілкування». На цьому етапі перевіряється інформація про користувача, після чого надсилається підтвердження про успішне створення облікового запису.

Далі, після вибору кімнати для спілкування, система передає облікові дані до наступного етапу – «Систематизація результатів спілкування у додатку». Цей підпроцес структурує збережені дані користувача та забезпечує їхню відповідність обраному варіанту взаємодії.

Система також дозволяє обробляти запити на організацію списку потреб, що може включати додавання нових контактів, вибір груп або налаштування профілю. Як результат, користувач отримує доступ до свого облікового запису, а система зберігає необхідну інформацію. Таким чином, дана діаграма

демонструє ключові етапи створення та організації облікового запису користувача, забезпечуючи послідовний процес інтеграції нового учасника у систему спілкування. Зв'язані між собою обліковими даними, які будуть передаватись при авторизації користувача для опрацювання даних та надання доступу до застосунку користувачу.

Наступний процесом описано дії користувача в межі пошуку результатів тематики спілкування, зображено на рис. 3.8. Сформований запит на перегляд результатів спілкування відправляється користувачем системи та опрацьовується системою. Система реагує на запит та формує список доступних кімнат за тематикою.

Наступним складеним процесом є «Сформувати список» обраної тематики. Він складається із процесів, а саме: створити запит на завантаження даних; затвердити вибрані дані. Об'єднує між собою опрацьований матеріал, тобто програми які зараз мають можливість транслюватись. На виході отримуємо опрацьовані дані та запит на перегляд даних.

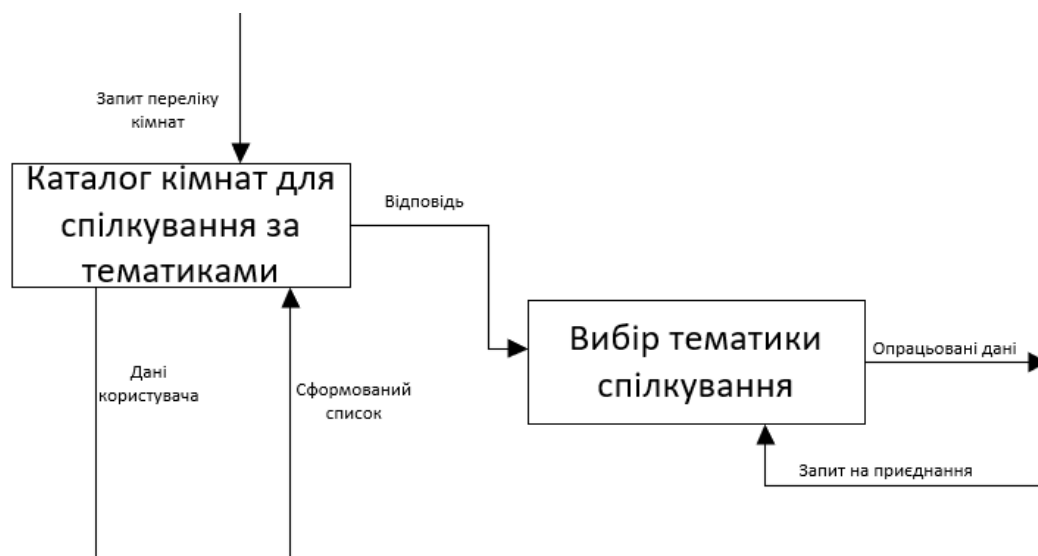


Рисунок 3.8 – Декомпозиція процесу «Формування списків» тематик

Рисунок 3.8 ілюструє декомпозицію процесу «Сформувати список» тем у веб-застосунку для інтерактивного спілкування. Процес починається з надходження даних користувача, на основі яких формується каталог кімнат за тематиками. Користувач надсилає запит на перелік кімнат, після чого система обробляє інформацію і надає сформований список доступних варіантів для

вибору. Наступним кроком є вибір тематики спілкування, під час якого система надсилає відповідь на запит користувача. Якщо користувач підтверджує бажання приєднатися до певної теми, надходить запит на приєднання, і система передає опрацьовані дані для подальшого використання в процесі комунікації.

Дані процеси пов'язані між собою за допомогою програм, які будуть обрані користувачем та опрацьованих запитів. На вході отримано запит пошуку архівних тем спілкування та відновлення переписки при запиті користувача.

В ході проєктування бази даних були створені наступні таблиці (Рис. 3.9):

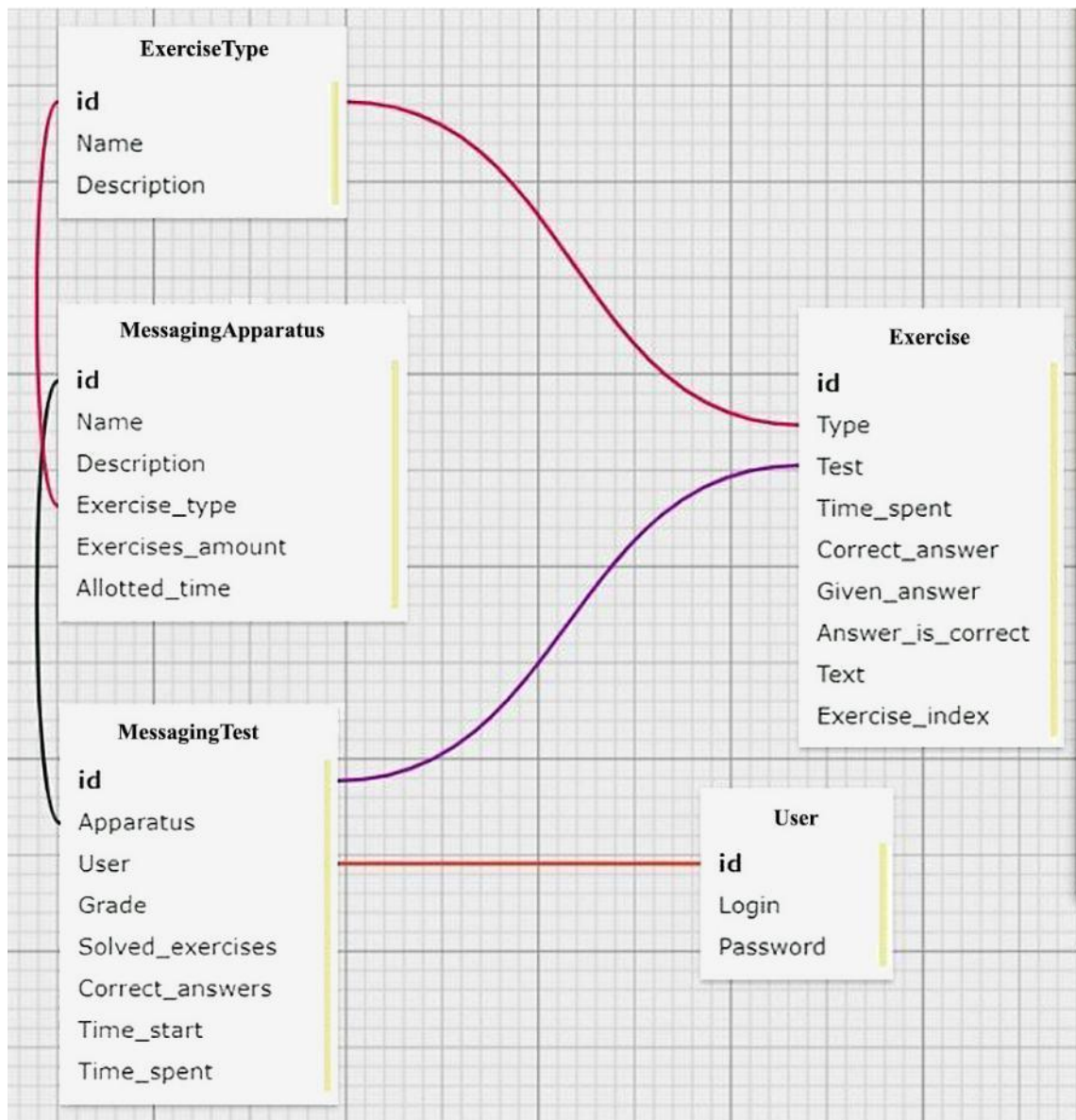


Рисунок 3.9 – База даних у вигляді схеми

1. MessagingApparatus – Конструктор тематик спілкування;
 - Name – Ім'я користувача;

- Description – формулювання запиту до групи спілкування;
 - Exercise_type – тема спілкування;
 - Exercises_amount – кількість користувачів у групі спілкування;
 - Allotted_time – виділений час на спілкування.
2. ExerciseType – тип спілкування;
- Name – Найменування типу;
 - Description – Опис.
3. MessagingTest – Проведене спілкування;
- Apparatus – Конструктор тематик спілкування
 - User – користувач, який бере участь у спілкуванні;
 - Grade – Оцінка якості і цікавості тематики спілкування;
 - Solved_exercises – кількість вирішених проблемних тем;
 - Correct_answers – кількість користувачів на кінець спілкування (актуальність теми);
 - Time_start – час початку;
 - Time_spent – витрачений час;
4. Exercise – предмет спілкування.
5. User – Користувач:
- Login – Логін;
 - Password – Пароль;

Проведемо наповнення та тестування таблиць БД. SQL-запити для наповнення таблиць бази даних:

```
- Insert data into the User table
INSERT INTO User (id, Login, Password)
VALUES
(1, 'kai_igor', 'pass123'),
(2, 'alex_shev', 'pass456'),
(3, 'maria_pet', 'pass789'),
(4, 'kat_sid', 'pass101');
```

```
- Insert data into the ExerciseType table
INSERT INTO ExerciseType (id, Name, Description)
VALUES
(1, 'Механіка', 'Теорія і практика механічних інструментів.'),
(2, 'Електрика', 'Основи роботи з електричними інструментами.'),
```

```

(3, 'Пайка', 'Практичні навички пайки компонентів.'),
(4, 'Безпека', 'Основи безпеки під час роботи з інструментами.');
```

– Insert data into the MessagingApparatus table

```

INSERT INTO MessagingApparatus (id, Name, Description,
Exercise_type, Exercises_amount, Allotted_time)
VALUES
(1, 'Симулятор дреля', 'Практичний тренажер для роботи з дрилем.',
1, 10, 30),
(2, 'Тренажер пайки', 'Модуль для навчання пайці дрібних
компонентів.', 3, 8, 25),
(3, 'Симулятор болгарки', 'Тренажер для шліфування і різання
металу.', 2, 12, 40),
(4, 'Модуль безпеки', 'Модуль для відпрацювання правил безпеки.',
4, 6, 15);
```

– Insert data into the Exercise table

```

INSERT INTO Exercise (id, Type, Test, Time_spent, Correct_answer,
Given_answer, Answer_is_correct, Text, Exercise_index)
VALUES
(1, 'Тест', 'Дриль', 10, 5, 5, TRUE, 'Які свердла підходять для
металу?', 101),
(2, 'Практика', 'Пайка', 20, 7, 6, FALSE, 'Підберіть припій для
мідного проводу.', 102),
(3, 'Практика', 'Шліфування', 25, 6, 6, TRUE, 'Виберіть диск для
різання металу.', 103),
(4, 'Тест', 'Безпека', 15, 4, 3, FALSE, 'Які засоби захисту
необхідні для роботи з болгаркою?', 104);
```

– Insert data into the TrainingTest table

```

INSERT INTO TrainingTest (id, Apparatus, User, Grade,
Solved_exercises, Correct_answers, Time_start, Time_spent)
VALUES
(1, 'Симулятор дреля', 'kai_igor', 85, 8, 7, '2025-02-01
10:00:00', 20),
(2, 'Тренажер пайки', 'alex_shev', 70, 6, 5, '2025-02-01
11:00:00', 30),
(3, 'Симулятор болгарки', 'maria_pet', 90, 10, 9, '2025-02-01
12:00:00', 25),
(4, 'Модуль безпеки', 'kat_sid', 60, 5, 3, '2025-02-01 13:00:00',
15);
```

Наведемо скріншоти розроблених таблиць БД, спроектованого додатку (Рис. 3.10–3.14).

	id	Login	Password	
1	1	kai_igor	pass123	
2	2	alex_shev	pass456	
3	3	maria_pet	pass789	
4	4	kat_sid	pass101	

Рисунок 3.10 – Таблиця User

Таблиця User зберігає дані про користувачів системи, включаючи їх логін, пароль і унікальний ідентифікатор.

	id	Type	Test	Time_spent	Correct_answer	Given_answer	Answer_is_correct	Text	Exercise_index	
1	1	Тест	Дриль	10	5	5	True	Які свердла підходять для металу?	101	
2	2	Практика	Пайка	20	7	6	False	Підберіть припій для мідного проводу.	102	
3	3	Практика	Шліфування	25	6	6	True	Виберіть диск для різання металу.	103	
4	4	Тест	Безпека	15	4	3	False	Які засоби захисту необхідні для роботи з болгаркою?	104	

Рисунок 3.11 – Таблиця Exercise

Таблиця Exercise містить завдання або вправи з деталями, такими як тип, текст завдання, правильні та дані відповіді, а також час, витрачений на виконання.

Training Apparatus Table (Updated Structure)							
	id	Name	Description	Exercise_type	Exercises_amount	Allotted_time	
1	1	Симулятор дреля	Практичний тренажер для роботи з дрилем.	1	10	30	
2	2	Тренажер пайки	Модуль для навчання пайці дрібних компонентів.	3	8	25	
3	3	Симулятор болгарки	Тренажер для шліфування і різання металу.	2	12	40	
4	4	Модуль безпеки	Модуль для відпрацювання правил безпеки.	4	6	15	

Рисунок 3.12 – Таблиця MessagingApparatus

У MessagingApparatus описані тренажери чи модулі, що використовуються для навчання, із зазначенням їхньої назви, типу вправ, кількості завдань і відведеного часу.

Exercise Type Table (Updated Structure)				
	id	Name	Description	
1	1	Механіка	Теорія і практика механічних інструментів.	
2	2	Електрика	Основи роботи з електричними інструментами.	
3	3	Пайка	Практичні навички пайки компонентів.	
4	4	Безпека	Основи безпеки під час роботи з інструментами.	

Рисунок 3.13 – Таблиця ExerciseType

Таблиця ExerciseType містить інформацію про типи вправ, включаючи назви та описи, що відображають тематичний напрямок навчання або тренувань.

Training Test Table (Updated Structure)

	id	Apparatus	User	Grade	Solved_exercises	Correct_answers	Time_start	Time_spent
1	1	Симулятор дреля	kai_igor	85	8	7	2025-02-01 10:00:00	20
2	2	Тренажер пайки	alex_shev	70	6	5	2025-02-01 11:00:00	30
3	3	Симулятор болгарки	maria_pet	90	10	9	2025-02-01 12:00:00	25
4	4	Модуль безпеки	kat_sid	60	5	3	2025-02-01 13:00:00	15

Рисунок 3.14 – Таблиця TrainingTest

TrainingTest зберігає результати проходження тестів або тренувань, включаючи оцінки, кількість виконаних вправ, правильні відповіді, час початку та тривалість сесії.

3.3. Розробка програмних модулів додатку

Веб-застосунок для спілкування був написаний на Django, і він складається з модулів, кожен з яких відповідає за свій розділ і складається з наступних елементів:

1. Модель. Моделі відображають інформацію про дані, з якими необхідно працювати. Вони містять поля та поведінку даних. Звичайна одна модель представляє одну таблицю в базі даних. Код, який описує моделі, знаходиться у файлі `models.py` всередині розроблюваного модуля. На основі цих файлів будуть створюватися міграція. Приклад моделі можна побачити нижче. У цьому прикладі ChatRoom – це модель, що представляє чат-кімнати, кожна з яких має назву, опис та дату створення. Поле `name` є унікальним, що запобігає дублюванню кімнат, а `created_at` автоматично фіксує дату створення запису. Такі моделі є основою веб-застосунку, оскільки визначають, як організовано збереження даних та їхню взаємодію з іншими частинами системи.

```
from django.db import models

class ChatRoom (models.Model):
```

```

name = models.CharField (max_length=100, unique=True)
description = models.TextField (blank=True, null=True)
created_at = models.DateTimeField (auto_now_add=True)

def __str__(self):
    return self.name

```

2. Міграції в Django – це механізм, що дозволяє відстежувати зміни в моделях і застосовувати їх до бази даних. Вони використовуються для додавання нових полів, видалення моделей або зміни структури даних без втрати інформації. Файли міграцій зберігаються у каталозі **migrations** кожного окремого застосунку і є частиною кодової бази, що дозволяє їх передавати та розповсюджувати разом із проєктом. Django забезпечує послідовність виконання міграцій, тому незалежно від середовища (розробка, тестування чи продакшн), зміни будуть застосовуватися однаково, гарантуючи узгодженість схеми бази даних. Міграції реєструють будь-які зміни в моделях, навіть якщо вони не впливають безпосередньо на структуру бази даних. Це необхідно, щоб можна було коректно відновити будь-яке поле або налаштування, зберігаючи цілісність історії змін. Приклад міграції можна побачити нижче.

```

from django.db import migrations, models
class Migration (migrations.Migration):
    dependencies = [(«migrations», «0001_initial»)]
    operations = [
        migrations.DeleteModel («Tribble»),
        migrations.AddField («Author», «rating», models.IntegerField
            (default=0)),
    ]

```

Цей код демонструє видалення моделі Tribble і додавання нового поля rating до моделі Author, де rating є цілим числом із початковим значенням 0. Django автоматично генерує такі файли при виконанні команди. Міграції є зручним та ефективним інструментом управління структурою бази даних у Django.

3. Уявлення (Views) у Django. У Django уявлення (Views) – це функції або класи, які обробляють вхідні HTTP-запити і повертають відповідні HTTP-відповіді. Відповідь може містити HTML-сторінку, JSON-дані, зображення

або будь-яку іншу інформацію. Уявлення є основною логікою програми, що визначає, як обробляти запити користувачів.

Django дозволяє організовувати уявлення у вигляді функцій або класів:

- Функціональні уявлення – стандартні Python-функції, які приймають запит і повертають відповідь.
- Класові уявлення (Class-Based Views, CBV) – використовують механізм об'єктно-орієнтованого програмування для поділу логіки.

За загальноприйнятим підходом уявлення зберігаються у файлі `views.py`, який знаходиться у каталозі програми в межах Django-проєкту.

Приклад класового уявлення в Django:

```
```python
from django.views.generic import ListView
from django.http import HttpResponse
from models import Book

class BookListView (ListView):
 model = Book

 def head (self, *args, kwargs):
 last_book = self.get_queryset().latest('publication_date')
 response = HttpResponse(
 headers={'Last-Modified': last_book.publication_date.
 strftime('%a, %d %b %Y %H:%M:%S GMT')}
)
 return response
```
```

Опис коду:

- `BookListView` – класове уявлення, яке використовує Django-клас `ListView` для відображення списку книг.
- `model = Book` – визначає модель, з якою працює уявлення.
- Метод `head()` – реалізує HEAD-запит, який повертає дату останньої модифікації ресурсу.
- `HttpResponse` – створює відповідь із заголовком `Last-Modified`, який містить дату останньої публікації книги у форматі RFC 1123.

Класові уявлення полегшують створення веб-додатків, розділяючи логіку обробки запитів та забезпечуючи зручний механізм для створення відповідей.

4. Сериалізатори у Django REST Framework (DRF) дозволяють перетворювати складні об'єкти, такі як набори запитів або екземпляри моделей, у зручні для передачі формати, наприклад JSON або XML. Вони також виконують зворотний процес – десериалізацію, перетворюючи отримані дані у структуровані Python-об'єкти після перевірки вхідних даних.

У DRF серіалізатори працюють аналогічно до Django Form та ModelForm, забезпечуючи зручний механізм для валідації та трансформації даних.

Основні типи серіалізаторів:

- ``Serializer`` – базовий клас серіалізаторів, який дає можливість налаштовувати структуру даних вручну.
- ``ModelSerializer`` – спрощений варіант, який автоматично генерує поля на основі Django-моделі.

Сериалізатор:

```
```python
from rest_framework import serializers

class CommentSerializer(serializers.Serializer):
 email = serializers.EmailField()
 content = serializers.CharField(max_length=200)
 created = serializers.DateTimeField()
```
```

Опис коду:

- ``CommentSerializer`` – клас серіалізатора, що визначає структуру коментаря.
- ``email = serializers.EmailField()`` – поле для зберігання електронної пошти.
- ``content = serializers.CharField(max_length=200)`` – текстове поле з максимальним обмеженням у 200 символів.
- ``created = serializers.DateTimeField()`` – поле для зберігання дати та часу створення коментаря.

Сериалізатори допомагають легко працювати з API, забезпечуючи зручну взаємодію між клієнтськими додатками та сервером.

5. Конфігурація URL у Django. У Django кожна веб-сторінка має унікальну URL-адресу, яка вказує додатку, яку інформацію потрібно показати

користувачеві. Для цього використовується механізм URLconf (URL Configuration) – набір правил, які дозволяють співставити отриманий запит з відповідним поданням (view).

Основні аспекти конфігурації URL:

- Файл `urls.py` містить шаблони URL-адрес, які спрямовують запити на відповідні функції у `views.py`.
- Використовується функція `path()`, що визначає маршрут та відповідне подання.
- Динамічні параметри, такі як `<int: year>`, `<slug: slug>`, дозволяють передавати змінні через URL.

Конфігурація URL:

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

urlpatterns = [
 path('articles / 2003/', views.special_case_2003), # Статична
 URL-адреса
 path('articles/<int: year>', views.year_archive), # Динамічний
 рік у URL
 path('articles/<int: year>/<int: month>', views.month_archive),
 # Рік і місяць
 path('articles/<int: year>/<int: month>/<slug: slug>', views.
 article_detail) # Стаття за slug
]
```
```

Опис коду:

- `path('articles / 2003/', views.special_case_2003)` – обробка статичної URL-адреси для конкретного випадку.
- `path('articles/<int: year>', views.year_archive)` – динамічна URL-адреса, яка передає рік у подання `year_archive`.
- `path('articles/<int: year>/<int: month>', views.month_archive)` – передача року та місяця.
- `path('articles/<int: year>/<int: month>/<slug: slug>', views.article_detail)` – використання slug для унікальної ідентифікації статей.

Ця система дозволяє гнучко налаштовувати маршрутизацію у Django-додатку, що значно полегшує роботу з URL-адресами.

Розробимо код моделі `MessagingApparatus`, яка відповідає вашому проєкту «Розробка кросплатформенного месенджера для внутрішньої комунікації співробітників підприємства Dnipro-M». Ця модель буде містити поля для організації комунікації між співробітниками, а також систему оцінювання ефективності спілкування.

Код моделі:

```
```python
from django.db import models

class CommunicationEfficiency (models.Model):
 «» «Модель оцінки ефективності комунікації між співробітниками.»»
 efficiency_level = models.CharField (max_length=50,
 verbose_name=«Рівень ефективності»)
 min_response_time = models.PositiveIntegerField
 (verbose_name=«Мінімальний час відповіді (сек)»)
 message_success_rate = models.FloatField (verbose_name=«Відсоток
 успішно доставлених повідомлень»)

 class Meta:
 verbose_name = «Ефективність комунікації»
 verbose_name_plural = «Ефективність комунікації»
 ordering = ['min_response_time']

 def __str__(self):
 return f»{self.efficiency_level} (Час відповіді: {self.
 min_response_time} сек, Доставка: {self.message_success_rate}%)»

class MessagingApparatus (CommunicationEfficiency):
 «» «Модель месенджера для внутрішньої комунікації
 співробітників.»»
 user = models.ForeignKey('auth.User', on_delete=models.CASCADE,
 verbose_name=«Користувач»)
 chat_name = models.CharField (max_length=100, verbose_name=«Назва
 чату»)
 chat_description = models.TextField (blank=True,
 verbose_name=«Опис чату»)
 encryption_enabled = models.BooleanField (default=True,
 verbose_name=«Шифрування активоване»)
 max_participants = models.PositiveIntegerField (default=50,
 verbose_name=«Максимальна кількість учасників»)
 created_at = models.DateTimeField (auto_now_add=True,
 verbose_name=«Дата створення»)
```

```

updated_at = models.DateTimeField (auto_now=True,
verbose_name=«Останнє оновлення»)

class Meta:
verbose_name = «Месенджер»
verbose_name_plural = «Месенджери»
ordering = ['created_at']

def __str__(self):
return f»{self.chat_name} (Користувач: {self.user.username})»
` ``

```

#### Пояснення:

- `CommunicationEfficiency` - модель для оцінки ефективності комунікації: включає рівень ефективності, час відповіді та рівень успішної доставки повідомлень.
- `MessagingApparatus` - головна модель месенджера, яка містить інформацію про чат, користувачів, шифрування та максимальну кількість учасників.
- `ForeignKey('auth. User')` - зв'язок із вбудованою системою аутентифікації Django для збереження інформації про учасників чату.
- `encryption_enabled` - визначає, чи активоване шифрування.
- `max_participants` - обмеження на кількість учасників у груповому чаті.
- `created_at` та `updated_at` - автоматично фіксують час створення та останнього оновлення.

Ця модель є основою для нашого корпоративного месенджера та дозволить керувати чатами та їхніми параметрами.

Після реалізації моделі `MessagingApparatus`, яка відповідає за організацію чатів та ефективність комунікації, потрібно розробити інші ключові складові месенджера. Серед основних модулів:

1. Система обміну повідомленнями
2. Механізм обробки користувацьких статусів та ролей
3. Система шифрування повідомлень
4. Повідомлення та push-сповіщення
5. Логування та моніторинг активності
6. Система управління групами та правами доступу

1. Модель повідомлень (Messaging System). Ця модель відповідальна за збереження історії повідомлень, відправку, перегляд та статус доставки.

### Код моделі:

```

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

class Message (models.Model):
 «» «Модель збереження повідомлень у месенджері.»»
 sender = models.ForeignKey (User, on_delete=models.CASCADE,
 related_name=«sent_messages», verbose_name=«Відправник»)
 receiver = models.ForeignKey (User, on_delete=models.CASCADE,
 related_name=«received_messages», verbose_name=«Отримувач»)
 chat = models.ForeignKey('MessagingApparatus', on_delete=models.
 CASCADE, verbose_name=«Чат»)
 content = models.TextField (verbose_name=«Текст повідомлення»)
 sent_at = models.DateTimeField (auto_now_add=True,
 verbose_name=«Час відправки»)
 is_read = models.BooleanField (default=False,
 verbose_name=«Прочитано»)

class Meta:
 verbose_name = «Повідомлення»
 verbose_name_plural = «Повідомлення»
 ordering = ['-sent_at']

def __str__(self):
 return f'Повідомлення від {self.sender.username} до {self.
 receiver.username} у {self.chat.chat_name}»
`

```

### Основні можливості:

- sender / receiver – зв'язки з користувачами (відправник, отримувач).
- chat – повідомлення відносяться до конкретного чату.
- content – збереження тексту повідомлення.
- sent\_at – автоматичне збереження часу надсилання.
- is\_read – позначка, чи було прочитано повідомлення.

2. Користувацькі статуси та ролі. Для визначення рівнів доступу необхідно створити окрему модель UserStatus.

### Код моделі:

```

`python
class UserStatus (models.Model):
 «» «Модель для управління статусами користувачів у месенджері.»»
 user = models.OneToOneField (User, on_delete=models.CASCADE,
 verbose_name=«Користувач»)

```

```

status_text = models.CharField (max_length=100, default=«Онлайн»,
verbose_name=«Статус»)
last_seen = models.DateTimeField (auto_now=True,
verbose_name=«Остання активність»)

class Meta:
verbose_name = «Статус користувача»
verbose_name_plural = «Статуси користувачів»

def __str__(self):
return f»{self.user.username}: {self.status_text} (Остання
активність: {self.last_seen})»
` ``

```

#### Функціонал:

- Кожен користувач має статус (онлайн, офлайн, зайнятий тощо).
- Поле last\_seen автоматично оновлюється при зміні статусу.

### 3. Шифрування повідомлень

Безпека комунікації є важливою складовою месенджера. Для цього можна реалізувати базове AES-шифрування повідомлень.

#### Код механізму шифрування:

```

` ``python
from cryptography.fernet import Fernet

class EncryptionManager:
«» «Клас для шифрування та дешифрування повідомлень.»"»
key = Fernet.generate_key()
cipher_suite = Fernet (key)

@staticmethod
def encrypt_message (message):
return EncryptionManager.cipher_suite.encrypt (message.
encode()). decode()

@staticmethod
def decrypt_message (encrypted_message):
return EncryptionManager.cipher_suite.decrypt
(encrypted_message.encode()). decode()
` ``

```

#### Функціонал:

- Генерація ключа шифрування для кожного чату.
- Шифрування перед надсиланням та дешифрування при отриманні.

### 4. Push-сповіщення

Щоб користувачі отримували миттєві повідомлення, можна реалізувати Push-сповіщення через Django Channels.

Код для push-сповіщень:

```
```python
from django.db import models

class Notification (models.Model):
    «» «Модель push-сповіщень.»»
    user = models.ForeignKey (User, on_delete=models.CASCADE,
        verbose_name=«Користувач»)
    message = models.ForeignKey (Message, on_delete=models.CASCADE,
        verbose_name=«Повідомлення»)
    is_seen = models.BooleanField (default=False,
        verbose_name=«Переглянуто»)
    created_at = models.DateTimeField (auto_now_add=True,
        verbose_name=«Час створення»)

class Meta:
    verbose_name = «Сповіщення»
    verbose_name_plural = «Сповіщення»
    ordering = ['-created_at']

def __str__(self):
    return f'Сповіщення для {self.user.username} про повідомлення
    {self.message.id}»
```
```

Функціонал:

- Кожен користувач отримує сповіщення при надходженні нового повідомлення.
  - `is_seen` визначає, чи було переглянуто сповіщення.
5. Логування активності користувачів. Для моніторингу подій месенджера необхідно вести лог-файл усіх важливих змін.

Код логування активності:

```
```python
import logging

logger = logging.getLogger(__name__)

def log_message_activity (sender, receiver, chat, content):
    logger.info (f'Повідомлення від {sender} до {receiver} в {chat}:
    {content}»)
```
```

#### Функціонал:

- Фіксація подій (відправка, прочитання, помилки).
  - Запис у log-файл для подальшого аналізу.
6. Управління групами та правами доступу. Щоб забезпечити розмежування прав доступу, необхідно створити модель GroupChat.

#### Код моделі:

```
```python
class GroupChat (models. Model):
    «» «Модель групових чатів у месенджері.»»
    name = models. CharField (max_length=100, verbose_name=«Назва
    групи»)
    admin = models. ForeignKey (User, on_delete=models. CASCADE,
    related_name=«admin_groups», verbose_name=«Адміністратор»)
    members = models. ManyToManyField (User,
    related_name=«group_members», verbose_name=«Учасники»)
    created_at = models. DateTimeField (auto_now_add=True,
    verbose_name=«Дата створення»)

    class Meta:
        verbose_name = «Груповий чат»
        verbose_name_plural = «Групові чати»
        ordering = ['created_at']

    def __str__(self):
        return f»{self. name} (Адміністратор: {self. admin. username})»
```
```

#### Функціонал:

- Адміністратор групи визначає права доступу.
  - Можливість додавати учасників та видаляти їх.
- Отже, розроблена архітектура месенджера дозволяє:
- Виконувати обмін повідомленнями між співробітниками.
  - Гарантувати безпеку даних через шифрування.
  - Керувати статусами та ролями користувачів.
  - Вести логування активності для моніторингу безпеки.
  - Створювати групові чати з розширеними можливостями.
  - Реалізовувати Push-сповіщення для швидкої комунікації.

Наступним етапом буде розробка API для інтеграції з мобільними та десктопними додатками для кросплатформенного месенджера Dnipro-M. На

цьому етапі зосередимося на створенні основних ендпоінтів для обробки повідомлень, управління чатами та аутентифікації користувачів.

7. Аутентифікація користувачів. Для управління користувачами та їхньою аутентифікацією використаємо Django REST Framework (DRF) та Simple JWT для роботи з токенами.

Модель користувача (User). Оскільки месенджер має підтримувати ідентифікацію користувачів, створимо кастомну модель користувача:

```
```python
from django.contrib.auth.models import AbstractUser
from django.db import models

class CustomUser (AbstractUser):
    email = models.EmailField (unique=True)
    avatar = models.ImageField (upload_to=«avatars/», null=True,
    blank=True)

    def __str__(self):
    return self.username
```
```

Серіалізатор для користувачів:

```
```python
from rest_framework import serializers
from .models import CustomUser

class UserSerializer (serializers.ModelSerializer):
    class Meta:
    model = CustomUser
    fields = («id», «username», «email», «avatar»)
```
```

Ендпоінти для реєстрації та аутентифікації:

```
```python
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework import status
from rest_framework.permissions import AllowAny
from django.contrib.auth import get_user_model
from rest_framework_simplejwt.tokens import RefreshToken
from .serializers import UserSerializer

User = get_user_model()
```

```

class RegisterUser (APIView):
    permission_classes = [AllowAny]

    def post (self, request):
        serializer = UserSerializer (data=request. data)
        if serializer. is_valid():
            user = serializer. save()
            refresh = RefreshToken. for_user (user)
            return Response({
                «user»: serializer. data,
                «refresh»: str (refresh),
                «access»: str (refresh. access_token)
            }, status=status. HTTP_201_CREATED)
        return Response (serializer. errors, status=status.
            HTTP_400_BAD_REQUEST)
    ``

```

Маршрутизація для аутентифікації (`urls. py`):

```

``python
from django. urls import path
from. views import RegisterUser
from rest_framework_simplejwt. views import TokenObtainPairView,
TokenRefreshView

urlpatterns = [
    path('register/', RegisterUser. as_view(), name='register'),
    path('token/', TokenObtainPairView. as_view(),
        name='token_obtain_pair'),
    path('token / refresh/', TokenRefreshView. as_view(),
        name='token_refresh'),
]
``

```

8. Управління чатами. Месенджер повинен підтримувати групові та приватні чати, тому створимо моделі для чатів.

Модель чату:

```

``python
class Chat (models. Model):
    name = models. CharField (max_length=255, null=True, blank=True)
    is_group = models. BooleanField (default=False)
    members = models. ManyToManyField (CustomUser,
        related_name=«chats»)

    def __str__(self):
        return self. name if self. name else f'Chat {self. id}»
    ``

```

Серіалізатор для чату:

```
```python
class ChatSerializer (serializers. ModelSerializer):
members = UserSerializer (many=True, read_only=True)

class Meta:
model = Chat
fields = («id», «name», «is_group», «members»)
```
```

Ендпоінти для створення та отримання чатів:

```
```python
from rest_framework import generics
from . models import Chat
from . serializers import ChatSerializer
from rest_framework.permissions import IsAuthenticated

class ChatListCreateView (generics. ListCreateAPIView):
serializer_class = ChatSerializer
permission_classes = [IsAuthenticated]

def get_queryset (self):
return Chat. objects. filter (members=self. request. user)

def perform_create (self, serializer):
chat = serializer. save()
chat. members. add (self. request. user)
```
```

Маршрутизація для чатів:

```
```python
urlpatterns += [
path('chats/', ChatListCreateView. as_view(), name='chat-list-
create'),
]
```
```

9. Обробка повідомлень

Модель повідомлення:

```
```python
class Message (models. Model):
chat = models. ForeignKey (Chat, on_delete=models. CASCADE,
related_name=«messages»)
sender = models. ForeignKey (CustomUser, on_delete=models.
CASCADE)
content = models. TextField()
```

```
timestamp = models.DateTimeField (auto_now_add=True)

def __str__(self):
 return f'Message {self.id} from {self.sender.username}'
'''
```

### Серіалізатор для повідомлень:

```
'''python
class MessageSerializer (serializers.ModelSerializer):
 sender = UserSerializer (read_only=True)

class Meta:
 model = Message
 fields = («id», «chat», «sender», «content», «timestamp»)
'''
```

### Ендпоінти для відправки та отримання повідомлень:

```
'''python
class MessageListCreateView (generics.ListCreateAPIView):
 serializer_class = MessageSerializer
 permission_classes = [IsAuthenticated]

def get_queryset (self):
 chat_id = self.kwargs [«chat_id»]
 return Message.objects.filter (chat_id=chat_id).order_by
 («timestamp»)

def perform_create (self, serializer):
 chat_id = self.kwargs [«chat_id»]
 chat = Chat.objects.get (id=chat_id)
 serializer.save (chat=chat, sender=self.request.user)
'''
```

### Маршрутизація для повідомлень:

```
'''python
urlpatterns += [
 path('chats/<int: chat_id>/messages/', MessageListCreateView.
 as_view(), name='message-list-create'),
]
'''
```

## 10. Захист API

Щоб API було безпечним, додамо обмеження доступу:

- Використаємо `IsAuthenticated`, щоб тільки авторизовані користувачі могли створювати чати та надсилати повідомлення.

- Для аутентифікації використовуємо JWT-токени.
- Дані кожного користувача будуть доступні лише для нього.

11. Підключення WebSocket для реального часу. Месенджер повинен працювати в реальному часі. Для цього використаємо Django Channels.

Додавання WebSocket:

```
```python
from channels.generic.websocket import AsyncWebsocketConsumer
import json

class ChatConsumer (AsyncWebsocketConsumer):
    async def connect (self):
        self.chat_id = self.scope['url_route']['kwargs']['chat_id']
        self.room_group_name = f'chat_{self.chat_id}'

        await self.channel_layer.group_add (self.room_group_name, self.channel_name)
        await self.accept()

    async def disconnect (self, close_code):
        await self.channel_layer.group_discard (self.room_group_name, self.channel_name)

    async def receive (self, text_data):
        data = json.loads (text_data)
        message = data ['message']
        sender = self.scope ['user'].username
        await self.channel_layer.group_send(
            self.room_group_name,
            {
                'type': 'chat_message',
                'message': message,
                'sender': sender
            }
        )
    async def chat_message (self, event):
        message = event ['message']
        sender = event ['sender']
        await self.send (text_data=json.dumps({
            'message': message,
            'sender': sender
        }))
```
```

Маршрутизація WebSocket у `routing.py`:

```
```python
from django.urls import path
from consumers import ChatConsumer

websocket_urlpatterns = [
```

```
path('ws / chat/<int: chat_id>/', ChatConsumer.as_asgi()),
]
```

3.4. Візуалізація роботи створеного застосунку

Розроблений месенджер для внутрішньої комунікації підприємства Dnipro-M має низку особливостей, які визначають його ефективність та специфіку використання:

1. Масштабованість та підтримка розподіленої інфраструктури. Месенджер здатний працювати у великих корпоративних мережах із високою кількістю користувачів. Завдяки використанню Django та WebSockets, можна ефективно обробляти одночасні запити та зменшувати затримки в обміні повідомленнями. Архітектура дозволяє масштабувати систему, додаючи сервери для балансування навантаження та розподіленого зберігання повідомлень.
2. Кібербезпека та захист корпоративних даних. У внутрішній корпоративній мережі важливо забезпечити надійний рівень безпеки комунікації між співробітниками. Впроваджено аутентифікацію через JWT-токени, шифрування повідомлень та обмеження доступу за ролями. Це мінімізує ризики витоку конфіденційної інформації та потенційних атак на серверну інфраструктуру.
3. Захист внутрішньої інформації та інтелектуальної власності. На відміну від відкритих месенджерів, цей додаток функціонує виключно у межах компанії, що запобігає несанкціонованому доступу сторонніх осіб. Передача повідомлень та файлів здійснюється у зашифрованому вигляді, а збереження даних можливе лише на внутрішніх серверах компанії, що підвищує рівень захисту корпоративної інформації.
4. Гнучкість у налаштуванні та підтримка сучасних бізнес-процесів. Месенджер адаптується під специфічні потреби компанії. Підтримка

різних форматів повідомлень, групові чати, інструменти керування завданнями та інтеграція з внутрішніми сервісами роблять систему ефективним засобом для комунікації. Крім того, реалізовано постійну підтримку та оновлення функціональності, що відповідає сучасній концепції безперервного вдосконалення програмних продуктів.

Загалом, месенджер Dnipro-M поєднує безпеку, гнучкість та продуктивність, що робить його ідеальним рішенням для внутрішньої комунікації на підприємстві (Рис. 3.15).

Оскільки, розробка корпоративного месенджера потребує особливого підходу до забезпечення стійкості, безпеки та ефективності системи, відповідно оптимізація продуктивності в нашому випадку має на меті балансування навантаження на сервер, використання WebSockets для швидкої передачі повідомлень та механізми кешування для зменшення затримок. Захист інтелектуальної власності та корпоративної інформації реалізовано через шифрування повідомлень, контроль доступу та резервне копіювання даних.

Для управління доступом до системи передбачено призначення ролей користувачів. Адміністратор месенджера має можливість керувати правами доступу через спеціальний модуль у панелі управління. У цьому розділі він може переглядати список користувачів, призначати їм ролі (наприклад, звичайний співробітник, менеджер, адміністратор), а також контролювати статуси активності. Крім того, реалізована можливість відстеження нових користувачів, які ще не підтвердили свою електронну пошту, а також тих, хто був тимчасово заблокований або деактивований через порушення політик компанії.

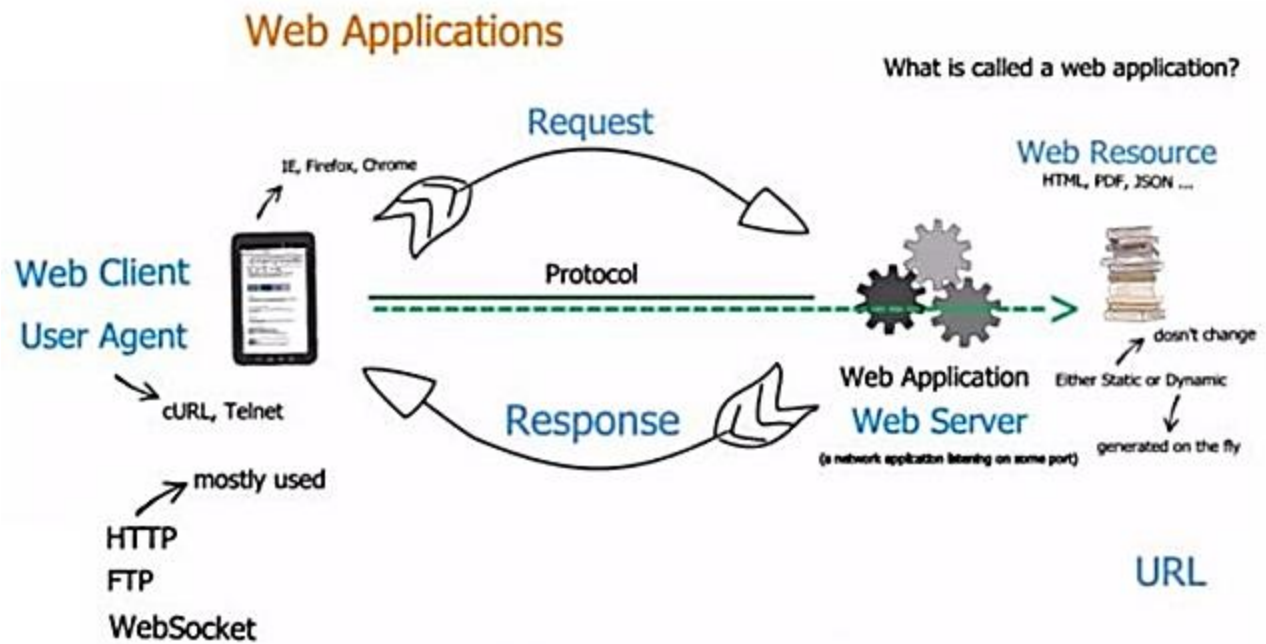


Рисунок 3.15 – Схема функціонування кросплатформенного веб-застосунку

Цей підхід забезпечує контрольований доступ до внутрішньої комунікації, запобігає несанкціонованому використанню месенджера та допомагає керівництву ефективно адмініструвати процес взаємодії між співробітниками. В то же час, адміністратор може переглянути всіх відвідувачів включно з тими, хто не підтвердив свій e-mail або ж були заблоковані (рис. 3.16).

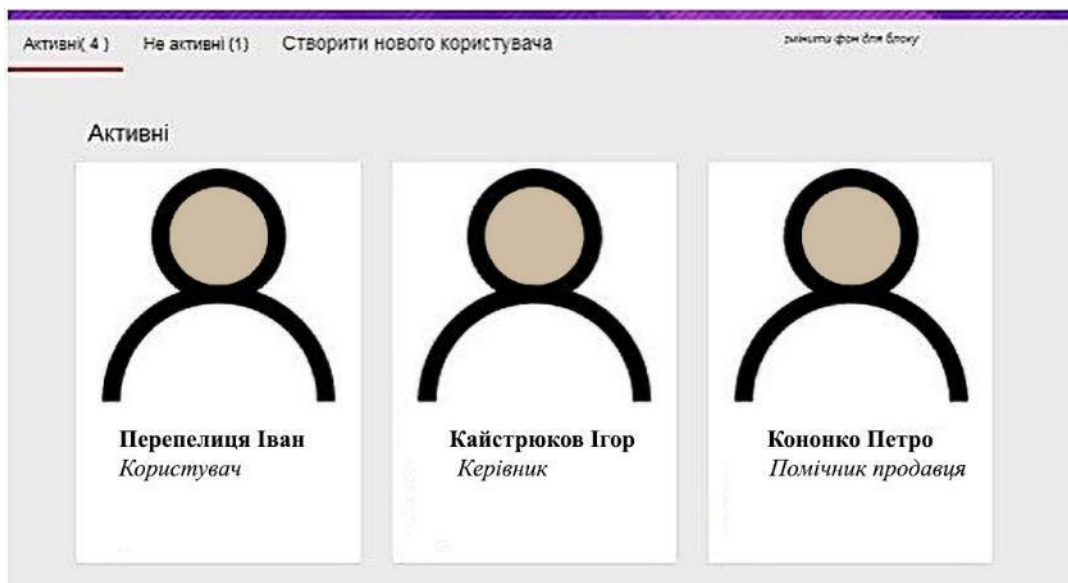


Рисунок 3.16 – Список всіх активних користувачів

Після того, як відвідувач отримав статус «в системі», він може виконати вхід і отримати доступ до свого робочого простору. Відвідувач повинен заповнити необхідну форму для входу, вказавши свій пароль та логін (Рис. 3.17).

Рисунок 3.17 – Форма авторизації користувача

Опишемо модуль авторизації у корпоративному месенджері Dnipro-M. Форма авторизації обробляється у файлі `views. ру`. Якщо користувач успішно входить у систему, його автоматично перенаправляють на головну сторінку або інший необхідний розділ. У разі невірного введення даних система генерує повідомлення про помилку. Програмний код форми наведено в Додатку Д. 2.

Алгоритм роботи модуля авторизації (рис. 3.17)

1. Користувач вводить свої реєстраційні дані (логін та пароль).
2. Відбувається HTTP-запит методом POST на сервер.
3. Контролер перевіряє валідність введених облікових даних.
4. Якщо дані коректні, здійснюється запит до бази даних для визначення ролі користувача.
5. Якщо автентифікація неуспішна, система повертає відповідне повідомлення про помилку.
6. У разі успішного входу генерується динамічна URL-адреса, і користувач перенаправляється до свого персонального кабінету.

Після входу до системи користувач отримує доступ до особистого кабінету, який містить такі ключові розділи (рис. 3.18):

- Профіль користувача – дозволяє переглядати та редагувати персональні дані, такі як ім'я, фото, контактна інформація.

- Налаштування – містить параметри конфіденційності, управління сповіщеннями, зміну пароля та інші опції.
- Створення повідомлення – можливість написати нове повідомлення або опублікувати інформацію для внутрішньої комунікації.
- Архів повідомлень – перегляд історії листування, включаючи активні, видалені та архівовані повідомлення.

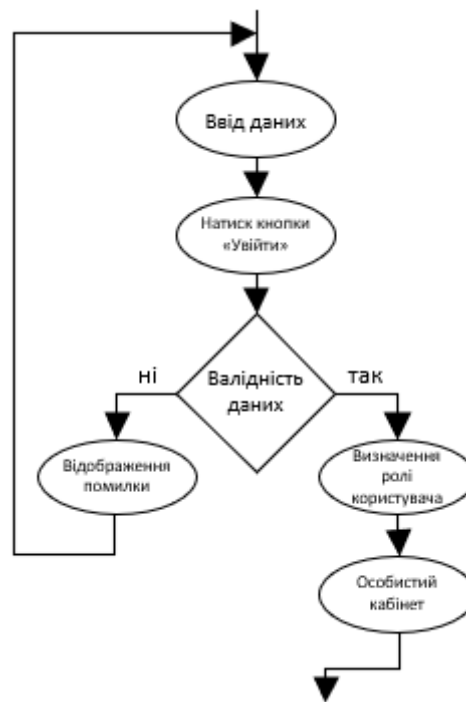


Рисунок 3.17 – Діаграма алгоритму входу користувача в систему

Такий підхід дозволяє зробити систему зручною у використанні, підвищити рівень персоналізації та забезпечити ефективну комунікацію між співробітниками підприємства.

Ці пункти навігації дозволяють користувачу легко переміщатися по особистому кабінету та виконувати необхідні дії пов'язані з керуванням своїм обліковим записом та створенням контенту (рис. 3.19).

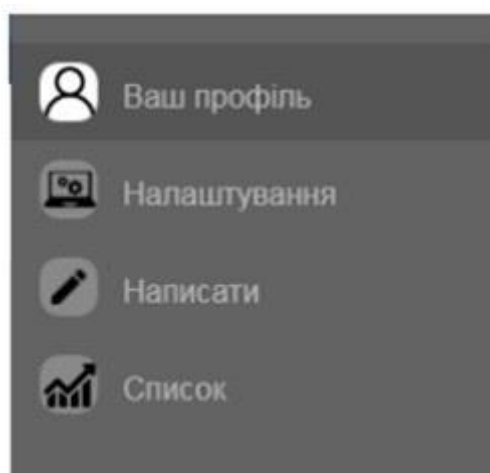


Рисунок 3.19 – Особисте меню користувача

Опишемо налаштування профілю користувача у месенджері Dnipro-M. У розділі «Мій профіль» користувач має можливість переглядати свою особисту інформацію та загальну статистику щодо взаємодії у месенджері. Для внесення змін у профіль достатньо натиснути на кнопку «Редагувати профіль».

У цьому розділі користувач може виконувати такі дії:

- Додавати посилання на профілі в соціальних мережах – це дозволяє іншим користувачам зручніше підтримувати зв'язок через альтернативні канали комунікації.
- Оновлювати аватар (фото профілю) – користувач може змінити своє зображення, яке відображається під час обміну повідомленнями та в списках контактів.
- Обирати тип відображення імені – можливо використовувати повне ім'я або псевдонім для участі в комунікації.

Такі можливості дають змогу персоналізувати профіль, додати додаткові контакти та забезпечити зручність використання месенджера.

Крім того, у будь-якому розділі веб-застосунку користувач може швидко перемикатися між чатами різних категорій, які доступні у верхньому навігаційному меню (Рис. 3.20). Це дозволяє оперативно переходити до потрібної тематики спілкування, забезпечуючи комфортну та швидку взаємодію в корпоративному середовищі.

Категорія:

Заголовок

Лід(аріз)

Теги:

Натисніть щоб відкрити медіатеку

Показати на сторінці статті
☒ Так ☐ Ні

Фото для репосту в соціальних мережах
Натисніть щоб відкрити медіатеку

Коментар до фото

Джерело

Рисунок 3.20 – Форма для створення нового спілкування за певною тематикою

На головному екрані корпоративного месенджера відображаються актуальні повідомлення, розміщені у верхньому інформаційному блоці. Завдяки механізму «lazy load» повідомлення завантажуються поступово – при прокручуванні сторінки виконується асинхронний AJAX-запит, що довантажує наступний набір даних. Це дозволяє уникнути перевантаження інтерфейсу та оптимізувати продуктивність системи. Загальна кількість завантажених повідомлень обмежена 60 елементами (Рис. 3.21).

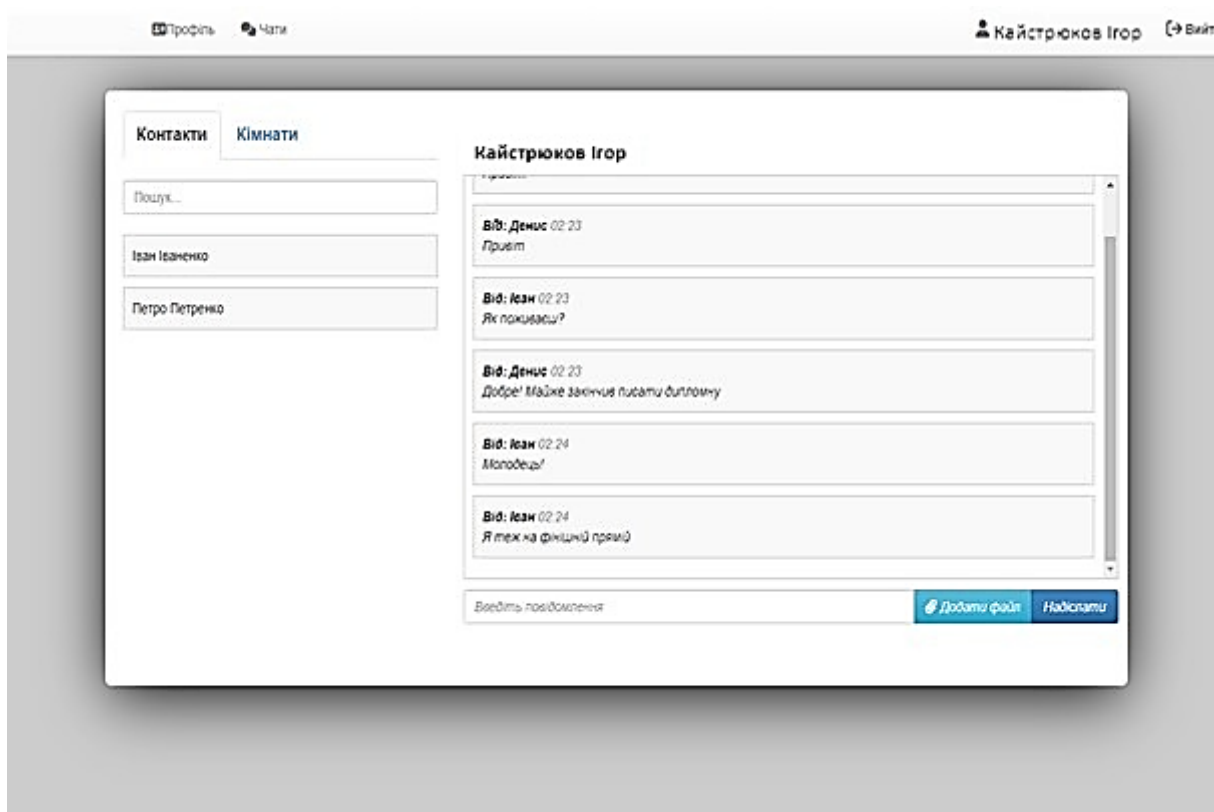


Рисунок 3.21 – Вікно спілкування співробітника

При виборі певного чату або теми обговорення, користувач переходить на окрему сторінку, де доступний повний зміст повідомлення разом із вкладеними медіа-файлами. У цьому розділі відображається інформація про автора повідомлення, його контактні дані та кількість переглядів. Крім того, у боковому блоці відображаються найпопулярніші теми за останню добу або тиждень, що можуть бути пов'язані зі змістом поточного обговорення.

Для зареєстрованих користувачів передбачена можливість коментування та взаємодії через відповіді на коментарі інших учасників дискусії. Це сприяє активному спілкуванню між співробітниками та створює інтерактивне середовище для обміну думками (Рис. 3.22).

При розробці месенджера враховано принципи кросплатформенності та адаптивного дизайну. Інтерфейс коректно відображається на всіх пристроях – від настільних ПК та ноутбуків до планшетів і смартфонів. Завдяки гнучкому масштабуванню та динамічному розташуванню елементів, система автоматично адаптується до різних розмірів екранів, забезпечуючи комфортну взаємодію користувачів із месенджером.

The screenshot displays a web interface for comments. At the top, a header reads "КОМЕНТАРІ (2)". Below this is a form for adding a new comment, consisting of a text input field with the placeholder "Залиште свій коментар..." and a blue button labeled "Коментувати". A horizontal line separates the form from a list of existing comments. The first comment is from "test_admin" at "21:59", with the text "Чудова новина!" and a "Закрити" button. The second comment is from "aa" at "22:02", with the text "Повністю згоден!" and a "Закрити" button. To the left of the second comment is a left-pointing arrow icon. Below the list of comments is another text input field with the placeholder "Залиште свій коментар...".

Рисунок 3.22 – Форма коментарів від користувачів

На **мобільних пристроях** система автоматично адаптує інтерфейс до компактних екранів, змінюючи розташування елементів для збереження зручності користування. Використання технологій **CSS Flexbox i Grid Layout**, а також медіа-запитів дозволяє динамічно змінювати розташування компонентів залежно від роздільної здатності екрана. При цьому розміри кнопок, тексту та інтерактивних елементів підлаштовуються під можливості сенсорного керування, забезпечуючи зручну взаємодію.

Для планшетів реалізовано проміжний варіант відображення, який зберігає функціональність десктопної версії, але водночас адаптується до сенсорного введення. Це дає можливість користувачам зручно переглядати контент і виконувати дії без необхідності збільшення або прокручування сторінки.

На настільних комп'ютерах і ноутбуках веб-додаток використовує розширений функціонал, включаючи багатовіконний режим, можливість

масштабування контенту та швидкі клавіші для продуктивної роботи. Завдяки підтримці кросплатформенності, доступ до сервісу можливий не лише через браузер, а й через мобільні застосунки та десктопні клієнти, які використовують веб-технології, зокрема Electron або React Native.

Такий підхід до розробки забезпечує гнучкість і доступність веб-додатку незалежно від типу пристрою, що використовує користувач. Це особливо важливо для корпоративного середовища, де співробітники можуть працювати в різних умовах – за офісним комп'ютером, на планшеті у відрядженні чи зі смартфона під час комунікації в реальному часі. Інтеграція адаптивного дизайну з технологіями кросплатформенності дозволяє досягти максимальної ефективності використання додатку, не прив'язуючи його до конкретного типу пристрою чи операційної системи.

РОЗДІЛ 4. ОХОРОНА ПРАЦІ

4.1. Загальний стан охорони праці на підприємстві «Dnipro-M»

Охорона праці на підприємстві «Dnipro-M» є ключовим напрямком корпоративної політики, спрямованої на забезпечення безпечних і комфортних умов роботи для співробітників. Як мережа магазинів, що спеціалізується на продажу різної техніки та інструментів, компанія стикається з низкою професійних ризиків, пов'язаних із роботою в торгових приміщеннях, логістичних складах та сервісних центрах. Забезпечення охорони праці передбачає виконання законодавчих норм, впровадження сучасних методів захисту працівників, навчання персоналу та використання технологій для мінімізації виробничих ризиків [11].

1. Нормативно-правова база охорони праці. Охорона праці в «Dnipro-M» регулюється відповідно до Закону України «Про охорону праці», Кодексу законів про працю, а також низки підзаконних актів, що визначають вимоги до безпеки робочого середовища, санітарно-гігієнічних норм і заходів щодо запобігання травматизму [19]. Серед основних нормативних документів, які регулюють безпечну діяльність підприємства, можна виокремити:
 - ДСТУ щодо безпечного зберігання, транспортування та використання інструментів та обладнання.
 - Державні будівельні норми (ДБН), що визначають вимоги до облаштування торгових приміщень, складських приміщень та сервісних центрів.
 - Правила пожежної безпеки в Україні (НАПБ А. 01.001–2014), що встановлюють вимоги до протипожежного захисту магазинів і складів.
 - Гігієнічні нормативи ДСП та ДСН, які регламентують рівні шуму, освітлення, вентиляції та мікроклімату на робочих місцях [19].

Відповідно до законодавства, кожен працівник перед початком роботи проходить інструктаж з охорони праці, включаючи первинний, повторний, позаплановий та цільовий.

2. Організація безпечних умов праці. На підприємстві «Dnipro-M» організовано систему управління охороною праці (СУОП), що охоплює всі рівні діяльності компанії, від магазинів до логістичних комплексів. Основними напрямками забезпечення безпеки є:

- Контроль робочого середовища: перевірка рівня шуму, освітлення, якості вентиляції та мікроклімату у приміщеннях, де працюють співробітники.
- Безпечна експлуатація обладнання: всі співробітники магазинів, складів та сервісних центрів мають доступ лише до сертифікованого інструменту, який відповідає вимогам технічних регламентів.
- Протипожежна безпека: у торгових приміщеннях встановлено автоматичні пожежні сигналізації, протипожежні двері, вогнегасники, плани евакуації та систему димовидалення. Співробітники регулярно проходять інструктаж із пожежної безпеки та навчання щодо користування засобами гасіння.
- Використання засобів індивідуального захисту (ЗІЗ): працівники складів і сервісних центрів використовують рукавиці, окуляри, захисний одяг та взуття відповідно до специфіки своєї роботи.
- Організація безпечного зберігання техніки: усі товари розміщуються у відповідності до технічних норм зберігання, що запобігає падінню важких предметів та мінімізує ризики травматизму.
- Запобігання електротравматизму: електрообладнання магазинів та складів регулярно проходить діагностику та перевірку на відповідність нормам електробезпеки [3].

3. Навчання та інструктажі персоналу. Співробітники компанії проходять комплексне навчання з охорони праці, включаючи:

- Первинний інструктаж перед початком роботи.
- Періодичні повторні інструктажі кожні 6 місяців.

4.2. Навчання з користування засобами пожежогасіння та першої медичної допомоги

- Окремі програми навчання для керівників підрозділів та співробітників, що працюють з обладнанням підвищеної небезпеки [3].

Регулярно проводяться тренування з евакуації та пожежної безпеки, що дозволяє перевірити готовність персоналу до надзвичайних ситуацій.

4. Психофізіологічний захист працівників. З метою підтримки фізичного та психологічного здоров'я працівників компанія впроваджує:

- Зменшення навантаження на персонал шляхом автоматизації процесів (впровадження CRM-систем, мобільних додатків для інвентаризації, автоматичних складських систем).
- Гнучкий графік роботи та змінний режим праці для зменшення перевтоми.
- Медичне страхування та профілактичні огляди для раннього виявлення професійних захворювань.
- Підтримку здорового мікроклімату на робочих місцях, включаючи регулювання температурного режиму, освітлення та рівня шуму [19].

5. Запобігання професійному травматизму. Для мінімізації ризиків травм на робочому місці вживаються такі заходи:

- Контроль дотримання правил техніки безпеки всіма співробітниками.
- Регулярний технічний огляд обладнання та усунення несправностей у торгових приміщеннях та на складах [19].

Завдяки цим заходам рівень травматизму на підприємстві суттєво знижений, а працівники мають змогу працювати у безпечних та комфортних умовах. Отже, охорона праці на підприємстві «Dnipro-M» базується на сучасних підходах до управління безпекою робочого середовища. Використання профілактичних заходів, навчання персоналу та автоматизації робочих процесів дозволяє підтримувати високий рівень безпеки та знижувати ризики травматизму. Подальша модернізація та цифровізація процесів забезпечить ще

ефективніший контроль за умовами праці та сприятиме створенню безпечного та комфортного робочого середовища для всіх співробітників компанії.

4.3. Питання навчання та інструктажу персоналу «Dnipro-M»

Навчання та інструктажі персоналу є важливим елементом системи охорони праці в компанії «Dnipro-M», оскільки вони забезпечують працівників знаннями та навичками, необхідними для безпечного виконання їхніх обов'язків. Усі співробітники, незалежно від їхньої посади чи досвіду, проходять комплексне навчання з охорони праці. Особлива увага приділяється первинному інструктажу, який проводиться перед початком роботи. Це дозволяє ознайомити працівників із загальними правилами безпеки, потенційними ризиками на робочому місці, а також із заходами, спрямованими на їх мінімізацію.

Періодичні повторні інструктажі, що проводяться кожні шість місяців, мають на меті оновлення знань співробітників, нагадування їм про важливість дотримання правил охорони праці та адаптацію до змін у виробничих процесах або обладнанні. Ці інструктажі допомагають забезпечити постійний рівень безпеки на робочих місцях, а також виявити та виправити можливі прогалини в знаннях працівників [19].

Окремим блоком навчання є підготовка персоналу до дій у надзвичайних ситуаціях. Це включає навчання з користування засобами пожежогасіння, що забезпечує працівників необхідними навичками для оперативного реагування у разі виникнення пожежі. Крім того, працівники компанії навчаються наданню першої медичної допомоги, що дозволяє зменшити наслідки травм або інших непередбачуваних ситуацій до приїзду медичних служб.

Для керівників підрозділів, а також співробітників, які працюють з обладнанням підвищеної небезпеки, розробляються окремі програми навчання. Вони спрямовані на більш детальне вивчення специфіки роботи з ризикованим обладнанням, а також на розвиток навичок управління безпекою праці у своїх

командах. Таке навчання дозволяє не тільки підвищити кваліфікацію працівників, але й запобігти нещасним випадкам на виробництві.

Таким чином, система навчання та інструктажів у компанії «Dnipro-M» є багаторівневою, охоплює всі категорії працівників та забезпечує високий рівень обізнаності у питаннях охорони праці. Це сприяє створенню безпечних умов праці та підтримці ефективної роботи компанії [19].

4.4. Психофізіологічний захист працівників є ключовим аспектом політики охорони праці в компанії «Dnipro-M»

Психофізіологічний захист працівників є ключовим аспектом політики охорони праці в компанії «Dnipro-M», оскільки здоров'я та добробут співробітників мають прямий вплив на продуктивність і якість роботи. Для підтримки фізичного та психологічного здоров'я працівників компанія впроваджує комплексний підхід, що охоплює організаційні, технічні та соціальні заходи.

Одним із головних напрямків є зменшення фізичного та розумового навантаження на персонал шляхом автоматизації виробничих і адміністративних процесів. Впровадження сучасних технологій, таких як CRM-системи, мобільні додатки для інвентаризації та автоматичні складські системи, значно спрощує рутинні операції, скорочуючи кількість ручної праці. Це дозволяє співробітникам зосередитися на більш складних і креативних завданнях, знижуючи ризик перевтоми та стресу. Автоматизація також сприяє підвищенню точності виконання робіт, що зменшує ймовірність помилок і, відповідно, стресових ситуацій, пов'язаних із їх виправленням [3].

Гнучкий графік роботи є ще одним важливим елементом психофізіологічного захисту. Компанія надає можливість працівникам обирати змінний режим праці або працювати за індивідуальним графіком, що дозволяє краще балансувати професійні та особисті обов'язки. Такий підхід мінімізує ризики перевтоми, сприяє зниженню рівня стресу та покращує загальну якість

життя працівників. Гнучкість у плануванні робочого часу також дозволяє уникнути пікових навантажень, які можуть призводити до професійного вигорання.

Охорона здоров'я працівників підтримується через впровадження медичного страхування та регулярні профілактичні огляди. Медичне страхування надає працівникам доступ до якісної медичної допомоги, включаючи лікування та профілактику захворювань. Регулярні медичні огляди спрямовані на раннє виявлення професійних захворювань, що дозволяє вчасно запобігти їхньому розвитку та мінімізувати негативний вплив на працездатність співробітників. Окрім того, компанія організовує інформаційні кампанії з питань здоров'я, спрямовані на підвищення обізнаності працівників щодо важливості профілактики захворювань і ведення здорового способу життя [3].

Особлива увага приділяється створенню комфортного мікроклімату на робочих місцях. Це включає регулювання температурного режиму відповідно до сезонних умов, забезпечення належного рівня освітлення для зменшення навантаження на зір, а також контроль рівня шуму, щоб уникнути зайвого подразнення та стресу. Для забезпечення оптимального повітрообміну використовуються сучасні системи вентиляції, які сприяють збереженню високої якості повітря в приміщеннях.

Крім того, компанія «Dnipro-M» сприяє формуванню позитивного психологічного клімату в колективі, організовуючи різноманітні тренінги, командні заходи та консультації з психологами. Це дозволяє працівникам не лише ефективніше співпрацювати, але й знаходити емоційну підтримку у разі виникнення стресових ситуацій. Усе це сприяє зміцненню корпоративної культури, що базується на взаємній довірі та підтримці [3].

Таким чином, комплекс заходів із психофізіологічного захисту в компанії «Dnipro-M» забезпечує створення сприятливих умов для роботи, які відповідають сучасним стандартам охорони праці. Це дозволяє не лише підтримувати фізичне та психологічне здоров'я співробітників, а й підвищувати

їхню задоволеність роботою, що позитивно впливає на ефективність діяльності компанії загалом.

4.5. Запобігання професійному травматизму компанії «Dnipro-M»

Щодо запобігання професійному травматизму, то воно є одним із пріоритетних напрямів діяльності компанії «Dnipro-M» у сфері охорони праці. Для забезпечення безпечних умов роботи впроваджується низка заходів, спрямованих на мінімізацію ризиків травм на робочих місцях, що охоплюють як організаційні, так і технічні аспекти.

Особлива увага приділяється контролю за дотриманням правил техніки безпеки всіма співробітниками. Компанія активно проводить інформаційно-освітню роботу, забезпечуючи працівників детальними інструкціями та рекомендаціями щодо безпечного виконання їхніх обов'язків. Усі співробітники проходять обов'язкові інструктажі з техніки безпеки, що включають ознайомлення з потенційними ризиками на їхніх робочих місцях, а також алгоритми дій у випадку виникнення небезпечних ситуацій. Для підвищення ефективності контролю створено систему перевірок, під час яких оцінюється, наскільки працівники дотримуються встановлених норм і стандартів безпеки.

Окрім цього, проводиться регулярний технічний огляд обладнання, яке використовується в торгових приміщеннях і на складах. Інженерно-технічний персонал компанії забезпечує своєчасне обслуговування та ремонт устаткування, щоб запобігти його несправності. Впроваджено графіки періодичного огляду обладнання, які враховують особливості його експлуатації та ризики зношення. Особлива увага приділяється стану підйомного та вантажного обладнання, яке використовується на складах, а також торгового інвентарю, який застосовується в магазинах. У разі виявлення несправностей їх усувають до початку використання обладнання, що дозволяє мінімізувати ризик травм.

Компанія також забезпечує працівників засобами індивідуального захисту (ЗІЗ), відповідно до специфіки їхньої діяльності. Наприклад, для співробітників

складів це можуть бути каски, захисні рукавиці, окуляри та взуття зі стійкими підсошвами. Такі засоби створюють додатковий бар'єр, що захищає працівників від можливих травм під час виконання їхніх обов'язків [19].

Важливим елементом профілактики професійного травматизму є моніторинг і аналіз потенційно небезпечних зон у торгових приміщеннях і на складах. Наприклад, компанія регулярно перевіряє наявність належного освітлення в приміщеннях, стан підлогового покриття та забезпечення вільного проходу у зонах евакуації. Застосування спеціальних знаків безпеки та маркування ризикованих ділянок дозволяє зменшити ймовірність виникнення аварійних ситуацій.

У випадку, якщо нещасний випадок все ж таки стався, компанія має чіткий алгоритм дій для його розслідування та виявлення причин. Це включає збір інформації, аналіз обставин та розробку рекомендацій для запобігання подібним інцидентам у майбутньому. Висновки з кожного випадку використовуються для вдосконалення існуючих процедур безпеки та навчання персоналу [19].

Таким чином, комплексний підхід до запобігання професійному травматизму, який застосовується компанією «Dnipro-M», спрямований на створення безпечного робочого середовища для співробітників. Це не лише знижує ризик травм, але й сприяє формуванню культури безпеки, яка є невід'ємною частиною корпоративної політики. Завдяки цьому компанія забезпечує високу якість виконання робіт та підтримує довіру працівників до свого роботодавця [19].

4.6. Правила поводження з електроінструментом в торгових точках компанії Dnipro-M

Правила поводження з електроінструментом, який продається в торгових точках компанії Dnipro-M, є важливою частиною забезпечення безпеки як для працівників компанії, так і для клієнтів, які взаємодіють з цим обладнанням. Оскільки електроінструмент є джерелом підвищеної небезпеки, компанія

впровадила чіткі інструкції щодо його зберігання, демонстрації, обслуговування та експлуатації [3].

Зберігання електроінструменту здійснюється в умовах, що виключають можливість його випадкового пошкодження або несанкціонованого використання. Інструменти розміщуються на спеціально облаштованих стендах або у закритих шафах із забезпеченням належного рівня захисту від пилу, вологи та механічних пошкоджень. У зонах зберігання передбачено позначки та вказівки, які вказують на специфічні вимоги до кожного типу інструменту, наприклад, рекомендації щодо температурного режиму або уникнення прямого впливу сонячних променів.

Демонстрація електроінструменту для клієнтів здійснюється відповідно до суворих стандартів безпеки. Під час презентації працівники магазину зобов'язані використовувати справне обладнання та демонструвати його можливості лише у спеціально відведених для цього зонах, обладнаних захисними екранами або обмежувачами. Перед початком демонстрації співробітники проводять візуальний огляд інструменту на наявність пошкоджень кабелю живлення, корпусу чи інших елементів, які можуть впливати на безпеку. Клієнтам надаються чіткі вказівки щодо спостереження за демонстрацією з безпечної відстані, щоб уникнути можливих травм [3].

Обслуговування інструментів є обов'язковою складовою забезпечення їхньої безпеки. Регулярно проводиться технічний огляд кожної одиниці обладнання, що включає перевірку роботи двигуна, стану електричних з'єднань та ізоляції. У разі виявлення несправностей електроінструмент знімається з продажу до моменту повного усунення дефектів. Обслуговування проводиться виключно кваліфікованим персоналом, який має відповідну підготовку та допуск до роботи з електричним обладнанням.

Експлуатація електроінструменту під час демонстрації або перевірки роботи на прохання клієнта здійснюється з дотриманням усіх норм охорони праці. Перед початком використання співробітники забезпечують наявність засобів індивідуального захисту, таких як захисні окуляри або рукавиці, залежно

від типу інструменту. Крім того, обов'язково перевіряється цілісність кабелю живлення та справність вимикачів, а також забезпечується підключення інструменту до джерела живлення через захисні пристрої, наприклад, УЗО (пристрій захисного відключення).

Для клієнтів, які купують електроінструмент, компанія забезпечує їх необхідною інформацією щодо правил безпечної експлуатації. Кожен інструмент постачається з детальною інструкцією, у якій зазначено, як безпечно його використовувати, зберігати та обслуговувати. Додатково в торгових точках співробітники надають усні рекомендації щодо використання обладнання, зокрема попереджають про можливі ризики при недотриманні правил безпеки.

У компанії також проводяться регулярні тренінги для працівників торгових точок, спрямовані на підвищення обізнаності щодо правил поводження з електроінструментом. Це дозволяє їм не лише забезпечувати власну безпеку, але й якісно консулювати клієнтів, надаючи їм всю необхідну інформацію про використання обладнання [3].

Таким чином, правила поводження з електроінструментом у торгових точках Dnipro-M спрямовані на забезпечення максимальної безпеки та зниження ризиків для всіх учасників процесу. Завдяки дотриманню цих правил компанія не лише захищає своїх працівників і клієнтів, але й демонструє високий рівень відповідальності та професіоналізму у своїй діяльності.

ВИСНОВКИ

У ході дослідження було детально проаналізовано концепцію веб-застосунків, які є інтеграцією різних способів представлення інформації. Веб-додатки об'єднують текст, звук, графіку, відеозображення та інтерактивні елементи, що дозволяє користувачам ефективно взаємодіяти з системою. Оскільки розробка веб-застосунків передбачає структуроване планування процесів, у межах дослідження було побудовано сценарій створення корпоративного месенджера, який охоплює всі етапи його проектування, реалізації та тестування.

Особливу увагу було приділено питанню **кросплатформенності**, оскільки сучасні користувачі взаємодіють із програмним забезпеченням з різних пристроїв – настільних комп'ютерів, мобільних телефонів, планшетів та ноутбуків. Для забезпечення доступності веб-додатку на різних платформах використано технології адаптивного дизайну та кросплатформенних фреймворків, що дає змогу запускати месенджер на **Windows, macOS, Linux, Android та iOS** без значних змін у коді.

На сьогоднішній день найбільш популярними платформами для створення мобільних застосунків є **Android та iOS**, оскільки вони забезпечують розширений функціонал і високу продуктивність. Обидві платформи надають безкоштовні та потужні засоби для розробки, що сприяє швидкому оновленню додатків і впровадженню нових можливостей.

Окрім мобільних платформ, розглянуто варіанти **серверної та веб-розробки**. Для створення надійної серверної частини було використано **фреймворк Django (Python)**, який забезпечує швидку розробку та високий рівень безпеки. Крім Django, було розглянуто альтернативні технології, такі як **Spring (Java), Yii (PHP), Laravel (PHP), Ruby on Rails (Ruby)**, які також активно використовуються для створення веб-систем інтерактивного спілкування.

Розроблений веб-додаток є адаптивним і оптимізованим для всіх типів пристроїв, що забезпечує зручний доступ до комунікаційних функцій незалежно від екрану, операційної системи чи потужності пристрою. Для цього використано сучасні фронтенд-технології (React.js, Vue.js, Bootstrap, CSS Flexbox та Grid Layout), що дають змогу динамічно змінювати інтерфейс відповідно до параметрів пристрою.

На настільних комп'ютерах і ноутбуках месенджер функціонує через браузер, а також у вигляді десктопного клієнта, створеного на основі Electron, що дозволяє користувачам працювати з додатком без необхідності постійного використання браузера.

На мобільних пристроях використання React Native забезпечує ефективну роботу месенджера на Android та iOS, мінімізуючи витрати на розробку окремих нативних застосунків для кожної платформи.

У практичній частині дослідження розглянуто етапи розробки корпоративного месенджера з використанням фреймворку Django. Проведено структурний аналіз, що включає формування функціональних та нефункціональних вимог, розроблено діаграму класів та цілей веб-застосунку, що демонструє взаємодію між його компонентами.

Програмна реалізація веб-застосунку передбачає такі можливості:

- Обмін текстовими повідомленнями у реальному часі
- Групові чати та створення робочих каналів
- Обмін файлами та документами з доступом через хмарні сервіси
- Інтеграція з календарями та внутрішніми корпоративними системами
- Нотифікації та управління статусами користувачів

Тестування месенджера включало функціональне, навантажувальне та безпекове тестування, що дало змогу перевірити стабільність роботи системи та її готовність до використання у корпоративному середовищі.

Проведене дослідження та реалізація прототипу корпоративного месенджера демонструють ефективність впровадження кросплатформених

рішень для внутрішньої комунікації підприємств. Основними перевагами запропонованої системи є:

- Гнучкість у використанні на будь-яких пристроях та операційних системах
- Високий рівень безпеки завдяки впровадженню наскрізного шифрування
- Інтеграція з корпоративними системами для підвищення продуктивності
- Адаптивний інтерфейс, що забезпечує зручність роботи на всіх видах пристроїв

Подальший розвиток месенджера передбачає впровадження штучного інтелекту (AI) для автоматизації рутинних процесів, інтеграцію ботів та голосових команд, а також розширення функціоналу управління завданнями та проєктами.

Загалом, результати дослідження підтверджують актуальність використання кросплатформених технологій у корпоративній комунікації. Запропонований підхід забезпечує гнучкість, масштабованість та безпеку інформаційного середовища, що є ключовими факторами для підвищення ефективності внутрішньої взаємодії співробітників підприємства **«Dnipro-M»**.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Азарян А. А., Карабут Н. О. Основи алгоритмізації та програмування: Навчальний посібник. Кривий Ріг: Освіта, 2014. 308 с.
2. Алгоритми та структури даних / уклад. О. В. Щербаков, Ю. Є. Парфьонов, В. М. Федорченко. Київ: Освіта, 2017. 58 с.
3. Безпека праці під час використання електроінструменту. Охорона праці та пожежна безпека. 2024. URL: <https://oppb.com.ua/news/bezpeka-praci-pid-chas-vykorystannya-elektroinstrumentu> (дата звернення: 21.01.2025).
4. Беллиньясо М. Розробка Web-додатків у середовищі Visual Studio Code: Website Programming. Київ: Освіта, 2007. 640 с.
5. Боднарчук І. О. Експертна система проектування архітектури програмного забезпечення / О. Г. Харченко, І. О. Боднарчук, В. В. Яцишин. Київ, 2013. 210 с.
6. Брайн Харді, Білл Філіпс. Android. Програмування для професіоналів. Київ: Освіта, 2016. 640 с.
7. Веб-фреймворк. URL: <https://www.goodfirms.co/glossary/web-framework/> (дата звернення: 21.01.2025).
8. Використання Spring Framework для розробки веб-додатків. ResearchGate. URL: https://www.researchgate.net/publication/371178975_Using_Spring_Framework_to_develop_web_applications (дата звернення: 20.01.2025).
9. Від Hello World до Spring Web MVC і до чого тут сервлети. JavaRush. URL: <https://javarush.com/ua/groups/posts/uk.1956.vd-hello-world-do-spring-web-mvc-do-chogo-tut-servleti> (дата звернення: 20.01.2025).
10. Глібовець Н. М. Створення рекомендаційної системи навчального типу з використанням фреймворку. Проблеми інтелектуалізації комп'ютера. Київ: Освіта, 2012. 176 с.
11. Головні вимоги безпеки праці на об'єктах роздрібної торгівлі. Безпека праці в Україні. 2023. URL: <https://bezpeka.isu.net.ua/news/534574-holovni-vymohy-bezpeky-pratsi-na-obyektakh-rozdribnoyi-torhivli> (дата звернення: 21.01.2025).

12. Дронов В. А. Django – практика створення Web-сайту на Python. Київ: Освіта, 2016. 528 с.
13. Метод багатокритеріальної оптимізації програмної архітектури на основі аналізу компромісів / Харченко О. Г., Боднарчук І. О., Галай І. О. *Інженерія програмного забезпечення*. 2012. № 3–4 (11–12). С. 5–11.
14. Назаров Д. Є., Драничников П. С. Клієнт-серверна взаємодія інтелектуальних мобільних пристроїв. Комп'ютерні та інформаційні науки. Київ: Освіта, 2001. С. 120–220.
15. Огляд REST. Частина 3: створення RESTful сервісу на Spring Boot. JavaRush. URL: <https://javarush.com/ua/groups/posts/uk.2488.ogljad-rest-chastina-3-stvorennja-restful-servsu-na-spring-boot> (дата звернення: 20.01.2025).
16. Опис стандарту IDEF0. URL: http://syque.com/quality_tools/tools/Tools19.htm (дата звернення: 21.01.2025).
17. Офіційна документація фреймворку UIKit. URL: https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIKit_Framework (дата звернення: 21.01.2025).
18. Порівняння фреймворків. URL: <https://www.educba.com/laravel-vs-ruby-on-rails/?source=leftnav> (дата звернення: 21.01.2025).
19. Правила охорони праці для об'єктів роздрібної торгівлі. Онлайн-бібліотека нормативних документів Budstandart. URL: https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=21927 (дата звернення: 21.01.2025).
20. Правила охорони праці під час роботи з інструментом та пристроями. Верховна Рада України. URL: <https://zakon.rada.gov.ua/go/z0327-14> (дата звернення: 21.01.2025).
21. Рахімов Б. К. Оптимальний алгоритм взаємодії інформаційного ресурсу з мобільними додатками. *Молодий вчений*. 2016. № 4 (108). С. 158–162.
22. Реальний приклад використання Spring Global Lock. DOU. URL: <https://dou.ua/lenta/articles/using-spring-global-lock> (дата звернення: 20.01.2025).

23. Розробка месенджера з використанням Spring framework, ReactJS. Національний університет «Києво-Могилянська академія». URL: <https://ezp.ukma.edu.ua/bitstreams/fc510385-bf18-492f-b0a8-e93c8d03f19a/download> (дата звернення: 20.01.2025).
24. Форсьє Д. Django. Розробка веб-додатків на Python. Київ: Освіта, 2010. 456 с.
25. Фреймворк Laravel. URL: <https://laravel.com/docs/8.x/facades> (дата звернення: 21.01.2025).
26. Фреймворк Laravel. URL: <https://medium.com/@ronak8036/op-features-of-laravel-php-framework-9082ac9eb4f2> (дата звернення: 21.01.2025).
27. Фреймворк Laravel. URL: <https://solidbrain.com.ua/ua/news/rozrobka-php-s-laravel-porivniannia-pliusiv-i-minysiv> (дата звернення: 21.01.2025).
28. Фреймворк Yii. URL: <https://www.yiiframework.com/doc/guide/2.0/ru/structure-overview> (дата звернення: 21.01.2025).
29. Фреймворк. URL: <https://cocosolution.com/en/what-is-a-framework/> (дата звернення: 21.01.2025).
30. Фреймворк. URL: <https://techterms.com/definition/framework> (дата звернення: 21.01.2025).
31. Харченко О. Г. Експертна система проектування архітектури програмного забезпечення. *Комп'ютерні технології друкарства*. № 29. 2013. С. 10–26.
32. Java. URL: <https://www.theserverside.com/definition/Java> (дата звернення: 21.01.2025).
33. Ruby. URL: <https://www.geeksforgeeks.org/ruby-introduction-to-multi-threading/?ref=rp> (дата звернення: 21.01.2025).

ДОДАТКИ

ДОДАТОК А

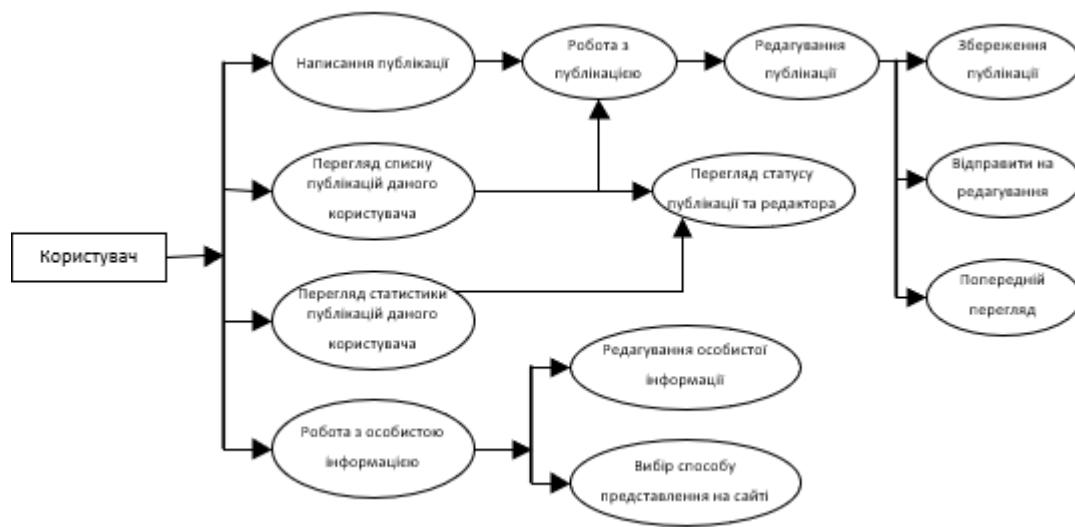


Рисунок А.1 – UML-діаграма можливостей співробітника в системі

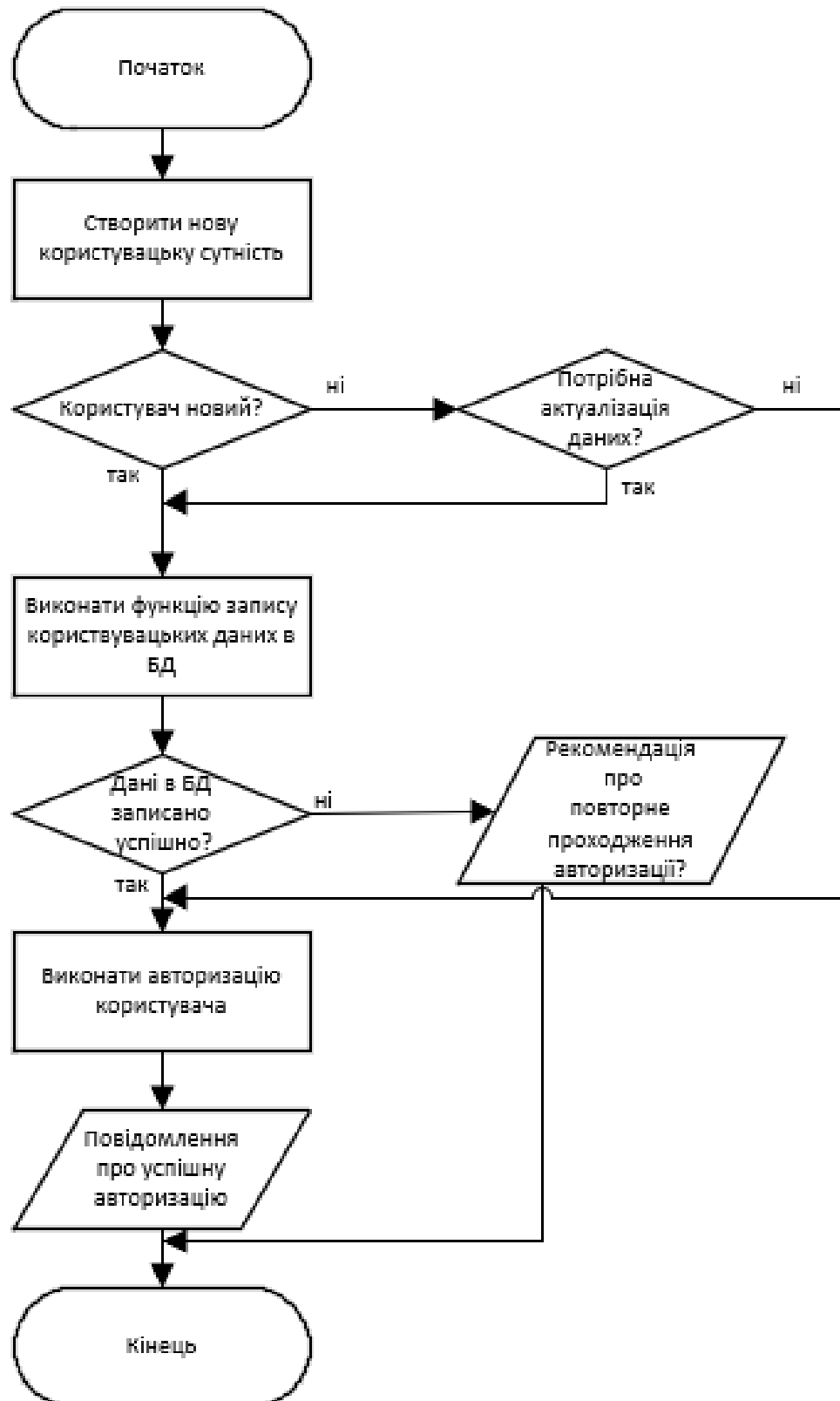


Рисунок Б.1 – Алгоритм роботи модуля обробки нового користувача

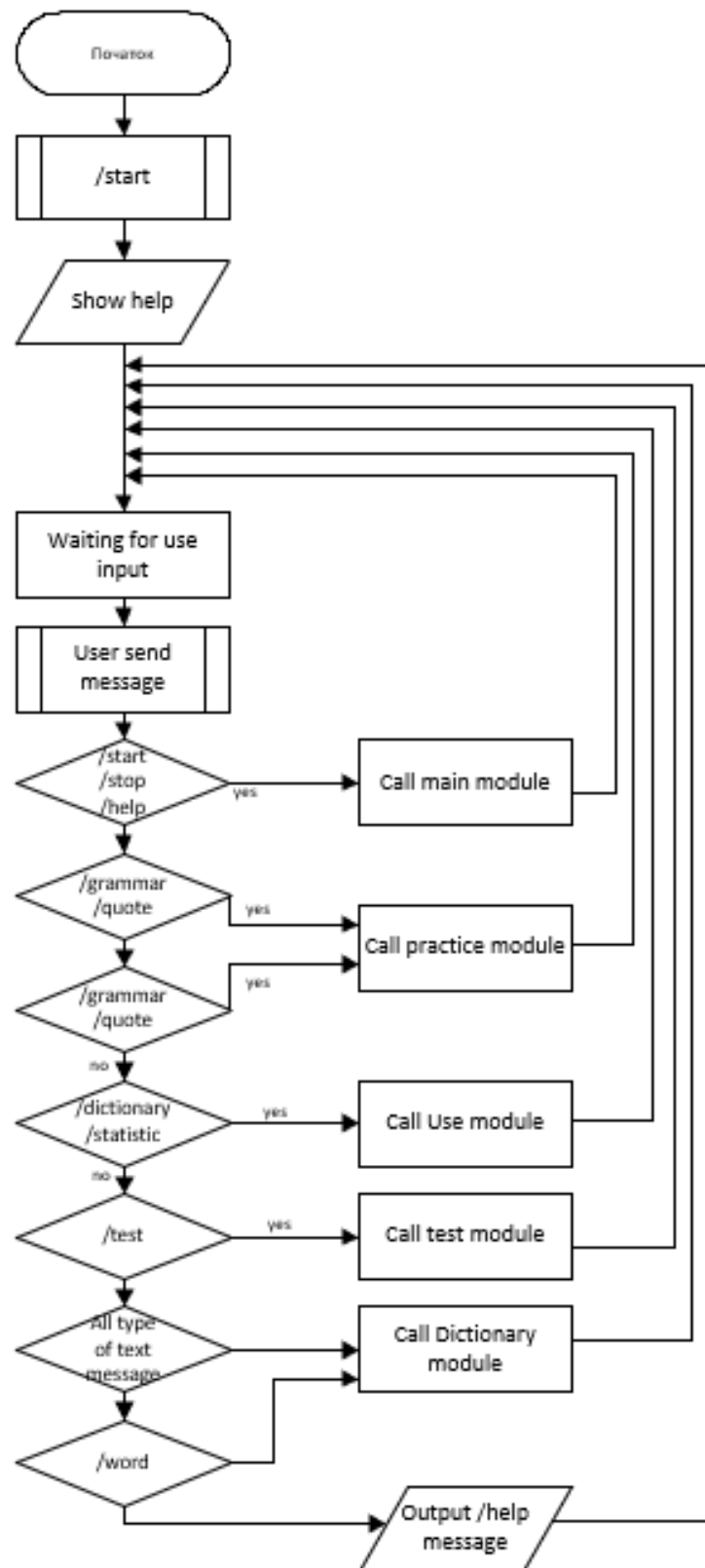


Рисунок В.1 – Командна мапа користувача

Код для модулю model.py

```

from django.db import models
from django.contrib.auth.models import User
from django.utils import timezone
from ckeditor.fields import RichTextField
from ckeditor_uploader.fields import RichTextUploadingField
from django.conf import settings

# Status options for sites
STATUS_SITE = (
    ('safe', 'Надійне джерело'),
    ('unsafe', 'Ненадійне джерело'),
)

# Status options for articles
STATUS_ARTICLE = (
    ('inactive', 'Не опрацьована'),
    ('edited', 'На опрацюванні'),
    ('to_main_admin', 'У головного редактора'),
    ('back_to_user', 'Повернено автору'),
    ('done', 'В публікації'),
)

class Category (models.Model):
    name = models.CharField (max_length=100, unique=True)
    description = models.TextField (blank=True)
    created_at = models.DateTimeField (auto_now_add=True)
    updated_at = models.DateTimeField (auto_now=True)

    def __str__(self):
        return self.name

class Article (models.Model):
    title = models.CharField (max_length=255)
    content = RichTextUploadingField()
    user = models.ForeignKey (User, on_delete=models.CASCADE)
    category = models.ForeignKey (Category, on_delete=models.SET_NULL, null=True, blank=True)
    status = models.CharField (max_length=50, choices=STATUS_ARTICLE, default='inactive')
    created_at = models.DateTimeField (auto_now_add=True)
    updated_at = models.DateTimeField (auto_now=True)

    def __str__(self):
        return self.title

    def publish (self):
        self.status = 'done'
        self.save()

```

```

class Photo (models. Model):
    title = models. CharField (max_length=255, blank=True)
    file = models. ImageField (upload_to='photos_uploads/')
    uploaded_at = models. DateTimeField (auto_now_add=True)

    def delete (self, *args, kwargs):
        # Delete the file from storage before deleting the model
        storage, path = self. file. storage, self. file. path
        super (Photo, self). delete(*args, kwargs)
        storage. delete (path)

    def __str__(self):
        return self. title or f'Photo {self. id}»

class Comment (models. Model):
    article = models. ForeignKey (Article, on_delete=models. CASCADE,
        related_name='comments')
    user = models. ForeignKey (User, on_delete=models. SET_NULL,
        null=True)
    content = models. TextField()
    created_at = models. DateTimeField (auto_now_add=True)
    updated_at = models. DateTimeField (auto_now=True)

    def __str__(self):
        return f'Comment by {self. user} on {self. article}»

class CompanyEmployee (models. Model):
    user = models. OneToOneField (User, on_delete=models. CASCADE)
    position = models. CharField (max_length=100)
    department = models. CharField (max_length=100)
    date_joined = models. DateTimeField (auto_now_add=True)

    def __str__(self):
        return f»{self. user. username} ({self. position})»

class Log (models. Model):
    action = models. CharField (max_length=255)
    user = models. ForeignKey (User, on_delete=models. SET_NULL,
        null=True, blank=True)
    timestamp = models. DateTimeField (auto_now_add=True)

    def __str__(self):
        return f»{self. action} by {self. user or 'System'} at {self.
        timestamp}»

# Example of future-proofing for deleted users
def get_deleted_user():
    return User. objects. get_or_create (username='deleted') [0]

class Feedback (models. Model):
    user = models. ForeignKey (User, on_delete=models. SET
        (get_deleted_user))

```

```

message = models. TextField()
submitted_at = models. DateTimeField (auto_now_add=True)

def __str__(self):
return f'Feedback from {self. user. username}»

# For tagging articles
class Tag (models. Model):
name = models. CharField (max_length=50, unique=True)
created_at = models. DateTimeField (auto_now_add=True)

def __str__(self):
return self. name

class ArticleTag (models. Model):
article = models. ForeignKey (Article, on_delete=models. CASCADE,
related_name='tags')
tag = models. ForeignKey (Tag, on_delete=models. CASCADE,
related_name='articles')

def __str__(self):
return f»{self. article. title} - {self. tag. name}»

class ThematicRoom (models. Model):
name = models. CharField (max_length=100, unique=True)
description = models. TextField (blank=True)
created_by = models. ForeignKey (User, on_delete=models. SET
(get_deleted_user))
created_at = models. DateTimeField (auto_now_add=True)
members = models. ManyToManyField (User,
related_name='thematic_rooms')

def __str__(self):
return self. name

def add_member (self, user):
self. members. add (user)
self. save()

def remove_member (self, user):
self. members. remove (user)
self. save()

class RoomMessage (models. Model):
room = models. ForeignKey (ThematicRoom, on_delete=models.
CASCADE, related_name='messages')
sender = models. ForeignKey (User, on_delete=models. SET
(get_deleted_user))
content = models. TextField()
timestamp = models. DateTimeField (auto_now_add=True)

def __str__(self):
return f'Message from {self. sender} in {self. room}"

```

ДОДАТОК Д

Блок, що відповідає за створення тематичних кімнат

```
class ThematicRoom (models. Model):
    name = models. CharField (max_length=100, unique=True)
    description = models. TextField (blank=True)
    created_by = models. ForeignKey (User, on_delete=models. SET
    (get_deleted_user))
    created_at = models. DateTimeField (auto_now_add=True)
    members = models. ManyToManyField (User,
    related_name='thematic_rooms')

    def __str__(self):
        return self. name

    def add_member (self, user):
        self. members. add (user)
        self. save()

    def remove_member (self, user):
        self. members. remove (user)
        self. save()

class RoomMessage (models. Model):
    room = models. ForeignKey (ThematicRoom, on_delete=models.
    CASCADE, related_name='messages')
    sender = models. ForeignKey (User, on_delete=models. SET
    (get_deleted_user))
    content = models. TextField()
    timestamp = models. DateTimeField (auto_now_add=True)

    def __str__(self):
        return f'Message from {self. sender} in {self. room}»
```

Блок, що відповідає за авторизацію у формі входу

```
from django import forms
from django.contrib.auth.forms import AuthenticationForm
from django.contrib.auth import authenticate
from django.core.exceptions import ValidationError

class LoginForm (AuthenticationForm):
    username = forms.CharField(
        label=«Ім'я користувача»,
        max_length=150,
        widget=forms.TextInput (attrs={
            'class': 'form-control',
            'placeholder': «Введіть ім'я користувача»,
        })
    )
    password = forms.CharField(
        label=«Пароль»,
        widget=forms.PasswordInput (attrs={
            'class': 'form-control',
            'placeholder': 'Введіть пароль',
        })
    )

    def clean (self):
        cleaned_data = super(). clean()
        username = cleaned_data. get('username')
        password = cleaned_data. get('password')

        if username and password:
            user = authenticate (username=username, password=password)
            if user is None:
                raise ValidationError («Неправильне ім'я користувача або пароль.»)
            if not user. is_active:
                raise ValidationError («Обліковий запис деактивований.»)
            return cleaned_data
```

Використання у views. py:

```
from django.shortcuts import render, redirect
from django.contrib.auth import login
from django.contrib import messages
from . forms import LoginForm

def user_login (request):
    if request. method == 'POST':
        form = LoginForm (request, data=request. POST)
        if form. is_valid():
```

```

user = form.get_user()
login (request, user)
messages.success (request, «Ви успішно увійшли в систему!»)
return redirect('home') # Замініть 'home' на потрібну сторінку
else:
messages.error (request, «Помилка авторизації. Перевірте дані.»)
else:
form = LoginForm()

return render (request, 'login.html', {'form': form})

```

Шаблон login.html:

```

<!DOCTYPE html>
<html lang=«uk»>
<head>
<title>Вхід</title>
</head>
<body>
<h2>Вхід у систему</h2>
<form method=«post»>
{% csrf_token %}
{{ form.as_p }}
<button type=«submit»>Увійти</button>
</form>
</body>
</html>

```

Форма обробляється у views.py. У разі успішного входу користувача перенаправляють на головну сторінку або іншу потрібну. У разі помилок, вони відображаються у вигляді повідомлень.