

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, назва факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавра

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ КЕРУВАННЯ НОТАТКАМИ З
РОЗДІЛЬНИМ ДОСТУПОМ
SOFTWARE FOR MANAGING NOTES WITH DISCRETIONARY ACCESS

Виконав: студент(ка) 4 курсу, групи КНТ-127
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

Маслов Д.О.

(прізвище та ініціали)

Керівник Скрупський С.Ю.

(прізвище та ініціали)

Рецензент Зеленьова І.Я.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти бакалавр
 Спеціальність 121 Інженерія програмного забезпечення
 (код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Маслова Данила Олександровича
 (прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення керування нотатками з роздільним доступом
Software for Managing Notes with Discretionary Access

керівник проєкту (роботи) Скрупський Степан Юрійович, к.т.н., доцент,
 (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “30” березня 2021 року № 103

2. Строк подання студентом проєкту (роботи) травень 2021 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

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

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Скрупський С.Ю., доцент		
Нормоконтролер	Липовець М.В., асистент		

7. Дата видачі завдання “9” березня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2-3 тижні	Розділ 1
3	Розробка архітектури програми	4 тиждень	Розділ 2
4	Розробка програми	5-6 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми	7 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	8 тиждень	Додатки
7	Захист роботи	9 тиждень	

Студент(ка)

(підпис) Маслов Д.О.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Скрупський С.Ю.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 89 с., 33 рис., 6 табл., 3 дод., 15 джерел.

КЕРУВАННЯ НОТАТКАМИ, РОЗДІЛЬНИЙ ДОСТУП, ВЕБЗАСТОСУНОК, PYTHON, DJANGO, РЕЛЯЦІЙНА БАЗА ДАНИХ.

Об'єкт дослідження – процес керування нотатками. Предмет дослідження – програмне забезпечення керування нотатками.

Мета роботи – розроблення програмного забезпечення керування нотатками для забезпечення роздільного та спільного доступу користувачів.

Матеріали, методи та технічні засоби: об'єктно-орієнтоване програмування, мова програмування Python, вебфреймворк Django, система керування базами даних MySQL.

Результати. Створено вебзастосунок, який за допомогою віддаленого доступу до сервера дозволяє універсуалізувати доступ до нотаток, надаючи його через пристрої різного типу на основі використання веббраузера, а також розділяючи рівень доступу на індивідуальний, розділений та загальний.

Висновки. Розроблено програмне забезпечення, що дозволяє виконувати створення, перегляд, редагування, розфарбовування, вилучення та пошук нотаток, визначення, створення тегів, відображення нотаток за тегом, всіх власних нотаток, відкриття зовнішнього доступу до нотатки для всіх користувачів, за посиланням, відображення нотаток за календарем за датою створення і виконання.

Галузь використання – збереження ідей, думок та інформації у текстовому вигляді для індивідуального та спільного використання.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 89 pages, 33 figures, 6 tables, 3 appendixes, 15 sources.

NOTE MANAGEMENT, SEPARATE ACCESS, WEB APPLICATION, PYTHON, DJANGO, RELATIVE DATABASE.

The object of research is the process of note management.

The subject of research is note management software.

The purpose of the work is to develop note management software to provide separate and shared access to users.

Materials, methods and technical means: object-oriented programming, Python programming language, Django web framework, MySQL database management system.

Results. Web application has been developed. It enables remote server access to universalize access to notes by providing it through usage of various types of devices based on the usage of a web browser, as well as dividing access level for notes into individual, shared and public.

Conclusions. The developed software enables users to create, view, edit, color, delete and search notes, to create and view tags, to display notes by tag, to display all user's notes, to open external access to notes for all users, to open external access to notes by link, to display all notes using calendar to define date of note creation and execution.

Field of usage – storage of ideas, opinions and information in text form for individual and shared usage.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	7
Вступ.....	8
1 Аналіз предметної області.....	10
1.1 Програмне забезпечення керування нотатками.....	10
1.2 Огляд існуючих програмних засобів керування нотатками.....	12
1.3 Висновки за розділом 1	18
2 Розробка архітектури програми.....	20
2.1 Вимоги до програмного забезпечення	20
2.2 Вибір мови програмування	21
2.3 Проектування архітектури програмного забезпечення.....	24
2.4 Проектування структури бази даних.....	26
2.5 Висновки за розділом 2	30
3 Розробка програми	32
3.1 Опис менеджера URL	32
3.2 Опис реалізованих моделей	34
3.3 Опис реалізованих подань.....	35
3.4 Опис розроблених шаблонів	38
3.5 Висновки за розділом 3	40
4 Експлуатація та тестування програми	41
4.1 Призначення програми	41
4.2 Умови виконання програми	41
4.3 Виконання програми.....	41
Висновки	56
Перелік джерел посилання	58
Додаток А Технічне завдання	60
Додаток Б Текст програми	65
Додаток В Слайди презентації.....	82

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

DTL – Django Template Language;

HTML – HyperText Markup Language;

MVT – Model-View-Template.

ВСТУП

Автоматизація охопила майже всі сфери людського життя. Вона полягає в перетворенні процесів, які людина звикла виконувати вручну, в цифрову форму. Паперова робота є одним з перших процесів, на що направлена автоматизація. Створення нотаток зазвичай відбувалося вручну раніше, відповідно задача перетворення даного процесу в цифрову форму є однією з важливих для автоматизації. Одночасно, незважаючи на появу таких інструментів, багато людей продовжило створювати нотатки на папері, таким чином захаращуючи своє робоче місце, незважаючи на наявність сучасної техніки поруч.

Відповідно важливою є не розробка програмного засобу керування нотатками як такого, а розробка програмного засобу, який буде простим у використанні і зробить даний процес на стільки дієвим, що замість паперу буде використовуватися такий програмний засіб.

Окрім того важливим є питання доступу до нотаток. Доступ може організовуватися як за дозволом користувача, так і навпаки заборонятися всім окрім автора. Окрім того можуть існувати нотатки, які можуть читатися всіма користувачами, які використовують програмне забезпечення. Такі нотатки можуть містити рецепти, цитати, тези за певними темами. Таке рішення має дозволити перетворити програмне забезпечення керування нотатками з програмного засобу для індивідуальної роботи на програмний засіб пошуку структурованої інформації. За такого варіанту автор не просто зберігає дані, але і створює певну структуру, яка може зацікавити інших потенційних користувачів.

Саме ідею різного рівня доступів, що включає індивідуальний доступ, доступ за посиланням та загальний доступ, покладено в концепцію розроблюваного програмного забезпечення.

Об'єкт дослідження – процес керування нотатками.

Предмет дослідження – програмне забезпечення керування нотатками.

В якості мети роботи відповідно було визначено розроблення програмного забезпечення керування нотатками для забезпечення роздільного та спільного доступу користувачів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Програмне забезпечення керування нотатками

Для визначення принципів, які покладаються в основу програмного забезпечення керування нотатками, спочатку потрібно визначити модель взаємодії користувачів з програмою.

Відносно роботи з нотатками існують три типи поведінки користувачів:

- архітектор: таким користувачам подобається планувати, розробляти процеси та рамки, і їм потрібен інструмент для створення нотаток, який дозволяє їм легко структурувати свої ідеї;

- садівник: таким користувачам подобається досліджувати, поєднувати різні думки разом, і їм потрібен інструмент для записів, який дозволяє їм легко розвивати свої ідеї;

- бібліотекар: дані користувачі з задоволенням збирають, складають каталог ресурсів, і їм потрібен інструмент, який дозволяє їм легко знаходити свої ідеї [1].

У даній роботі розглядається принцип роботи бібліотекара, для якого позначення тегами та спосіб фільтрування нотаток – це основний засіб, потрібний для швидкого збереження та доступу до того, що потрібно в даний момент, як то примітки до книги, примітки до зустрічі чи рецепт, а також тих користувачів, для яких такий програмний інструмент не потрібен, щоб зафіксувати, упорядкувати та знаходити свої думки, а потрібне полегшене рішення для тих моментів, коли потрібно зафіксувати коротку ідею, скласти швидкий список справ [1].

До програмного забезпечення керування нотатками висуваються наступні вимоги:

- програмне забезпечення повинно надавати доступ до нотаток і дозволяти взаємодіяти з ними достатньо зручно: не кожна програма для створення нотаток повинна мати функції перетворення мовлення в текст або

підтримувати стилуси, але як додаткова можливість така функціональність може розглядатися, з іншого боку існує багато різних способів використання цифрових нотаток, відповідно те, що одна людина вважає необхідним, може просто захащувати інтерфейс іншим людям;

– програми для створення нотаток повинні бути швидкими та простими у використанні: звичайно, що ручку та клаптик паперу важко перемогти щодо швидкості та зручності, але будь-який застосунок для створення нотаток повинен прагнути до цієї зручності, відповідно створення, редагування та сортування нотаток повинно бути чимось природним, а не битвою з жахливим користувацьким інтерфейсом, тому якщо нотатку не вдалось створити за лічені секунди або потрібно було переходити з однієї сторінки на іншу, щоб використовувати набір різних інструментів, то програма даній вимозі не відповідає;

– принцип мобільності: аналогічно ноутбуку, який повинен бути завжди в наявності, незалежно від того, знаходиться людина за своїм столом або в польоті в повітрі, як мінімум, потрібні програми, доступні на настільному персональному комп'ютері і мобільній платформі та за можливості мати певні офлайн-функції, якщо не вдалося підключити доступ до блокнота, оскільки не було підключення до мережі Інтернет;

– програми повинні мати добре співвідношення ціни та якості: рішення може бути взагалі безкоштовним, але може і потребувати певної сплати коштів, коли важливою є також підтримка протягом найближчих декількох років та наявність серверних функцій, таких як синхронізація, зокрема на даний момент деякі з найкращих застосунків використовують невелику передплату, і якщо підтримка здійснюється на потрібному рівні, то навіть це не є перешкодою для вибору такого програмного засобу достатньо великою кількістю користувачів [2].

1.2 Огляд існуючих програмних засобів керування нотатками

Одним із найстаріших та найбільш повнофункціональних засобів, які використовуються для керування нотатками, є Evernote [3] (рис. 1.1), який дозволяє створювати як прості, так і складні робочі процеси, використовуючи комбінацію блокнотів, нотаток і тегів, щоб тримати все впорядковано, що відзначається наступними характеристиками:

- доступ до нотаток можна отримати на ноутбуках, мобільних пристроях та в Інтернеті, тому рідко виникає ситуація, коли не можна отримати раніше збережене;

- інші функції включають можливість встановлювати нагадування, представляти нотатки в стилі PowerPoint та об'єднувати їх разом;

- останні доповнення включають нові таблиці та інтеграцію Siri для тих, хто використовує версію під операційною системою iOS;

- незважаючи на те, що існує базовий безкоштовний тарифний план із обмеженими можливостями, є також два платні тарифи – план Evernote Premium для індивідуальних користувачів та бізнес-план Evernote Business: план Evernote Premium дозволяє інтегрувати засоби Slack, Outlook, Google Drive, Microsoft Teams та Salesforce, а також забезпечує засоби штучного інтелекту для знаходження подібного вмісту в інших примітках, а також інформації про вміст із медіа-сайтів, бізнес-план Evernote включає вищезазначене, але також має варіанти спільної роботи та адміністративні функції команди [4].

Evernote має величезну кількість розширених функцій, ефективність яких може бути під питанням в різних ситуаціях, проте потребує додаткового аналізу в межах цієї роботи, оскільки деякі з функцій можуть виявитися можливими потенційними функціональними особливостями програмного забезпечення, що розроблюється.

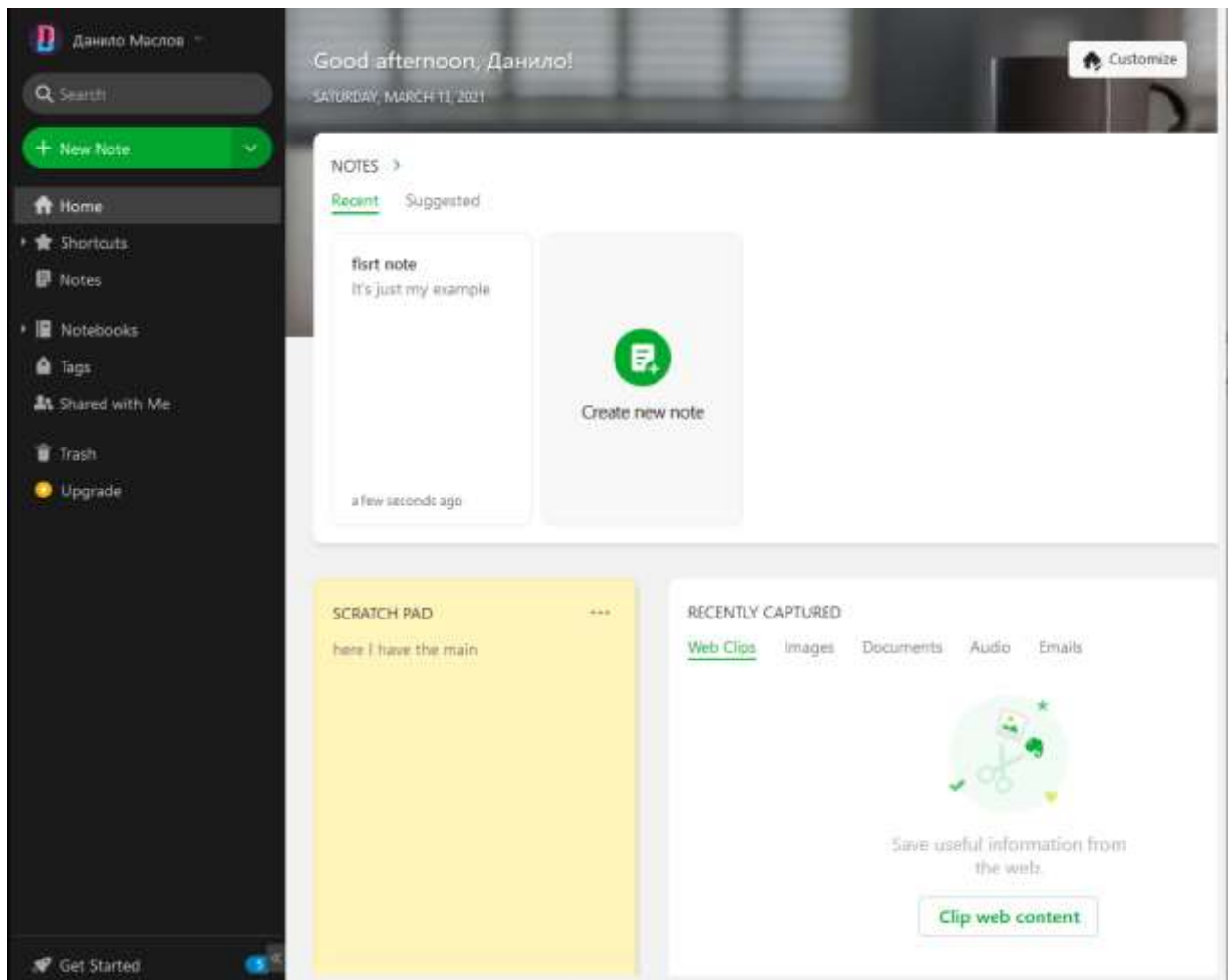


Рисунок 1.1 – Інтерфейс програми Evernote

Загалом Evernote – це місце для записування, збереження, упорядкування та пошуку усіх думок, нотаток, голосових нагадувань, зображень, файлів – все, що може завантажити користувач, все зберігається в хмарі, щоб можна було дістатися до цих даних зі свого комп’ютера, смартфона або планшета, поставляється з достатньо зручним інструментом пошуку, який дозволяє знаходити все, що було завантажено у власний обліковий запис, Evernote пропонує набір інструментів для сортування та пошуку нотаток іншими способами, такими як теги, мітки геолокації, блокноти та стоси блокнотів [5].

Evernote має декілька вдосконалених функцій, яких немає в більшості інших програм для створення нотаток:

– можливість пошуку тексту та пошуку його на зображеннях, незалежно від того, є вони знімками сторінки набраного тексту чи містять рукописний текст: з конкурентів, тільки Microsoft OneNote також може це зробити;

– якщо користувач робить фотографії дошок, презентацій, візитних карток, реклами, рецептів чи чогось іншого, що потрібно зберегти, Evernote зробить весь текст на зображенні доступним для пошуку, хоча звичайно, що дана функція не є на 100 відсотків надійною: час від часу програма щось пропускає;

– пошук має достатньо широкі засоби, адже не потрібно докладати багато зусиль, щоб ретельно впорядкувати свої нотатки, можна шукати текст, незважаючи на те, набраний він чи знаходиться на зображенні: для облікових записів Premium та Business Evernote також здійснює пошук тексту у вкладених PDF та офісних файлах;

– вдосконалена функція Work Chat розміщує вікно обміну миттєвими повідомленнями прямо у робочій області Evernote: воно доступне для всіх членів Evernote, включаючи безкоштовних членів Basic, можна запросити кого-небудь поспілкуватися, поки користувач знаходиться в Evernote;

– вебкліпер – це розширення браузера, яке дозволяє зберігати вебсторінку як нотатку Evernote: наприклад, якщо знайдено статтю, яку потрібно прочитати пізніше, то можна перетворити її на нотатку, і сказати Evernote зберегти її у вибраному блокноті, вебкліпер має можливості пропускати рекламу на сторінці, відсікати лише вибрану частину сторінки тощо, декілька конкурентів Evernote також мають вебкліпер, включаючи OneNote, Simplenote та Google Keep;

– функція під назвою «Конфліктні зміни» дозволяє виявляти випадки, коли якісь зміни в нотатці під час синхронізації не були коректно оброблені, при цьому зберігаються копії нотатки в створеній папці, що називається «Конфліктні зміни», зокрема конфлікти виникають, якщо редагувати нотатку

в автономному режимі без попередньої синхронізації змін із попередньою редакцією на іншому пристрої [5].

Відносно Evernote можна зробити наступні загальні висновки:

- до позитивних рис належить велика кількість функцій, щоправда для оцінки цього набору потрібний певний час;
- до негативних рис можна віднести те, що велика кількість викликає певне нагромадження, через що фактично важко користуватися всім набором наявних засобів, окрім того проблемою є те, що для отримання повного набору можливостей необхідний тарифний план, за який потрібно вносити кошти.

Ще одним засобом, який може використовуватися для досягнення поставлених вимог роботи з нотатками, є сервіс від компанії Google – Google Keep [6].

Застосунок Google Keep надає найбільш простий у використанні інструмент для створення нотаток з мінімалістичним дизайном (рис. 1.2), при цьому підтримуючи дану концепцію як на рівні функціональності, так і на рівні візуалізації:

- в інтерфейсі нотатки наче розкладені перед користувачем на столі, при чому кожна нотатка має певний колір та прикріплена до певної позиції на дошці;
- нотаткам можна присвоїти ярлики, закріпити їх у верхній частині, поєднати з нагадуваннями та спільно використовувати в режимі реального часу;
- надаються функції перетворення промовленого тексту в написаний текст, завдяки чому можна диктувати нотатки в дорозі, а не набирати текст, також є можливість встановити або зняти прапорці під час роботи зі списками [4].

Застосунок Google Keep доступний через вебінтерфейс і таким чином нотатки можуть використовуватись з пристроїв різних типів, на яких встановлено браузер.

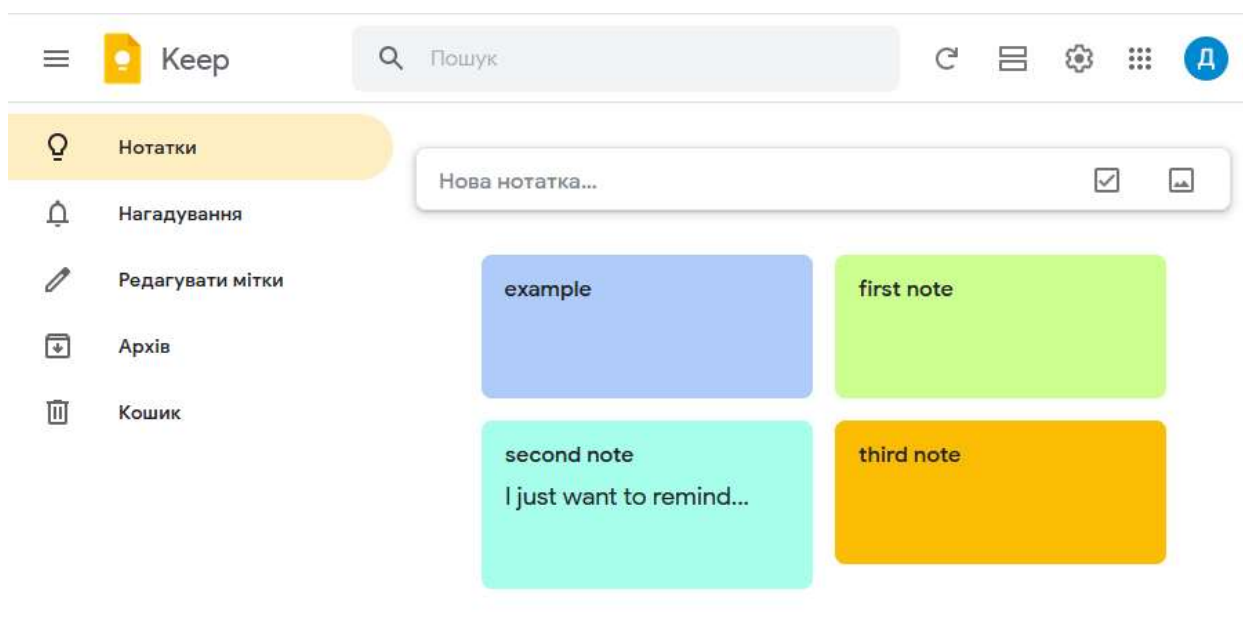


Рисунок 1.2 – Інтерфейс програми Google Keep

Збереженими нотатками можна легко ділитися з іншими користувачами електронною поштою, але при цьому важливо, що застосунок не дозволяє використовувати посилання для спільного використання чи ділитися нотатками лише для читання, відповідно підтримується спільна робота над нотатками, але не їх перегляд, що дещо обмежує простір їх практичного застосування [7].

Google Keep повністю безкоштовний у використанні засіб, але при цьому безкоштовні користувачі отримують 15 Гб безкоштовного сховища, що більше ніж у подібних платформах, таких як OneDrive та iCloud, які пропонують лише по 5 Гб, окрім того надається додаткове сховище, доступне через підписку на Google Workspace, а також широкий спектр інших пільг, зокрема:

- Business Starter (6 доларів на користувача на місяць) постачається із 30 Гб на користувача;
- Business Standard (12 доларів на користувача на місяць) та Business Plus (18 доларів на користувача на місяць) оновлюють обмеження до 2 Тб та 5 Тб відповідно [7].

Таким чином, щодо даного сервісу можна виділити наступні важливі фактори:

- набір позитивних характеристик, що включають вдалий як у сприйнятті, так і у використанні мінімалістичний дизайн, безкоштовність, зручність засобів пошуку та тегування нотаток;

- до недоліків належить система доступу до нотаток ззовні (не автором нотатки), адже не передбачається зовнішнє читання нотаток, а доступ до нотатки може бути організовано тільки через обліковий запис Gmail, що знову ж таки дещо звужує простір використання програмного забезпечення.

Google Keep та Apple Notes набагато ближче за формою та функціями до цифрових нагадувань або наліпок, вони зручні для того, щоб швидко записати ідею і залишити цю нотатку, щоб допомогти її запам'ятати, або додати до неї нагадування типу будильника, у свою чергу Evernote більше схожий на цифрову картотеку, де зберігання та можливість пошуку інформації є настільки ж важливими для досвіду, як і записування [5].

Microsoft 365 є одним з альтернативних засобів, які запропоновані компанією Microsoft для даної предметної області.

Microsoft 365 є хмарним засобом, який надає доступ до Word для початківців, що є найважливішим доступним програмним забезпеченням для документів, і дозволяє використовувати набагато більш гнучкий та керований формат, коли йдеться про створення нотаток, ніж інші основні програми, відповідно не потрібно турбуватися про імпорт своїх нотаток у Word – саме це і потрібно робити великій кількості користувачів, окрім того є важливі засоби синхронізації документів із персональним комп'ютером, а також спільної роботи з колеги, окрім того можна писати, редагувати та надсилати основні документи в дорозі, а не набирати нотатки для подальшого повторного використання чи добування, заощаджуючи час, окрім того надаються засоби на основі Excel для електронних таблиць,

Powerpoint для презентацій або Microsoft Teams для відеоконференцій, які вбудовані в платформу як стандарт [4].

При використанні Microsoft 365 варто враховувати, що це не безкоштовний засіб: використання будь-якого тарифного плану потребує сплати певної суми коштів на рік, при цьому безкоштовну версію можна лише тимчасово випробувати. Це можна вважати певним недоліком даного програмного засобу.

Узагальнюючи отримані результати аналізу, варто відзначити, що Evernote та Microsoft 365 дещо розширюють ідею керування текстовими нотатками до керування загалом нотатками у вигляді будь-яких даних та даних у формі документів офісних застосунків відповідно. Тому єдиним засобом, який концентрується на ідеї роботи в основному з текстовими даними, при цьому є достатньо універсальним щодо платформ, які можуть використовуватися, а до того ж безкоштовним, є Google Keep.

Важливо в розроблюваному програмному забезпеченні розширити ідею надання доступу до нотаток таким чином, щоб певні дані могли ставати не тільки персональними, але і загально доступними. Це дозволить ідею пошуку даних з персонального пошуку на загальний рівень.

1.3 Висновки за розділом 1

Виконано аналіз проблеми керування нотатками:

- розглянуто основні типи поведінки користувачів при роботі з нотатками, виділено бібліотекара в якості базового типу поведінки користувача для розроблюваного програмного забезпечення;

- виділено вимоги до сучасного програмного забезпечення керування нотатками;

- виконано огляд програмних аналогів, що включають зокрема Evernote, Google Keep, Microsoft 365, визначено основні позитивні та негативні риси даного програмного забезпечення;

– визначено основні функціональні особливості розроблюваного програмного забезпечення.

У даній дипломній кваліфікаційній роботі має бути розв’язано такі завдання:

– визначення вимог до програмного забезпечення керування нотатками;

– вибір мови і засобів розробки;

– проєктування програмного забезпечення;

– реалізація програмного забезпечення;

– тестування і налагодження програмного забезпечення.

2 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

2.1 Вимоги до програмного забезпечення

Створення програмного забезпечення має бути реалізовано за трьома базовими принципами роботи з нотатками:

- отримання доступу до власних нотаток;
- отримання доступу до нотаток за наявності посилання;
- виділення нотаток для загального доступу.

Програмне забезпечення, яке розробляється в роботі, має реалізовувати виконання таких функцій для підтримки роботи з електронними нотатками на основі описаних вище принципів:

- авторизація користувача;
- створення нотатки з визначенням назви, тексту та дати виконання;
- перегляд нотатки;
- редагування тексту, назви нотатки та дати виконання;
- розфарбовування нотатки;
- вилучення нотатки;
- пошук нотаток за назвою і текстом;
- визначення тегів нотатки;
- створення тегу;
- відображення нотаток за тегом;
- відображення всіх власних нотаток;
- відкриття зовнішнього доступу до нотатки для всіх користувачів;
- відкриття зовнішнього доступу до нотатки за посиланням;
- отримання доступу до нотатки за посиланням;
- відображення всіх відкритих нотаток;
- пошук серед всіх відкритих нотаток;
- відображення нотаток, на які підписався користувач;
- відображення нотаток за календарем за датою створення;
- відображення нотаток за календарем за датою виконання;

– реєстрація користувача.

Для визначених функціональних вимог було створено діаграму прецедентів [8] (рис. 2.1).

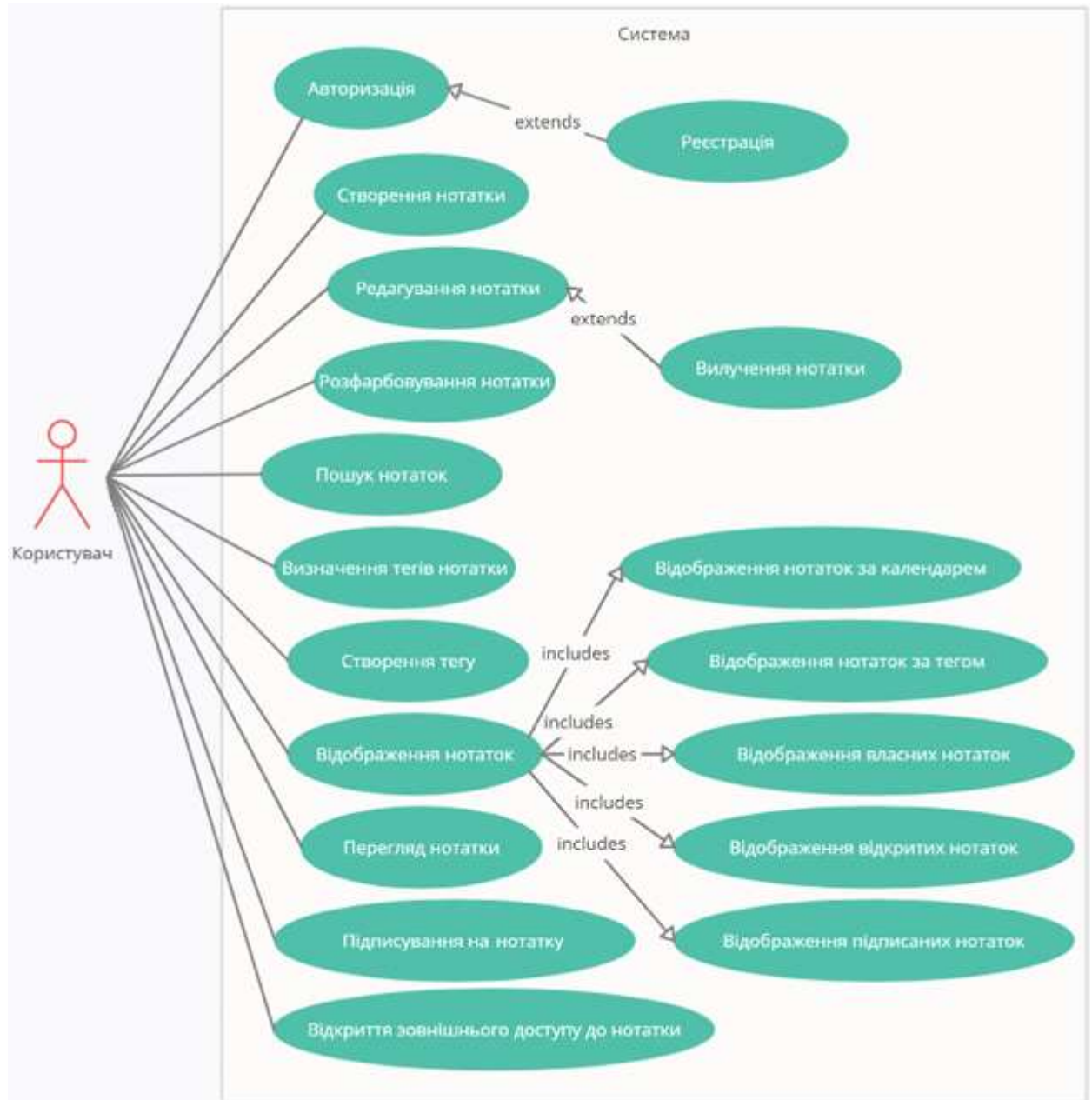


Рисунок 2.1 – Діаграма прецедентів роботи з системою

2.2 Вибір мови програмування

Зважаючи на описані особливості розроблюваного програмного забезпечення, а також те, що для забезпечення доступу до даних з різного

типу пристроїв загалом та з різних пристроїв зокрема необхідно розташовувати програмне забезпечення на сервері, а тому доцільною є розробка вебзастосунку, то в якості альтернатив для розробки було обрано мови програмування Python [9] і C# [10].

Мова Python [11]-[12] відзначається наступними характеристиками:

- є універсальною мовою, яка може використовуватися для різних прикладних областей, включаючи веброзробку, роботу з даними, штучний інтелект;
- має широкі засоби веброзробки, що включають різні вебфреймворки, зокрема Django і Python;
- в основі є об'єктно-орієнтованою мовою програмування, хоча надає також засоби процедурного, структурного, модульного, функціонального програмування;
- має достатньо простий синтаксис, який є як простим у вивченні, так і у використанні, а також подальшому читанні;
- надає безкоштовні засоби і об'єднує навколо себе величезну спільноту, що займається вільною розробкою;
- має величезну кількість бібліотек як у складі стандартної бібліотеки, так і серед розширень, які окрім того є безкоштовними, відповідно вирішення будь-якої задачі в різних предметних областях дещо спрощується за рахунок наявності готових засобів для вирішення підзадач.

У свою чергу мова програмування C# [10] відзначається наступними характеристиками:

- має засоби веброзробки, побудовані на основі окремої платформи розробки вебзастосунків – ASP.NET, надається набір фреймворків веброзробки, одним з яких є фреймворк ASP.NET MVC;
- є мультипарадигмальною мовою програмування, що включає в себе парадигми об'єктно-орієнтованого, процедурного, функціонального, узагальненого, подієвого, компонентно-орієнтованого, рефлексивного програмування;

– зважаючи на приналежність до екосистеми компанії Microsoft, вебсервер загалом має працювати під керуванням операційної системи Windows, хоча за рахунок використання ASP.NET Core ця вимога може бути знижена, при цьому варто також зважати на те, що незважаючи на те, що ASP.NET тепер є вільним відкритим рішенням, використання зокрема середовища Visual Studio для роботи з рішеннями мовою C# за бізнес-використання потребує сплати за підписку, хоча і є безкоштовним за індивідуального використання, що можливо під час роботи над дипломною кваліфікаційною роботою, проте за практичного використання розробленого програмного засобу це стане проблемою.

Отримані результати вибору мови програмування для реалізації програмного забезпечення представлено в табл. 2.1.

Таблиця 2.1 – Вибір мови програмування

Показник	C#	Python
Веброзробка	+	+
Вебфреймворки	+	+
Клас	мультипарадигмальний	об'єктно-орієнтований (з можливістю використання інших парадигм)
Синтаксична простота	—	+
Підтримка	відносно невелика кількість бібліотек	велика кількість вільних бібліотек
Вартість використання	потребує сплати за бізнес-використання	безкоштовно

У результаті для подальшої розробки було обрано саме мову програмування Python, оскільки вона є безкоштовною, має величезну

кількість додаткових бібліотек, простий синтаксис, що позитивно впливає на швидкість процесу розробки.

2.3 Проєктування архітектури програмного забезпечення

Проєктування вебзастосунку було виконано на основі використання вебфреймворку Django.

Послідовність оброблення запиту користувача в межах використання вебфреймворку Django (рис. 2.2) охоплює складові даного вебфреймворку.

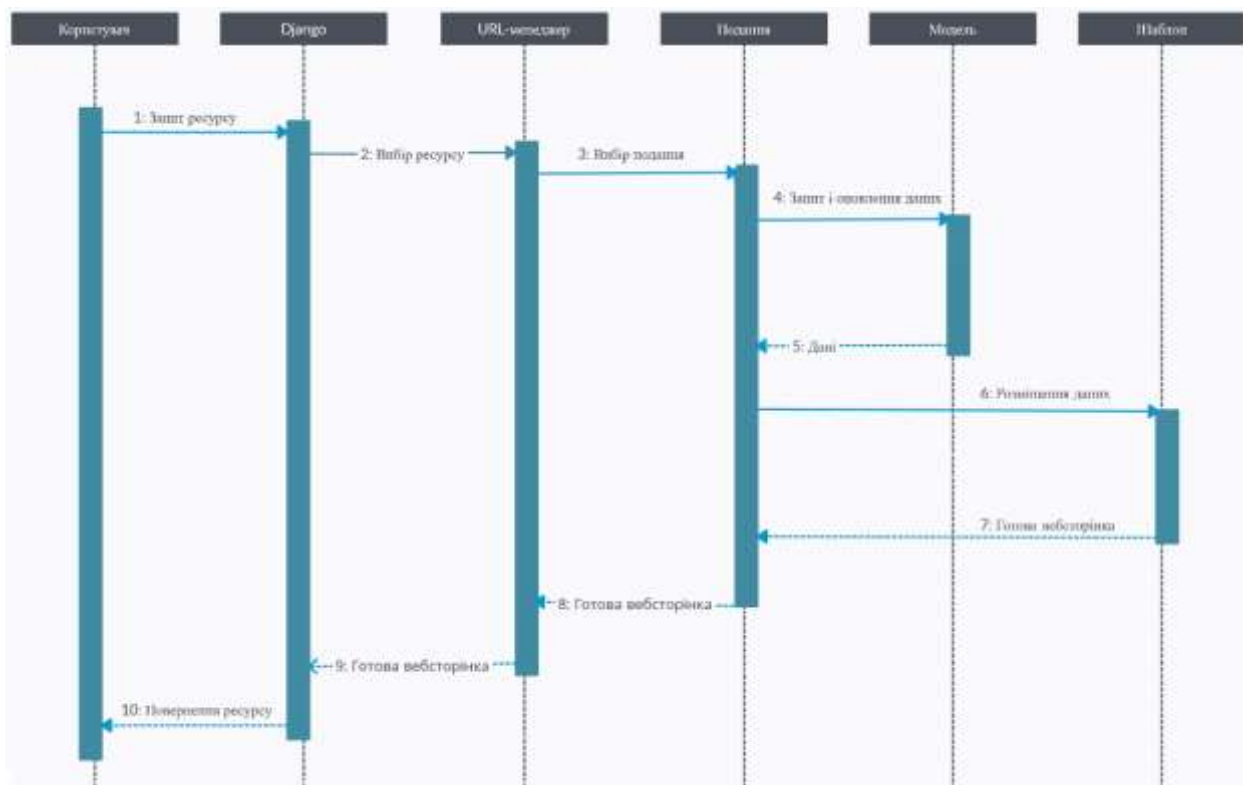


Рисунок 2.2 – Діаграма послідовності при роботі над запитами користувача у вебфреймворку Django

Під вебфреймворком загалом розуміється програмний інструмент, який забезпечує спосіб створення та запуску вебзастосунків, в результаті не потрібно створювати весь код власноруч, при цьому вебфреймворк, покладений в основу Django, визначає базову модель проєктування

програмного забезпечення, що складається з архітектури модель-контролер-подання для розробки вебзастосунків, але дещо відрізняється, оскільки при цьому реалізується концепція Model-View-Template (MVT):

- головна відмінність між MVT та модель-контролер-подання полягає в тому, що Django сам піклується про частину контролера (програмний код, який контролює взаємодію між моделлю та поданням), залишаючи розробникам шаблони – файли HyperText Markup Language (HTML), змішані з мовою шаблонів Django Template Language (DTL);

- модель представляє собою логічну структуру даних, це проміжний програмний рівень та обробник даних між базою даних та поданням, модель надає визначення того, як формати даних надходять з подання, тому вони зберігаються в базі даних і навпаки, тобто отримання інформації з бази даних передається до подання у форматі, що відображається;

- подання отримують дані, а також інформацію про метод запиту (“POST”, “GET”) з боку клієнта, і відповідно форматують дані для моделі, щоб їх можна було зберігати в базі даних, вони також зв'язуються з базою даних для отримання даних, які передаються в шаблон для перегляду;

- Django потребує зручного способу динамічного створення HTML, найпоширеніший підхід спирається на шаблони, що створюються мовою DTL, шаблон містить статичні частини бажаного виводу HTML, а також спеціальний синтаксис, що описує, як буде вставлено динамічний вміст, шаблон забезпечує набагато більшу розширюваність для розробників інтерфейсів, ніж те, що давала архітектура модель-контролер-подання, один шаблон може використовуватися різними поданнями для відображення різних форматів даних, він зберігає весь вміст, який відтворює браузер, ця частина є видимою для клієнта, модель та подання розміщені на стороні сервера [13].

2.4 Проектування структури бази даних

Системою керування базою даних було обрано MySQL [14], що обумовлено наступними її позитивними характеристиками:

- безкоштовна;
- широка функціональність;
- швидкість роботи.

Для реалізації описаної функціональності програмного забезпечення було виділено такі основні сутності:

- Account – обліковий запис користувача;
- Record – нотатка;
- Tag – тег.

Для реалізації функціональності підписування на відкриту для всіх користувачів нотатку необхідно зв'язувати обліковий запис з нотаткою. Відповідно для цього потрібна додаткова таблиця, що такий зв'язок буде організовувати, адже даний зв'язок належить до типу «багато-до-багатьох», тобто потребує введення додаткової таблиці в реляційній базі даних. Для цього було створено таблицю ARAccess.

Аналогічно кожна нотатка може мати набір тегів для її опису, в той же час кожен тег може бути представлений в різних нотатках, тому даний зв'язок належить до типу «багато-до-багатьох», тобто потребує введення додаткової таблиці в реляційній базі даних, що було реалізовано за допомогою таблиці RTInfo.

Таким чином, база даних повинна складатися з 5 таблиць.

Таблиця Account містить такі поля:

- id – номер запису;
- login – логін;
- password – пароль.

Відповідність полів таблиці типам наведена в табл. 2.2.

Таблиця 2.2 – Структура таблиці Account

Номер поля	Назва поля	Тип поля
1	id	INT
2	login	VARCHAR(50)
3	password	VARCHAR(50)

Таблиця Record містить такі поля:

- id – номер нотатки;
- name – назва нотатки;
- recordtext – текст нотатки;
- acessmode – рівень доступу до нотатки;
- acount – кількість підписок на нотатку;
- color – колір;
- creation – дата створення;
- execution – дата виконання;
- modification – дата останньої модифікації;
- author – автор.

Відповідність полів таблиці Record типам даних повністю наведена в табл. 2.3.

Поле author представляє зовнішній ключ з таблиці Account. Таблиця Account пов'язана з таблицею Record зв'язком один-до-багатьох.

Таблиця Tag містить такі поля:

- id – номер тегу;
- title – тег;
- tcount – кількість використань.

Відповідність полів таблиці Tag типам даних наведена в табл. 2.4.

Таблиця 2.3 – Структура таблиці Record

Номер поля	Назва поля	Тип поля
1	id	INT
2	name	VARCHAR(255)
3	recordtext	TEXT
4	acessmode	BOOL
5	acount	BIGINT
6	color	INT
7	creation	DATETIME
8	execution	DATE
9	modification	DATETIME
10	author	INT

Таблиця 2.4 – Структура таблиці Tag

Номер поля	Назва поля	Тип поля
1	id	INT
2	title	VARCHAR(30)
3	tcount	BIGINT

Таблиця ARAccess містить такі поля:

- id – номер запису;
- accountId – обліковий запис;
- recordId – нотатка;
- creation – дата підписки.

Поле accountId представляє зовнішній ключ з таблиці Account. Таблиця Account пов'язана з таблицею ARAccess зв'язком один до багатьох.

Поле recordId представляє зовнішній ключ з таблиці Record. Таблиця Record пов'язана з таблицею ARAccess зв'язком один до багатьох.

Відповідність полів таблиці типам наведена в табл. 2.5.

Таблиця 2.5 – Структура таблиці ARAccess

Номер поля	Назва поля	Тип поля
1	id	INT
2	accountId	INT
3	recordId	INT
4	creation	DATETIME

Таблиця RTInfo містить такі поля:

- id – номер запису;
- tagId – тег;
- recordId – нотатка.

Поле tagId представляє зовнішній ключ з таблиці Tag. Таблиця Tag пов'язана з таблицею RTInfo зв'язком один до багатьох.

Поле recordId представляє зовнішній ключ з таблиці Record. Таблиця Record пов'язана з таблицею RTInfo зв'язком один до багатьох.

Відповідність полів таблиці типам наведена в табл. 2.6.

Таблиця 2.6 – Структура таблиці RTInfo

Номер поля	Назва поля	Тип поля
1	id	INT
2	tagId	INT
3	recordId	INT

За результатами проектування було створено схему бази даних (рис. 2.3).

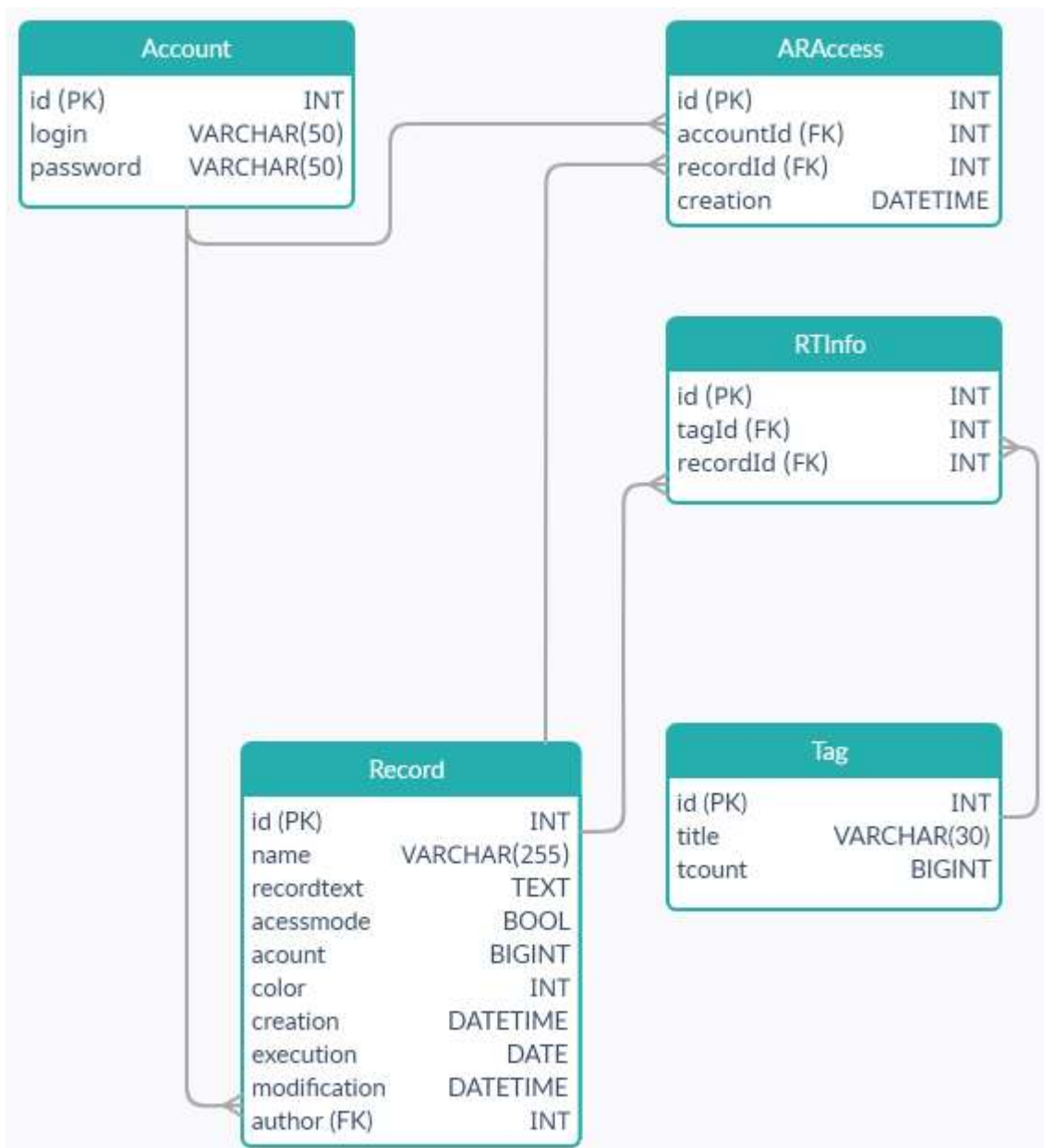


Рисунок 2.3 – Схема бази даних

2.5 Висновки за розділом 2

Виконано проєктування програмного забезпечення на основі отримання наступних результатів:

- визначено вимоги до програмного забезпечення з представленням відповідної діаграми прецедентів;
- виконано вибір мови програмування, в якості якої на основі порівняння обрано мову Python;

- проєктування вебзастосунку було виконано на основі використання вебфреймворку Django;
- проаналізовано архітектурний шаблон MVT та процес взаємодії в межах вебфреймворку Django для формування відповідей на запитання користувача;
- виконано проєктування структури бази даних на основі використання MySQL в якості системи керування базою даних та побудови схеми бази даних;
- визначено структурну схему розроблюваного програмного проєкту на основі застосування шаблону MVT.

3 РОЗРОБКА ПРОГРАМИ

3.1 Опис менеджера URL

Менеджер URL-адрес визначає способи перетворення URL-адрес на подання, яким обробляється задана адреса.

Всі способи розділені на дві частини на основі двох відповідних функцій:

- path, що визначає статичний шлях до відповідного розділу вебзастосунку;
- url, що визначає динамічну адресу, яка формується на основі статичної частини та номеру.

Менеджер URL встановлює зв'язок для наступних подань:

- IndexView – базова відповідність, що встановлена для порожньої адреси, тобто адреси для якої вказано тільки адресу вебсервера, така ж відповідність встановлена для посилання, що включає index;
- SearchView – відповідність для розширеного пошуку нотаток за адресою, що включає search;
- LoginView – відповідність для сторінки авторизації облікового запису для адреси, що включає login;
- LogoutView – відповідність для сторінки виходу з облікового запису для адреси, що включає logout;
- RecordView – відповідність для перегляду деталізованої інформації про нотатку з номером, який включено в адресу, побудовану на основі регулярного виразу через функцію path;
- NewRecordView – відповідність для створення нової нотатки за посиланням, що містить «record/new»;
- EditRecordView – відповідність для редагування нотатки під заданим номером, тому для визначення адреси побудовано регулярний вираз «^record/edit/(?P<pk>\d+)\$»;

– `RecordDeleteView` – відповідність для вилучення нотатки під заданим номером, для визначення адреси побудовано регулярний вираз «`^record/delete/(?P<pk>\d+)$`»;

– `PersonalView` – відповідність для перегляду власних приватних нотаток за адресою, що включає розділ `personal`;

– `SharedView` – відповідність для перегляду нотаток, на які підписався користувач, за адресою, що включає розділ `shared`;

– `SearchResultView` – відповідність для перегляду результатів пошуку нотаток серед приватних і підписаних нотаток за адресою, що включає розділ `sresult`;

– `SearchOpenResultView` – відповідність для перегляду результатів пошуку нотаток серед відкритих нотаток за адресою, що включає розділ `sresult2`;

– `TagView` – відповідність для перегляду переліку тегів нотаток за розділом `tags`;

– `TagRecordsView` – відповідність для перегляду нотаток за тегом за розділом `tagrecords`;

– `TagNumRecordsView` – відповідність для перегляду нотаток за номером тегу, що включено в адресу.

Приклад побудови регулярного виразу для визначення в посиланні номеру нотатки, деталізацію якої потрібно відтворити, даній відповідності встановлено ім'я `read_record`:

```
url(r'^record/(?P<pk>\d+)$', RecordView.as_view(), name='read_record')
```

Окрім того встановлено відповідність для роботи з адміністративною панеллю через розділ `admin`:

```
path('admin/', admin.site.urls)
```

3.2 Опис реалізованих моделей

Для реалізації моделей на основі структури бази даних було визначено наступні моделі.

Модель Record для збереження даних про нотатки. Даний класі як і всі інші класи моделей побудовані шляхом наслідування класу `models.Model`. Даний клас містить зокрема наступні методи:

- `__str__(self)` – повертає назву нотатки;
- `get_absolute_url(self)` – формує абсолютний шлях до нотатки з використанням її ідентифікаційного номеру;
- `shortinfo(self)` – метод, який повертає коротку версію нотатки в двох варіантах (для лівої та правої наліпок, тобто для вужчої та ширшої наліпок), що мають вміщатися в 10 рядків, формуючи текст з тексту нотатки та набору тегів нотатки, для цього застосовується відповідна функція `ShortText` з підпаketу `myrecord.funcs` вебзастосунку;
- `shortinfo2(self)` – метод, який повертає коротку версію відкритої нотатки в двох варіантах, що мають вміщатися в 10 рядків, формуючи текст з тексту нотатки, набору тегів нотатки та логіну автора нотатки, для цього застосовується відповідна функція `ShortText` з підпаketу `myrecord.funcs` вебзастосунку.

Окрім того даний клас включає поле `tags`, яке сформоване шляхом встановлення зв'язку багато до багатьох з моделлю `Tag` вебзастосунку наступним чином:

```
tags = models.ManyToManyField(Tag)
```

Модель `Tag` створює модель тегу нотатки.

Модель `ARAccess` створює модель підписування користувачем на нотатку. Дану модель було реалізовано окремо, а не на основі зв'язку багато

до багатьох, адже відповідна таблиця передбачає збереження дати і часу виконання операції.

3.3 Опис реалізованих подань

Для підготовки даних для роботи з шаблонами вебзастосунку було створено наступні подання:

- IndexView – подання перегляду відкритих нотаток (створене шляхом множинного наслідування класу ListView);
- PersonalView – подання перегляду власних приватних нотаток (створене шляхом множинного наслідування класів LoginRequiredMixin та ListView);
- SharedView – подання перегляду нотаток, на які підписався користувач (створене шляхом множинного наслідування класів LoginRequiredMixin та ListView);
- LoginView – подання авторизації;
- LogoutView – подання виходу з облікового запису;
- RecordView – подання перегляду деталізованої інформації про нотатку (створене шляхом множинного наслідування класів LoginRequiredMixin та DetailView);
- NewRecordView – подання створення нової нотатки (створене шляхом множинного наслідування класів LoginRequiredMixin та CreateView);
- EditRecordView – подання редагування нотатки (створене шляхом множинного наслідування класів LoginRequiredMixin та UpdateView);
- RecordDeleteView – подання підтвердження вилучення нотатки (створене шляхом множинного наслідування класів LoginRequiredMixin та DeleteView);
- SearchView – подання пошуку нотаток на основі визначення дат створення та виконання нотаток;

- `SearchResultView` – подання перегляду результатів пошуку приватних або підписаних нотаток (створене шляхом множинного наслідування класів `LoginRequiredMixin` та `ListView`);
- `SearchOpenResultView` – подання результатів пошуку відкритих нотаток (створене шляхом наслідування класу `ListView`);
- `TagView` – подання перегляду переліку тегів нотаток (створене шляхом наслідування класу `ListView`);
- `TagRecordsView` – подання перегляду нотаток за тегом (створене шляхом множинного наслідування класів `LoginRequiredMixin` та `ListView`);
- `TagNumRecordsView` – подання нотаток за номером тегу (створене шляхом множинного наслідування класів `LoginRequiredMixin` та `ListView`).

Для створення одного з ключових подань, `PersonalView`, для перегляду власних приватних нотаток було побудовано однойменний клас, створений шляхом наслідування класів `LoginRequiredMixin`, оскільки такий перегляд потребує початкової авторизації користувача, та `ListView`, який є базовим класом для створення переліку перегляду даних моделі. Для даного подання вказано в якості його основної моделі модель `Record`, а в якості шаблону, який буде використовуватися для відображення даних переліку нотаток, – `personal.html`.

Весь набір створених класів подань представлено на рис. 3.1.

Оскільки перегляд переліку власних приватних нотаток передбачає виведення не всіх нотаток, а тільки тих, що відповідають описаному критерію, то було реалізовано окремий метод `get_queryset(self, *args, **kwargs)`, який забезпечує формування цього переліку нотаток. Для цього спочатку відбувається фільтрування нотаток за автором, який має відповідати авторизованому обліковому запису, далі перевіряється рівень доступу до нотатки (1 визначає приватний доступ), а далі реалізується сортування нотаток за ідентифікаційним номером, але в протилежному порядку.

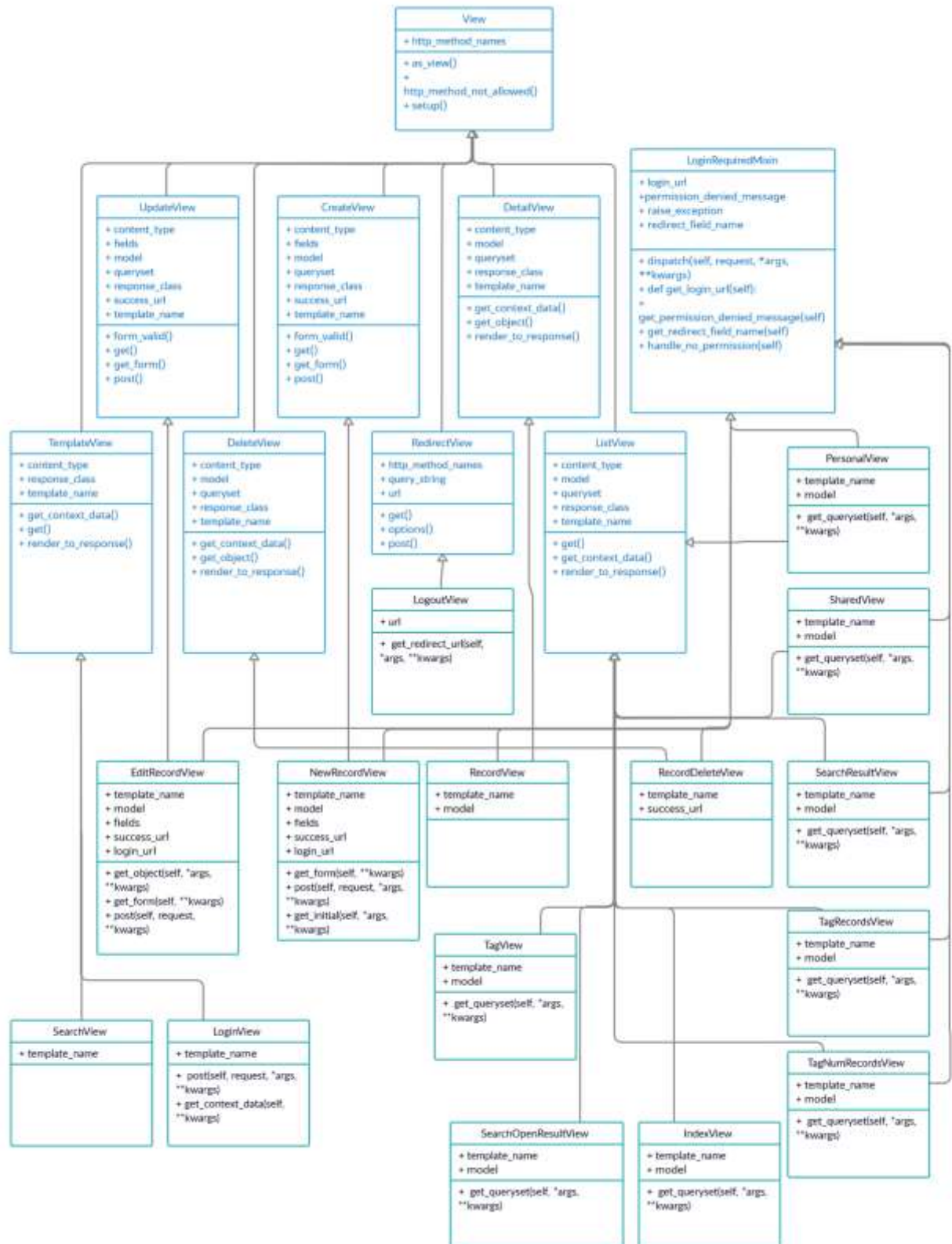


Рисунок 3.1 – Діаграма розроблених класів подань

В кодї виконання цих дій реалізується наступним чином:

```
def get_queryset(self, *args, **kwargs):
```

```

rl = super(PersonalView, self).get_queryset(*args, **kwargs)
rl = rl.filter(author = self.request.user)
rl = rl.filter(acesmode = 1)
rl = rl.order_by("-id")
return rl

```

3.4 Опис розроблених шаблонів

Для створення дизайну вебзастосунку було використано вільний безкоштовний шаблон [15].

У процесі було розроблено набір шаблонів для підтримки виведення результатів роботи кожного окремого описаного вище подання.

Зокрема шаблон `read_record.html` використовується для перегляду повної інформації обраної нотатки. Дані нотатки, які було передано з подання, представляються через об'єкт `record`, шляхом звернення до полів якого визначаються значення відповідних характеристик, значення підставляються динамічно у відповідні позиції HTML-документа, наприклад, в даному випадку підставляються назва та текст нотатки в якості заголовку та вмісту:

```

<header class="major">
    <h1>>{{ record.name }}</h1>
</header>
<div class="content">
    <p>{{ record.recordtext }}</p>
</div>

```

Шаблон `index.html` може використовуватися для відображення відкритих нотаток як для авторизованих користувачів, так і для користувачів

без авторизації. Відповідно в залежності від такого типу взаємодії визначається, які пункти меню відображати, а які ні.

Відображати чи ні власні та підписані нотатки:

```
{% if user.is_authenticated %}
<li><a href="{% url 'person' %}">Власні</a></li>
<li><a href="{% url 'shared' %}">Підписані</a></li>
{% endif %}
```

Відображати можливість виходу з облікового запису чи авторизації:

```
{% if user.is_authenticated %}
<li><a href="{% url 'logout' %}" class="button primary
fit">Вийти</a></li>
{% else %}
<li><a href="{% url 'login' %}" class="button primary
fit">Авторизуватись</a></li>
{% endif %}
```

Для відображення нотатки у відповідній позиції у вигляді стікера використовується циклічний перебір даних (подається фрагмент коду у спрощеному вигляді):

```
{% for rec in record_list %}
<article class="{{ 'art'|add: rec.color }}">
  <header class="major">
    <h3><a href="{{ rec.get_absolute_url }}" class="link">{{
rec.name }}</a></h3>
    <p> {% if forloop.counter|divisibleby:2 %} rec.shortinfo2[0]
{% else %} rec.shortinfo2[1] {% endif %}</p>
```

```
</header>  
</article>  
{% endfor %}
```

Окрім того в даному фрагменті виконується вибір скороченого подання тексту нотатки для відображення в стікері в залежності від того, це парна нотатка або непарна.

3.5 Висновки за розділом 3

Реалізовано програмне забезпечення керування нотатками з роздільним доступом. Описано основні реалізовані моделі, подання та шаблони вебзастосунку. Описано процес їх створення з основними рішеннями щодо їх реалізації. Описано створений менеджер URL.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Призначення програми

Програмне забезпечення призначене для виконання роботи з персональними нотатками, перегляду нотаток інших користувачів за посиланням, а також перегляду відкритих для всіх користувачів нотаток.

4.2 Умови виконання програми

Мінімальні системні вимоги до вебсервера включають:

- процесор: 4 ядра, тактова частота – 2,9 ГГц;
- оперативна пам'ять – 10 Гб;
- платформа – 32-розрядна або 64-розрядна;
- жорсткий диск – 1 Гб вільного простору.

4.3 Виконання програми

Для початку роботи з вебсервером потрібно звернутися до нього через веббраузер, після чого відкриється сторінка для перегляду доступних нотаток неавторизованим користувачем (рис. 4.1).

На даній сторінці відображаються нотатки у вигляді стікерів різних кольорів, які обрано для них їх авторами. Для кожної нотатки відображається її назва, автор, скорочений варіант тексту та теги. Текст скорочується таким чином, щоб інформація про автора, сам текст та набір тегів займали в сумі 10 рядків.

Звідси можна також виконувати пошук, це буде пошук за відкритими нотатками, а також переходити на нотатки за обраним тегом, натискаючи на посилання на відповідній нотатці.

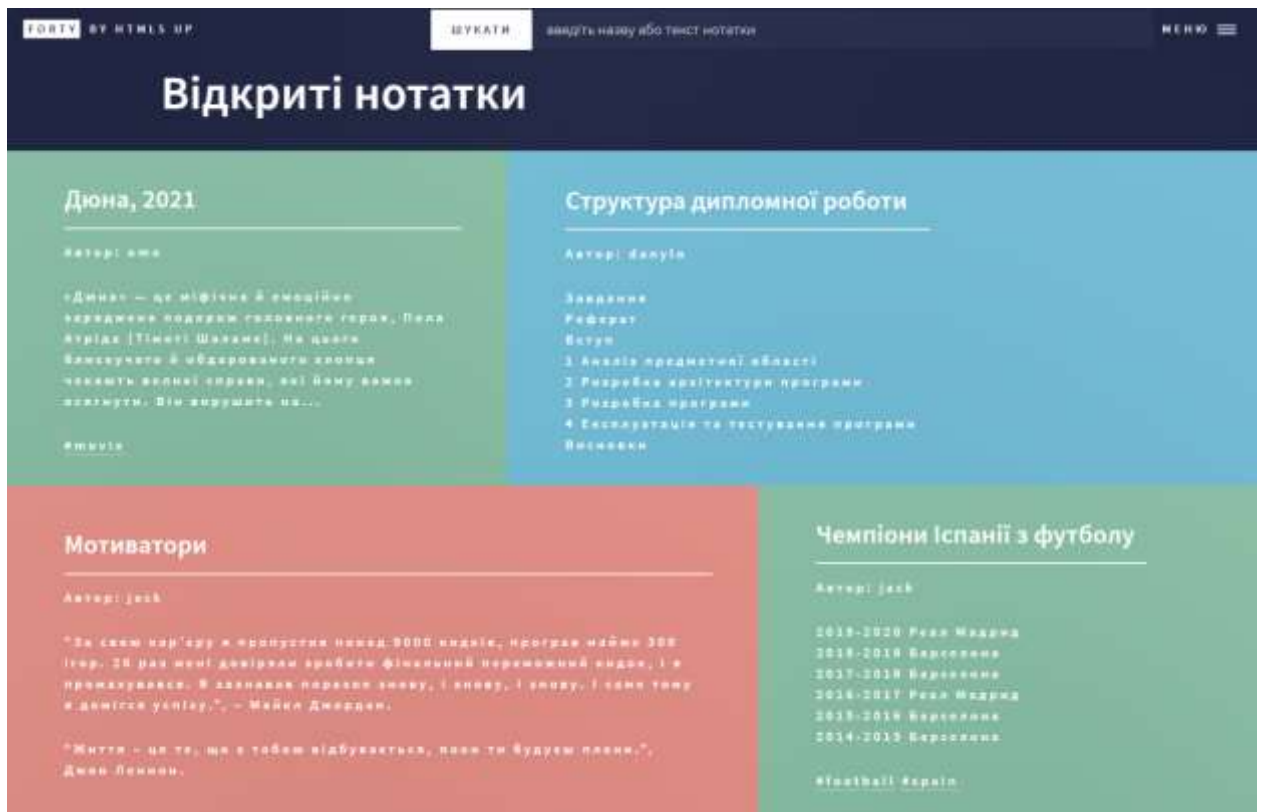


Рисунок 4.1 – Перегляд відкритих нотаток

Після виконання авторизації базовою сторінкою буде сторінка роботи з власними нотатками (рис. 4.2). Перейти до неї за потреби можна за допомогою пункту меню «Власні».

Сторінка переліку власних нотаток складається з 3 частин:

- верхньої шапки;
- середньої частини, яка дозволяє перейти до створення власних нотаток;
- основної частини, що безпосередньо містить перелік власних нотаток.

У верхній шапці розташовується посилання на шаблон, який був використаний для створення стилів вебзастосунку, поле пошуку, яке дозволяє виконувати пошук у власних нотатках (або відкритих, якщо відкрито сторінку з відкритими нотатками) за допомогою визначення назви, слів тексту нотатки або тегів. Окрім того в правому куті розташовано меню вебзастосунку.

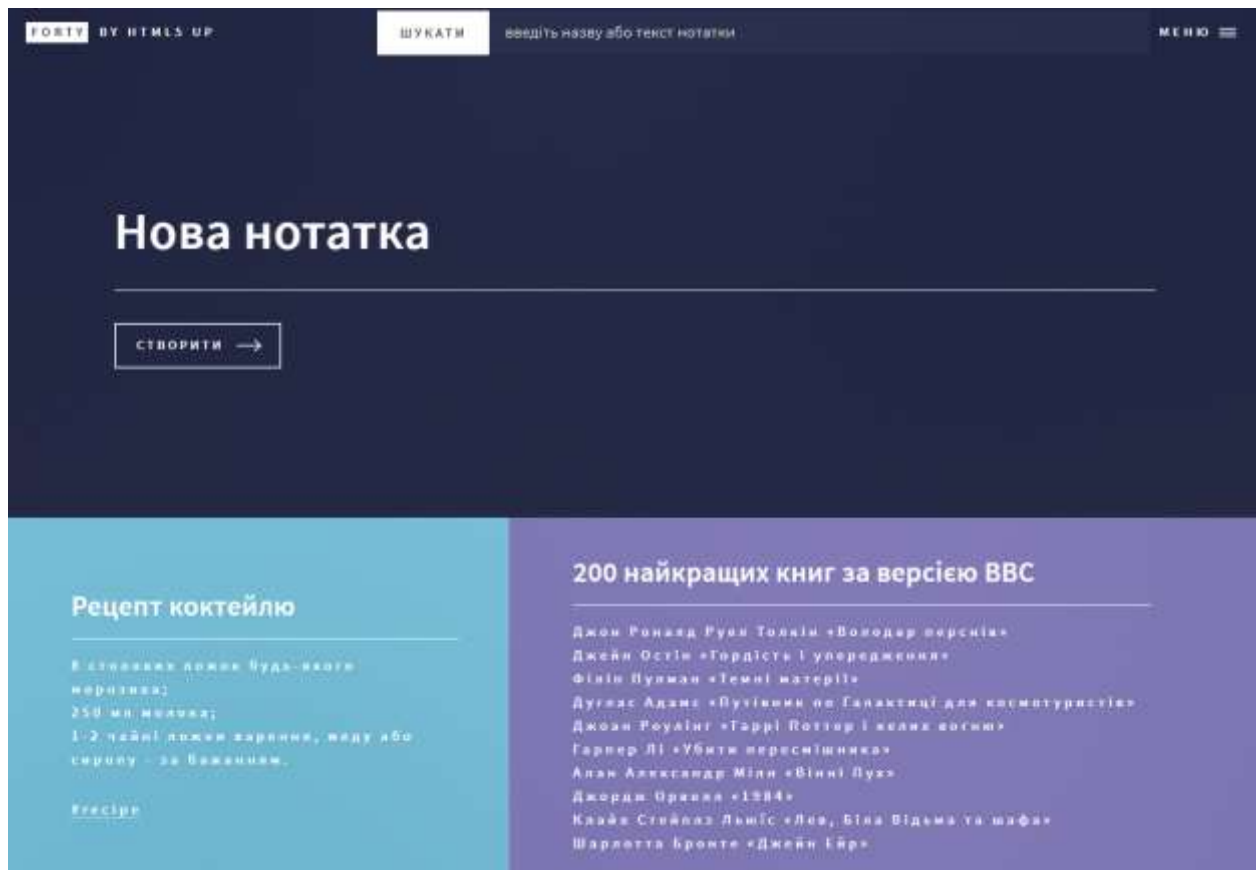


Рисунок 4.2 – Інтерфейс перегляду власних нотаток

Основна частина в розширеному варіанті (рис. 4.3) складається з нотаток, розташованих порядково, при цьому в кожному рядку розташовується 2 нотатки.

Кожна нотатка має власну назву, розташовану до підкреслення, текст нотатки, розташований під підкресленням, та за наявності теги. Теги відділяються окремим рядком знизу, кожне з них має посилання, за натискання на яке можна відкрити перелік власних нотаток, які відповідають даному тегу. Зі всього тексту нотатки розташовуються тільки 10 рядків максимально (за відсутності тегів), усі інші рядки можна переглянути тільки за відкриття самої нотатки.

Кожна нотатка розташовується на відповідному полі, що має колір, заданий користувачем під час створення або редагування її.



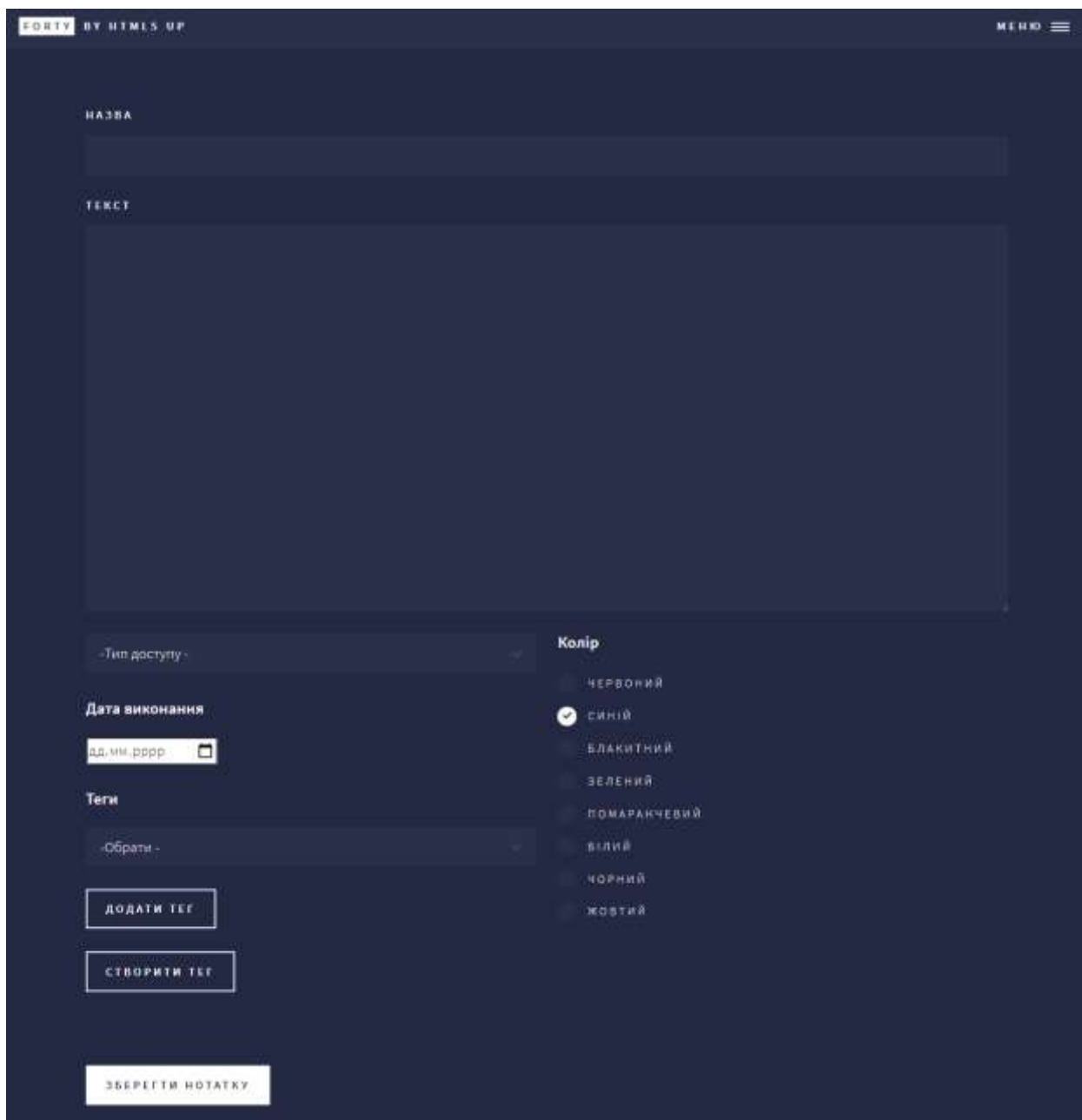
Рисунок 4.3 – Перегляд стікерів з власними нотатками

Для переходу до створення нової власної нотатки потрібно натиснути на кнопку «Створити» під заголовком «Нова нотатка». Після цього відкриється окрема сторінка керування даним процесом (рис. 4.4).

Сторінка створення нової нотатки дозволяє:

- ввести назву нотатки;
- ввести текст нотатки (дане поле може за потреби розширюватися, для чого достатньо потягти за правий нижній кут);
- визначити з випадуючого переліку тип доступу: індивідуальний, за посиланням, відкритий;
- обрати дату виконання з випадуючого календаря;
- пов'язати теги з нотаткою;
- обрати колір нотатки: для цього потрібно обрати один з кольорів за допомогою прапорця (одночасно вибирається тільки одне значення, працює як радіокнопка).

Підтвердити створення нотатки потрібно за допомогою кнопки «Зберегти нотатку».



FORTY BY HTML5 UP МЕНЮ

НАЗВА

ТЕКСТ

Дата виконання

dd.mm.yyyy

Теги

Обрати

ДОДАТИ ТЕГ

СТВОРИТИ ТЕГ

Колір

- ☐ ЧЕРВОНИЙ
- ☒ СИНІЙ
- ☐ БЛАКИТНИЙ
- ☐ ЗЕЛЕНИЙ
- ☐ ПОМАРАНКЧЕВИЙ
- ☐ БІЛИЙ
- ☐ ЧОРНИЙ
- ☐ ЖОВТИЙ

ЗБЕРЕГТИ НОТАТКУ

Рисунок 4.4 – Створення нової нотатки

Обране з календаря значення відображається в текстовому полі, а також його можна задати безпосередньо в текстовому полі (рис. 4.5).

Для того щоб додати тег до нотатки потрібно з випадаючого переліку обрати один з раніше створених (самим користувачем) тегів, після чого натиснути на кнопку «Додати тег». Результат буде відображено вище шляхом додавання відповідного тегу.

Якщо з випадającego переліку обрано тег, який вже раніше було додано до нотатки, то кнопка буде відображена як «Вилучити тег», відповідно натискаючи на неї, можна вилучити раніше доданий тег.

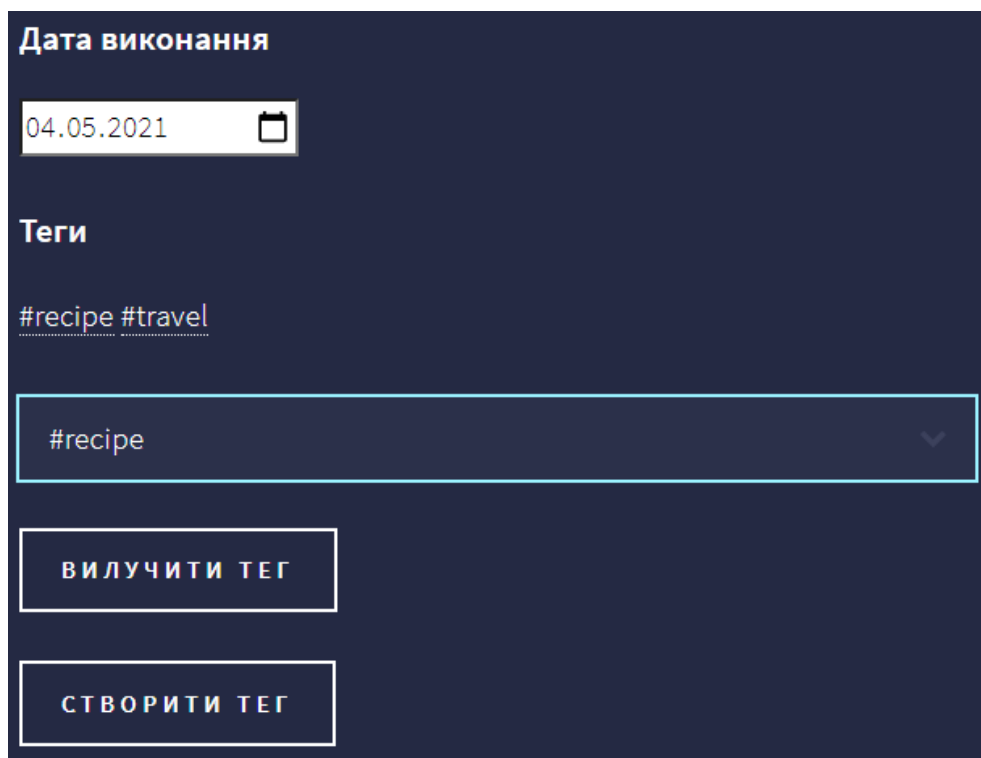
The image shows a dark-themed user interface for managing tags. At the top, there is a section titled "Дата виконання" (Execution Date) with a date input field showing "04.05.2021" and a calendar icon. Below this is a section titled "Теги" (Tags). It displays two existing tags: "#recipe" and "#travel". Underneath, there is a search or selection bar containing the text "#recipe" and a downward arrow. At the bottom, there are two buttons: "ВИЛУЧИТИ ТЕГ" (Remove Tag) and "СТВОРИТИ ТЕГ" (Create Tag).

Рисунок 4.5 – Додавання тегів для нової нотатки

Для того щоб створити новий тег, потрібно натиснути на кнопку «Створити тег».

Якщо з переліку власних нотаток обрати будь-яку, натиснувши на неї безпосередньо (будь-яка зона окрім посилання на тег), то відкриється окрема сторінка перегляду повної інформації про нотатку (рис. 4.6). Дана сторінка складається з назви нотатки, її тексту, розташованих на фоні заданого користувачем кольору.

В окремій зоні внизу (завжди темносинього кольору) відображається також набір тегів, дата створення та дата виконання нотатки (якщо вона була для нотатки встановлена автором).

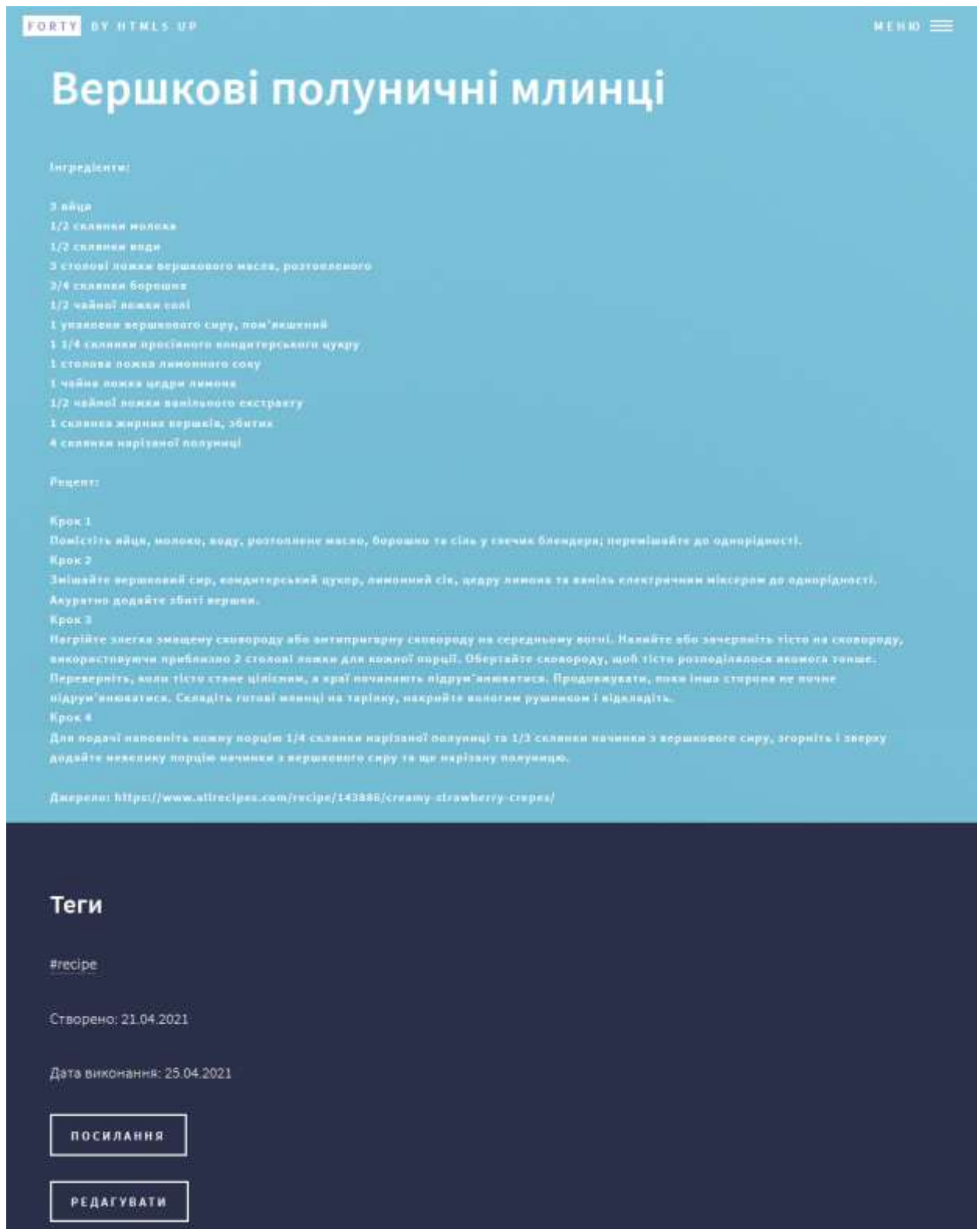


Рисунок 4.6 – Інтерфейс перегляду власної нотатки

Окрім того доступні дві кнопки для керування нотаткою:

– кнопка «Посилання» дозволяє скопіювати до буферу обміну посилання на нотатку, щоб його можна було відправити в листі або

повідомленні іншому користувачу для групового доступу (якщо було визначено приватний рівень доступу до нотатки або відкритий, то кнопка не відображається);

– кнопка «Редагувати», яка дозволяє перейти до відображення сторінки нотатки в режимі редагування, що є аналогічним сторінці створення нотатки, але при цьому відповідні поля заповнені збереженими значеннями і їх можна за потреби змінити.

У випадку перегляду відкритої нотатки (зокрема і неавторизованим користувачем) кнопки не будуть відображатися, а доступ за назвою тегу буде відбуватися для відкритих нотаток, позначених цим тегом.

Найбільш повний варіант меню (рис. 4.7) включає наступний перелік пунктів і відображається для авторизованого користувача в момент роботи, наприклад, над створенням нової нотатки:

– «Власні» – пункт, що дозволяє відобразити набір власних нотаток і надає доступ до засобів контролю власних нотаток, зокрема і створення нотаток;

– «Підписані» – пункт, що дозволяє отримати доступ до переліку нотаток, на які раніше підписався авторизований користувач;

– «Доступні» – пункт, що дозволяє переглядати всі доступні всім користувачам нотатки, тобто відкриті для всіх (є базовим засобом роботи з нотатками для неавторизованих користувачів);

– «Пошук» – пункт, що надає доступ до розширених засобів пошуку нотаток, при цьому він доступний тільки авторизованим користувачам, дозволяє виконувати пошук за датою створення та датою виконання нотатки, але пошук реалізується тільки серед власних нотаток;

– «Теги» – пункт, що надає доступ до переліку тегів і дозволяє працювати над його формуванням, а також швидко отримувати доступ до нотаток за тегом;

– «Вийти» – пункт меню, що дозволяє вийти з авторизованого облікового запису.

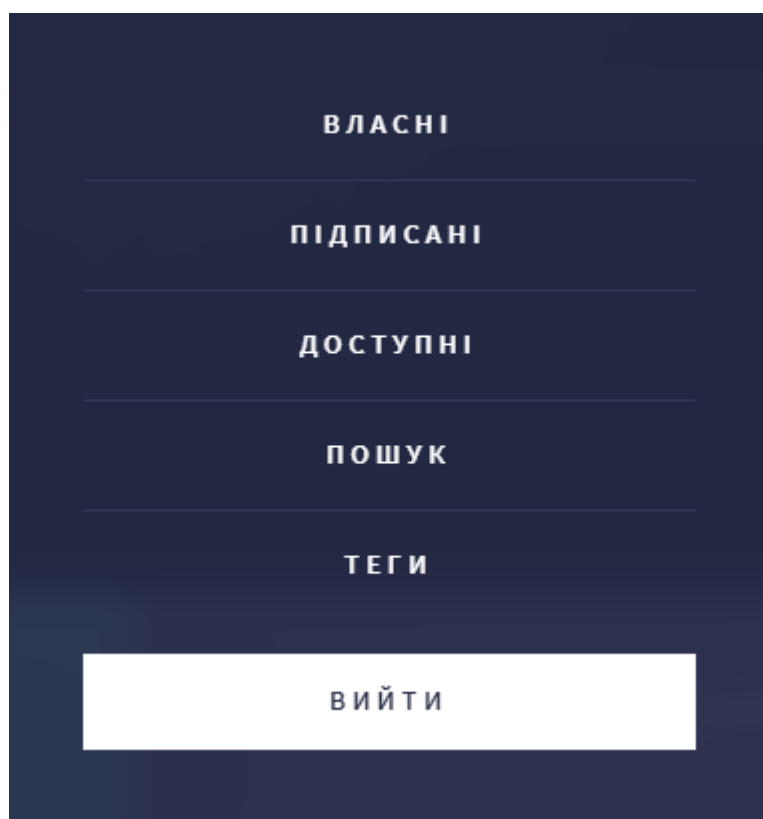


Рисунок 4.7 – Меню

Після вибору пункту «Теги» відкривається сторінка роботи з тегами (рис. 4.8). Дана сторінка відкривається також при натисканні на кнопку «Створити тег» зі сторінки роботи з нотаткою.

На даній сторінці відображається перелік тегів, які були створені користувачем для власних нотаток (до яких прив'язані нотатки). На даній сторінці не відображаються теги, які були створені іншими користувачами.

Перед кожним тегом відображається кнопка, при натисканні на яку даний тег буде вилучено, тобто зв'язок даного тегу зі всіма нотатками даного користувача будуть вилучені.

Окрім того з даної сторінки можна створити власний тег. Для цього потрібно ввести назву тегу в текстове поле, а далі натиснути на кнопку «Додати». Тоді вище буде відображено відповідний тег з можливістю вилучення.

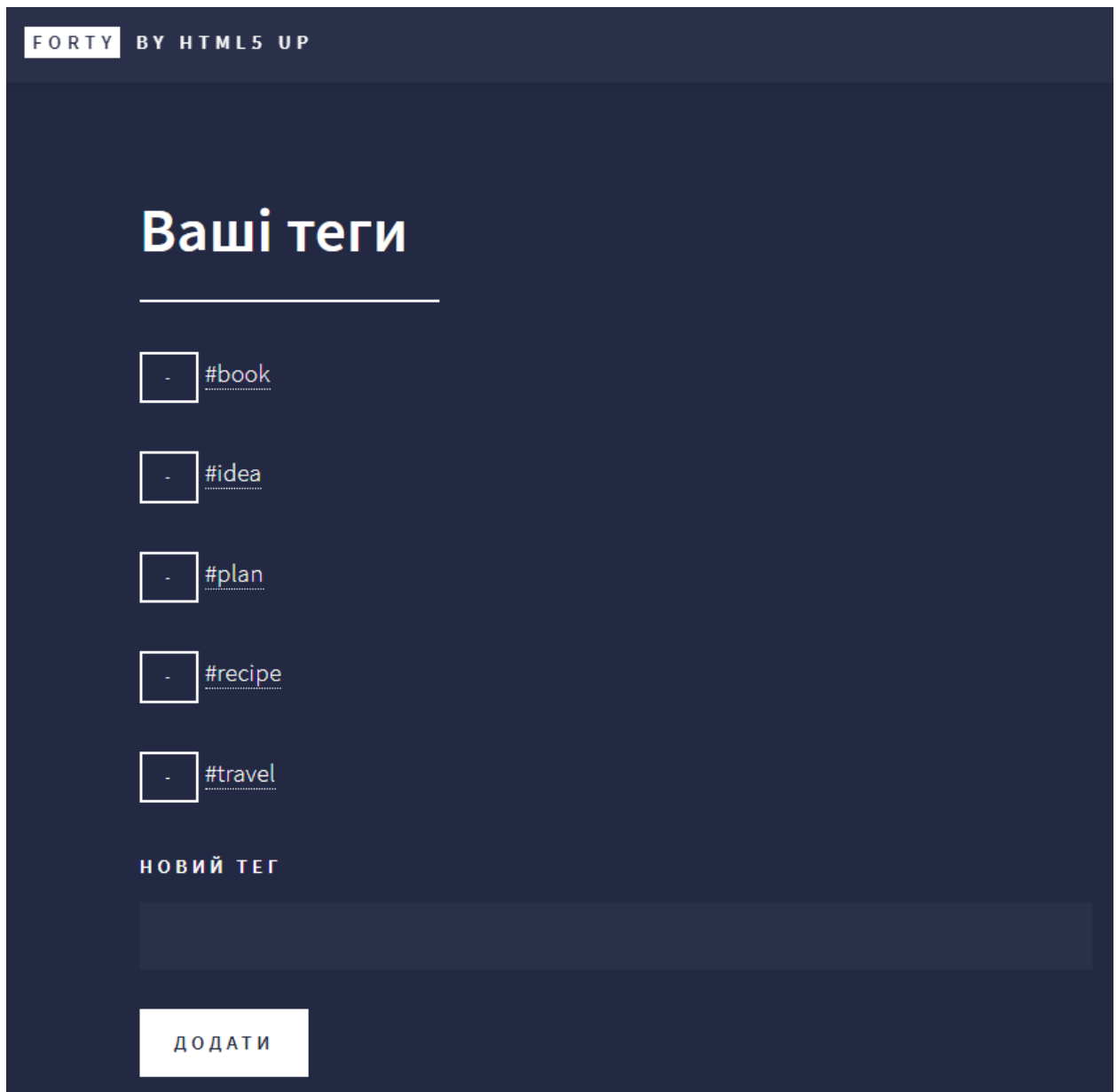


Рисунок 4.8 – Сторінка роботи з тегами

При натисканні на посилання, позначене тегом, відбудеться перехід до власних нотаток користувача, позначених даним тегом (рис. 4.9). Окрім того на даній сторінці відображаються також поширені нотатки, тобто нотатки, посилання на які було введено авторизованим користувачем. Поширені нотатки відображаються в кінці. Поширені нотатки відзначаються тим, що в їх назві спочатку вказується логін автора, а далі через двокрапку – сама назва нотатки.

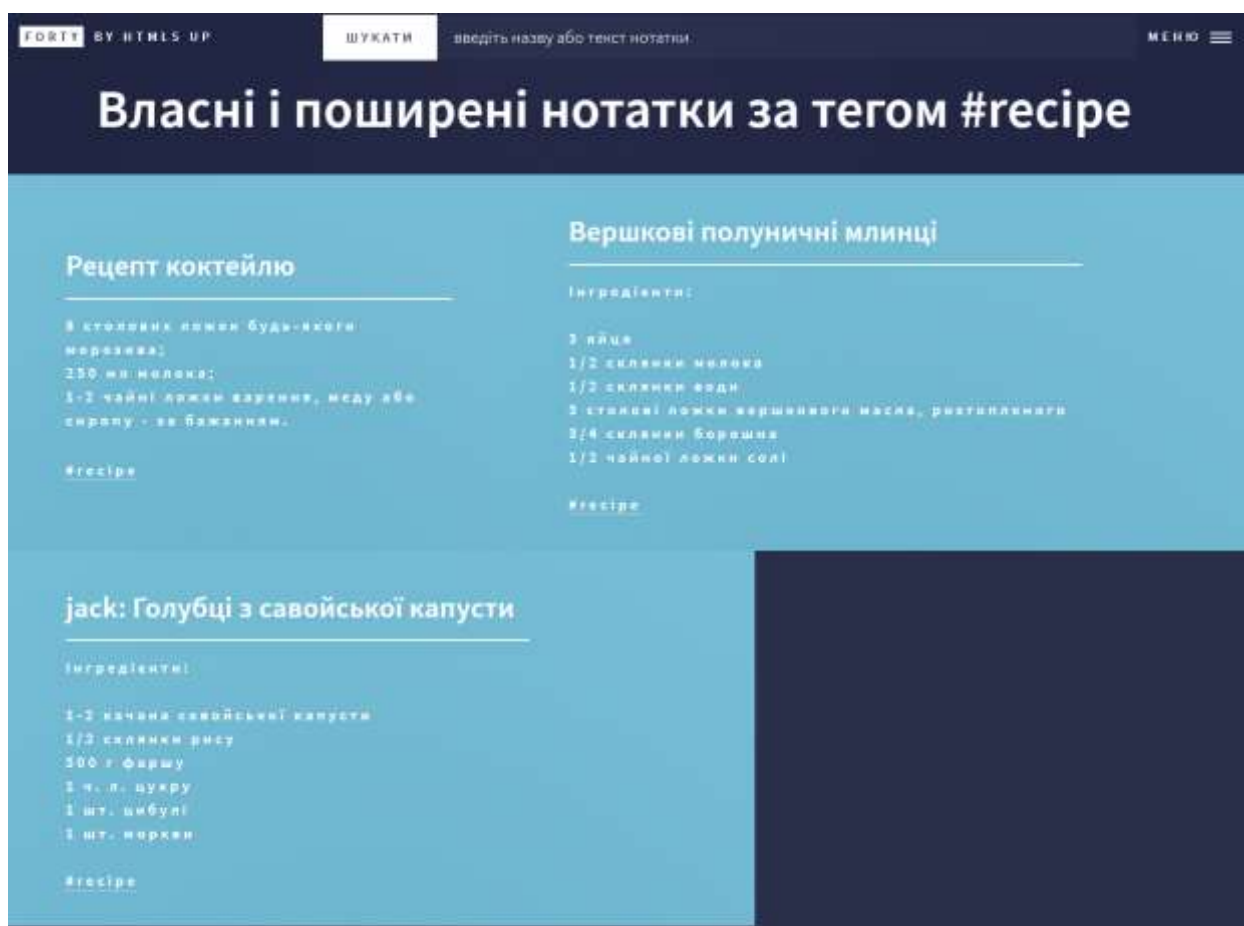


Рисунок 4.9 – Сторінка перегляду нотаток за тегом

У випадку, якщо на сторінці перегляду власної нотатки було натиснено на кнопку «Редагувати», то відкривається окрема сторінка, на якій можна змінити параметри нотатки, зокрема її назву, текст, рівень доступу, дату виконання, змінити набір тегів, а також колір відображення нотатки (рис. 4.10).

Для підтвердження внесених змін потрібно натиснути на кнопку «Зберегти нотатку».

Окрім того з цієї ж сторінки можна не тільки відредагувати нотатку, але і вилучити її. Для цього потрібно внизу сторінки натиснути на кнопку «Вилучити нотатку». Після цього потрібно підтвердити виконання цієї операції, і нотатку буде вилучено з бази даних, разом з усіма її зв'язками, тобто дозволами на перегляд нотатки користувачем.

Рисунок 4.10 – Сторінка редагування нотатки

При виборі з меню пункту «Підписані» відкривається окрема сторінка (рис. 4.11), на якій відображаються всі нотатки, на які був підписаний авторизований користувач. Нотатки відображаються в порядку підпису. Для

кожної нотатки в назві вказується її автор, після чого через двокрапку – сама назва.

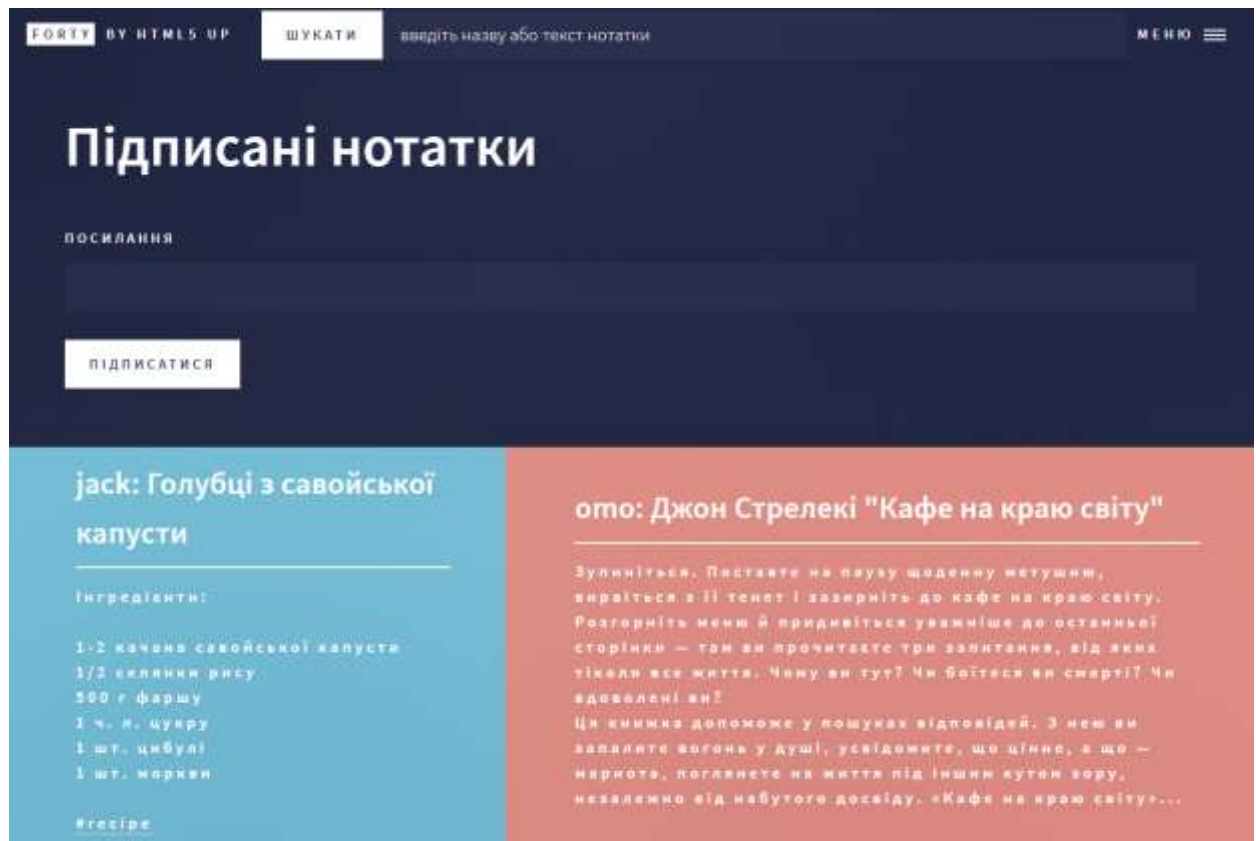


Рисунок 4.11 – Перегляд нотаток, на які підписався користувач

Для того щоб підписатися на нову нотатку, потрібно ввести в текстовому полі адресу нотатки (посилання), яку було сформовано на сторінці перегляду власної нотатки і передано автором. Після цього необхідно підтвердити підписку натисканням на кнопку «Підписатися».

У випадку вибору пункту меню «Пошук» відкривається сторінка розширеного пошуку (рис. 4.12), де можна також в верхньому полі задати частину назви або тексту нотатки, а також теги, а в центральній частині сторінки відображається форма вибору інших параметрів пошуку, включаючи вибір дати з можливістю вибору за допомогою прапорців, чи це визначається дата виконання, чи дата створення нотатки, та включаючи вибір типу нотаток, за яким виконується пошук (власні, підписані чи відкриті).

Рисунок 4.12 – Сторінка розширеного пошуку

Для виконання пошуку потрібно ввести потрібні параметри і натиснути на кнопку «Шукати» або «Знайти», тоді сторінка оновиться, а нижче форми визначення параметрів пошуку буде відображено набір знайдених нотаток (рис. 4.13).

Рисунок 4.13 – Результат пошуку нотаток за назвою

Окрім того за результатами пошуку нижче заголовка сторінки відображається запит, пошук за яким було виконано.

Якщо було визначено дату виконання або створення нотатки, то будуть знайдені нотатки тільки за цією датою (рис. 4.14). Якщо дане поле було залишено порожнім, то буде відображено нотатки без обмеження за даним параметром.

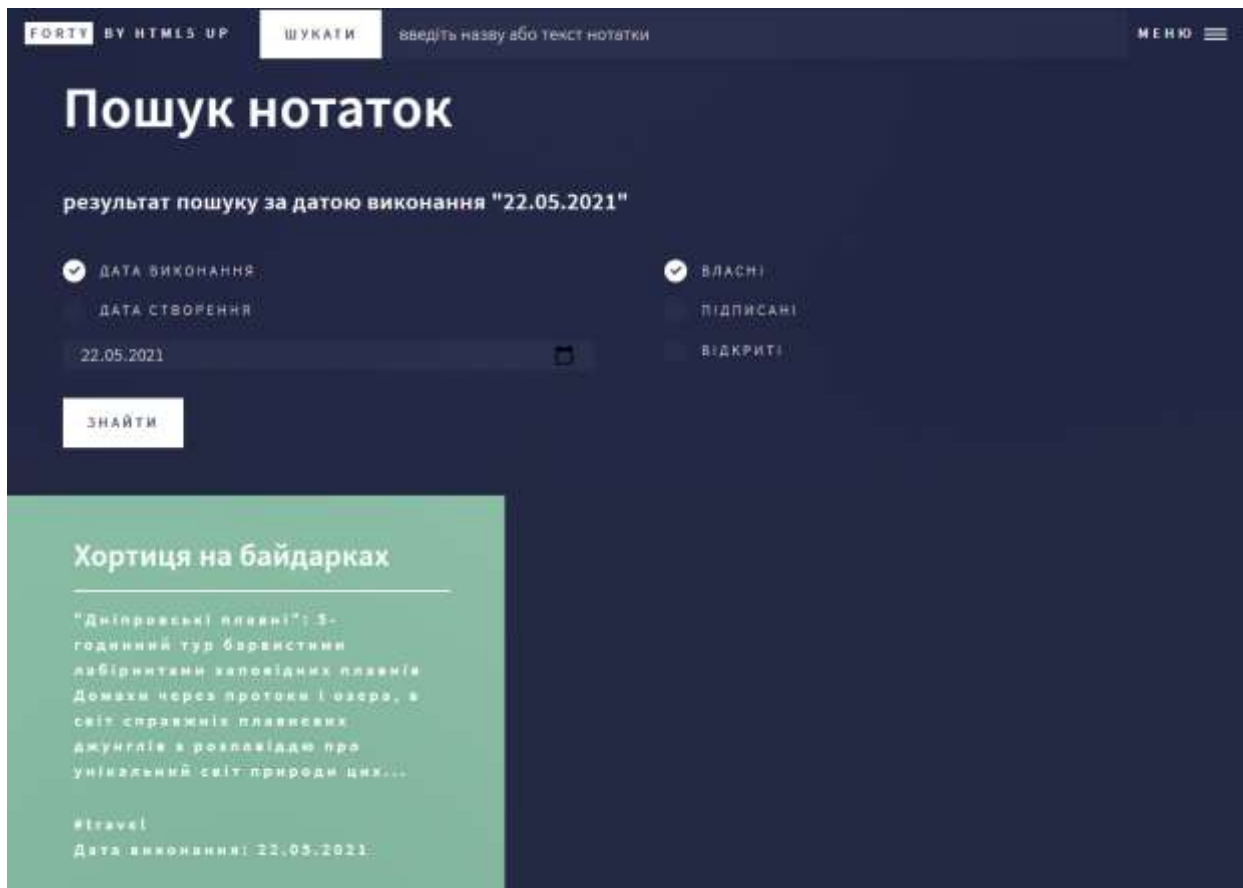


Рисунок 4.14 – Результат пошуку нотаток за датою виконання

ВИСНОВКИ

У дипломній кваліфікаційній роботі розглянуто проблему керування нотатками. Для цього розглянуто основні типи поведінки користувачів при роботі з нотатками, виділено вимоги до сучасного програмного забезпечення керування нотатками, виконано огляд програмних аналогів, визначено основні позитивні та негативні риси відповідних аналогів.

Виконано проєктування програмного забезпечення. Для цього визначено вимоги до програмного забезпечення з представленням відповідної діаграми прецедентів, виконано вибір мови програмування, в якості якої на основі порівняння обрано мову Python, виконано проєктування вебзастосунку було виконано на основі використання вебфреймворку Django, проаналізовано архітектурний шаблон MVT та процес взаємодії в межах вебфреймворку Django для формування відповідей на запитання користувача, виконано проєктування структури бази даних на основі використання MySQL в якості системи керування базою даних та побудови схеми бази даних, визначено структурну схему програмного проєкту на основі застосування шаблону MVT.

Реалізовано програмне забезпечення керування нотатками з роздільним доступом у вигляді вебзастосунку. Описано процес створення програмного забезпечення з основними поданнями, моделями, шаблонами та менеджером URL.

Розроблений вебзастосунок дозволяє виконувати створення нотатки з визначенням назви, тексту та дати виконання, перегляд нотатки, редагування тексту, назви нотатки та дати виконання, розфарбовувати нотатки, вилучати нотатки, виконувати пошук нотаток за назвою і текстом, визначення тегів нотатки, створення тегів, відображати нотатки за тегом, всі власні нотатки, відкривати зовнішній доступ до нотатки для всіх користувачів та за посиланням, отримувати доступу до нотатки за посиланням, відображати всі відкриті нотатки, виконувати пошук серед всіх відкритих

нотаток, відображати нотатки, на які підписався користувач, відображати нотатки за календарем за датою створення і за датою виконання.

Усі завдання, які було визначено на дипломну кваліфікаційну роботу, повністю виконано.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Le Cunff, A.-L. How to choose the right note-taking app [Electronic resource] / Anne-Laure Le Cunff. – Access mode : <https://nesslabs.com/how-to-choose-the-right-note-taking-app>
2. Guinness, H. The 5 best note-taking apps of 2021 [Electronic resource] / Harry Guinness. – Access mode : <https://zapier.com/blog/best-note-taking-apps/>
3. Best Note Taking App – Organize Your Notes with Evernote [Electronic resource]. – Access mode : <https://evernote.com/intl/en/>
4. Turner, B. Best note-taking apps of 2021 [Electronic resource] / Brian Turner. – Access mode : <https://www.techradar.com/best/best-note-taking-app>
5. Duffy, J. Evernote Review [Electronic resource] / Jill Duffy. – Access mode : <https://www.pcmag.com/reviews/evernote>
6. Google Keep [Electronic resource]. – Access mode : <https://keep.google.com/>
7. McOmie, A. Google Keep review [Electronic resource] / Alex McOmie. – Access mode : <https://www.techradar.com/reviews/google-keep>
8. Ларман, К. Применение UML 2.0 и шаблонов проектирования [Текст] / К. Ларман. – 3-е изд. – М. : Вильямс, 2006. – 736 с.
9. Лутц, М. Программирование на Python [Текст] : пер. с англ. / М. Лутц. – 4-е изд. – СПб. : Символ-Плюс, 2011. – 992 с.
10. Скит, Дж. С# для профессионалов [Текст] : тонкости программирования, 3-е издание, новый перевод / Дж. Скит. – М. : Вильямс, 2014. – 608 с.
11. Бизли, Д. М. Python. Подробный справочник, 4-е издание [Текст] / Д. М. Бизли. – СПб. : Символ-Плюс, 2010. – 864 с.
12. Саммерфилд, М. Программирование на Python 3. Подробное руководство [Текст] / М. Саммерфилд. – СПб. : Символ-Плюс, 2009. – 608 с.
13. Chaturvedi, J. A Brief introduction to Django MVT framework

[Electronic resource] / J. Chaturvedi. – Access mode :
<https://medium.com/@jaychaturvedi18/a-brief-introduction-to-django-mvt-framework-8ef46cc321ab>

14. Васвани, В. MySQL [Текст] : использование и администрирование / В. Васвани. – М. : Питер, 2011. – 368 с.

15. Forty | HTML5 UP [Electronic resource]. – Access mode :
<https://html5up.net/forty>.

ДОДАТОК А
Технічне завдання

Вступ

У результаті виконання дипломної кваліфікаційної роботи заплановано розробити програмне забезпечення, що буде надавати доступ для читання нотаток з роздільним доступом, а також виконувати їх пошук, редагувати за наявності відповідного рівня доступу.

A.1 Підстави для розробки

Підставою для виконуваної розробки в межах дипломної кваліфікаційної роботи бакалавра є завдання, що було визначене за темою «Програмне забезпечення керування нотатками з роздільним доступом» і затверджене у відповідності з наказом № 103 від 30 березня 2021 р. за Національним університетом «Запорізька політехніка».

A.2 Призначення розробки

Програмне забезпечення призначене для виконання роботи з персональними нотатками, перегляду нотаток інших користувачів за посиланням, а також перегляду відкритих для всіх користувачів нотаток.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

За результатами виконання дипломної кваліфікаційної роботи бакалавра програмне забезпечення, що розробляється, повинно мати наступні функціональні характеристики:

- авторизація користувача;
- створення нотатки з визначенням назви, тексту та дати виконання;
- перегляд нотатки;

- редагування тексту, назви нотатки та дати виконання;
- розфарбовування нотатки;
- вилучення нотатки;
- пошук нотаток за назвою і текстом;
- визначення тегів нотатки;
- створення тегу;
- відображення нотаток за тегом;
- відображення всіх власних нотаток;
- відкриття зовнішнього доступу до нотатки для всіх користувачів;
- відкриття зовнішнього доступу до нотатки за посиланням;
- відображення всіх відкритих нотаток;
- пошук серед всіх відкритих нотаток;
- підписування на нотатку за посиланням;
- відображення нотаток, на які підписався користувач;
- відображення нотаток за календарем за датою створення;
- відображення нотаток за календарем за датою виконання;
- реєстрація користувача.

A.3.2 Вимоги до надійності

Для забезпечення надійності має виконуватися валідація даних, що було введено користувачем в поля форм.

A.3.3 Умови експлуатації

Експлуатація програмного забезпечення має виконуватися на основі клієнт-серверної моделі. Для цього потрібно виділити окремий комп'ютер або скористатися виділеним сервером, який буде забезпечувати підтримку реалізації вебсервера. Вебсервер має бути доступним для запитів користувачів постійно.

А.3.4 Вимоги до складу та параметрів технічних засобів

Склад та параметри технічних засобів вебсервера було визначено для реалізації можливості доступу 300 користувачів одночасно. Відповідно вимоги можуть бути знижені, якщо потрібне забезпечення меншої кількості запитів, а для більшої кількості одночасних запитів вимоги навпаки мають бути збільшені.

Мінімальні системні вимоги до вебсервера включають:

- процесор: 4 ядра, тактова частота – 2,9 ГГц;
- оперативна пам'ять – 10 Гб;
- платформа – 32-розрядна або 64-розрядна;
- жорсткий диск – 1 Гб вільного простору.

А.3.5 Вимоги до транспортування та збереження

Безпосередньо програмне забезпечення має транспортуватися та зберігатися через репозиторій вебсервера. Збереження на окремих фізичних носіях не передбачається.

А.3.6 Вимоги до програмної документації

Основою розробки даного програмного проєкту є технічне завдання на нього.

Опис прийнятих рішень виконується в пояснювальній записці. У розділі «Експлуатація та тестування програми» має бути представлено опис процесу роботи з програмою. Інших програмних документів не передбачається.

A.4 Порядок контролю та приймання

Календарний план завдання на дипломну кваліфікаційну роботу визначає набір етапів, за якими має забезпечуватися виконання, а також результати, які мають бути отримані за виконання відповідного етапу. Саме основі даних результатів має забезпечуватися приймання роботи.

ДОДАТОК Б
Текст програми

Б.1 Текст файла models.py

```

from django.db import models
from django.urls import reverse
from django.contrib.auth.models import User

from myrecord.funcs import ShortText

class Record(models.Model):
    name = models.CharField(max_length=255)
    recordtext = models.TextField()
    acesssmode = models.IntegerField()
    acount = models.IntegerField()
    color = models.IntegerField()
    creation = models.DateTimeField(auto_now=True)
    execution = models.DateField()
    modification = models.DateTimeField()
    author = models.ForeignKey (User, on_delete=models.CASCADE)
    tags = models.ManyToManyField(Tag)

    def __str__(self):
        return self.name

    def get_absolute_url(self):
        return reverse('record', args=[self.id])

    def shortinfo(self):
        return ShortText(self.recordtext, self.tags)

    def shortinfo2(self):
        return ShortText(self.recordtext, self.tags,
self.author)

class Tag(models.Model):
    title = models.CharField(max_length=30)
    tcount = models.IntegerField()

    def __str__(self):
        return self.title

```

```

class ARAccess(models.Model):
    accountId = models.ForeignKey (User,
on_delete=models.CASCADE)
    recordId = models.ForeignKey (Record,
on_delete=models.CASCADE)
    creation = models.DateTimeField(auto_now=True)

```

Б.2 Текст файла urls.py

```

from django.contrib import admin
from django.urls import path
from django.conf.urls import url

from myrecord.views import IndexView
from myrecord.views import SearchView, SearchResultView,
SearchOpenResultView
from myrecord.views import LoginView, LogoutView
from myrecord.views import RecordView, NewRecordView,
EditRecordView, RecordDeleteView
from myrecord.views import PersonalView, SharedView
from myrecord.views import TagsView, TagRecordsView,
TagNumRecordsView

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', IndexView.as_view(), name='index'),
    path('index', IndexView.as_view(), name='index'),
    path('search', SearchView.as_view(), name='search'),
    path('login', LoginView.as_view(), name='login'),
    path('logout', LogoutView.as_view(), name='logout'),
    url(r'^record/(?P<pk>\d+)$', RecordView.as_view(),
name='read_record'),
    path('record/new', NewRecordView.as_view(),
name='new_record'),
    url(r'^record/edit/(?P<pk>\d+)$', EditRecordView.as_view(),
name='edit_record'),
    url(r'^record/delete/(?P<pk>\d+)$',
RecordDeleteView.as_view(), name='delete_record'),
    path('personal', PersonalView.as_view(), name='person'),
    path('shared', SharedView.as_view(), name='shared'),
    path('sresult', SearchResultView.as_view(), name='sresult'),

```

```

        path('sresult2', SearchOpenResultView.as_view(),
name='sresult2'),
        path('tags', TagView.as_view(), name='tags'),
        path('tagrecords', TagRecordsView.as_view(),
name='tagrecords'),
        url(r'^tag/(?P<pk>\d+)$', TagNumRecordsView.as_view(),
name='tagnum'),

]

```

Б.3 Текст файла /views/record.py

```

from datetime import datetime

from django.urls import reverse_lazy
from django.contrib.auth.mixins import LoginRequiredMixin
from django.utils import timezone
from django.views.generic.edit import CreateView, UpdateView,
DeleteView
from django.views.generic.detail import DetailView
from django.views.generic import TemplateView

from myrecord.models import Record

class NewRecordView(LoginRequiredMixin, CreateView):
    template_name = 'record/newrecord.html'
    model = Record
    fields = [
        'name',
        'recordtext',
        'accessmode',
        'acount',
        'color',
        'creation',
        'execution',
        'modification',
        'author',
        'tags',
    ]
    success_url = '/personal'
    login_url = '/login'

```

```

def get_form(self, **kwargs):
    form = super(NewRecordView, self).get_form()
    for visible in form.visible_fields():
        visible.field.widget.attrs['class'] = 'form-control'
    return form

def post(self, request, *args, **kwargs):
    newrecfrm = self.get_form (request.POST)
    if newrecfrm.is_valid():
        rec = newrecfrm.save()
        rec.save()
        return
    HttpResponseRedirect(reverse_lazy(success_url))
    return render(request, template_name, {'form': form})

def get_initial(self, *args, **kwargs):
    bform = super(NewRecordView, self).get_initial(**kwargs)
    bform ['acessmode'] = 1
    bform ['acount'] = 0
    bform ['color'] = 1
    bform ['author'] = self.request.user
    return bform

class EditRecordView(LoginRequiredMixin, UpdateView):
    template_name = 'record/editrecord.html'
    model = Record
    fields = [
        'name',
        'recordtext',
        'acessmode',
        'acount',
        'color',
        'creation',
        'execution',
        'modification',
        'author',
        'tags',
    ]
    success_url = '/personal'
    login_url = '/login'

def get_object(self, *args, **kwargs):

```

```

        rec = get_object_or_404(Record, pk=self.kwargs['pk'])
        return rec

    def get_form(self, **kwargs):
        form = super(EditRecordView, self).get_form()
        for visible in form.visible_fields():
            visible.field.widget.attrs['class'] = 'form-control'

        return form

    def post(self, request, **kwargs):
        rec = Record.objects.get(pk=kwargs.get('pk'))
        rec.modification = datetime.now()
        rec.save()

        return super(EditPackageView, self).post(request,
**kwargs)

```

```

class RecordView (LoginRequiredMixin, DetailView):
    model = Record
    template_name = 'record/read_record.html'

```

```

class RecordDeleteView(LoginRequiredMixin, DeleteView):
    model = Record
    success_url = reverse_lazy('personal')

```

Б.4 Текст файлы /views/records.py

```

from django.views.generic.list import ListView
from myrecord.models import Record

class PersonalView (LoginRequiredMixin, ListView):
    model = Record
    template_name = 'personal.html'

    def get_queryset(self, *args, **kwargs):
        rl = super(PersonalView, self).get_queryset(*args,
**kwargs)

        rl = rl.filter(author = self.request.user)

```

```

        r1 = r1.filter(acesmode = 1)
        r1 = r1.order_by("-id")
        return r1

class SharedView (LoginRequiredMixin, ListView):
    model = Record
    template_name = 'shared.html'

    def get_queryset(self, *args, **kwargs):
        acs = ARAccess.objects.all()
        r1 = super(SharedView, self).get_queryset(*args,
**kwargs)

        acs = acs.filter(accountId = self.request.user.id)
        r1 = r1.filter(acesmode = 2)
        aids=acs.values_list("recordId" , flat=True)["id"]
        r1.filter(pk__in=aids)
        r1 = r1.order_by("-id")
        return r1

```

Б.5 Текст файла /views/index.py

```

from django.views.generic.list import ListView
from myrecord.models import Record

class IndexView (ListView):
    model = Record
    template_name = 'index.html'

    def get_queryset(self, *args, **kwargs):
        r1 = super(IndexView, self).get_queryset(*args,**kwargs)
        r1 = r1.filter(acesmode = 3)
        r1 = r1.order_by("-account")
        return r1

```

Б.6 Текст файла /views/tags.py

```

from django.views.generic.list import ListView
from myrecord.models import Tag

```

```

class TagView (ListView):
    model = Tag
    template_name = 'tags.html'

    def get_queryset(self, *args, **kwargs):
        tg = super(TagView, self).get_queryset(*args, **kwargs)
        tg = tg.order_by("-account")
        return tg

```

Б.7 Текст файла read_record.html

```

{% load static %}

<!DOCTYPE HTML>
<!--
    Forty by HTML5 UP
    html5up.net | @ajlkn
    Free for personal and commercial use under the CCA 3.0
    license (html5up.net/license)
-->
<html>
    <head>
        <title>{{ record.name }}</title>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width,
initial-scale=1, user-scalable=no" />
        <link rel="stylesheet" href="{% static
'myrecord/assets/css/main.css' %}" />
        <noscript><link rel="stylesheet" href="{% static
'myrecord/assets/css/noscript.css' %}" /></noscript>
    </head>

    <body class="is-preload">

        <div id="wrapper">

            <header id="header" class="alt">
                <a href="{% url 'index' %}"
class="logo"><strong>Forty</strong> <span>by HTML5 UP</span></a>

                <nav>

```

```

                                <form action='sresult'
method="post" >
    <input type="submit" value="Шукати" class="primary" />
                                <input
type="text" size="90" name="name" id="name" value="Введіть назву або
текст нотатки" />

                                </form>
                                <a href="#menu">Меню</a>
                                </nav>
                                </header>

                                <nav id="menu">
                                <ul class="links">
                                    <li><a href="{% url
'person' %}">Власні</a></li>
                                    <li><a href="{% url
'shared' %}">Підписані</a></li>
                                    <li><a href="{% url 'index'
%}">Доступні</a></li>
                                    <li><a href="{% url
'search' %}">Пошук</a></li>
                                    <li><a href="{% url 'tags'
%}">Теги</a></li>
                                </ul>
                                <ul class="actions stacked">
                                    <li><a href="{% url
'logout' %}" class="button primary fit">Вийти</a></li>
                                </ul>
                                </nav>

                                <section id="banner"
class="{{ 'style'|add: record.color }}">
                                <div class="inner">
                                    <span class="image">
                                        
                                    </span>
                                    <header class="major">
                                        <h1>>{{ record.name
}}</h1>
                                </div>
                                </header>

```

```

                                <div class="content">
                                    <p>{{
record.recordtext }}</p>
                                </div>
                            </div>
                        </section>
                        <div id="main">

                            <section id="one">
                                <div class="inner">
                                    <header
class="major">

                                        <h2>Теги</h2>

                                            </header>
                                            {% for tg in
record.tags %}

                                                <p><a href="{%
url 'tag/'|add: tg.id %}">{{ tg.title }}</a></p>

                                                    {% endfor %}
                                                    <p>Створено: {{
record.creation |date }}</p>

                                                        <p>Дата
виконання: {{ record.execution|date }}</p>
                                                            <a href="{% url record.secret %}"
class="button">Посилання</a><br><br>
                                                                <a href="{% url 'record/edit/'|add: record.id %}"
class="button">Редагувати</a>

                                                                    </div>
                                                                </section>

                                                                    </div>

                                <script src="{% static 'assets/js/jquery.min.js'
%}"></script>

                                    <script src="{% static
'assets/js/jquery.scrolly.min.js' %}"></script>
                                        <script src="{% static
'assets/js/jquery.scrollex.min.js' %}"></script>
                                            <script src="{% static 'assets/js/browser.min.js'
%}"></script>

                                                <script src="{% static
'assets/js/breakpoints.min.js' %}"></script>
                                                    <script src="{% static 'assets/js/util.js'
%}"></script>

```

```

        <script src="{% static 'assets/js/main.js'
%}"></script>
    </div>

    </body>
</html>

```

Б.8 Текст файлу personal.html

```

{% load static %}

<!DOCTYPE HTML>
<!--
    Forty by HTML5 UP
    html5up.net | @ajlkn
    Free for personal and commercial use under the CCA 3.0
license (html5up.net/license)
-->
<html>
    <head>
        <title>Приватні нотатки</title>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width,
initial-scale=1, user-scalable=no" />
        <link rel="stylesheet" href="{% static
'myrecord/assets/css/main.css' %}" />
        <noscript><link rel="stylesheet" href="{% static
'myrecord/assets/css/noscript.css' %}" /></noscript>
    </head>

    <body class="is-preload">

        <div id="wrapper">

            <header id="header" class="alt">
                <a href="{% url 'index' %}"
class="logo"><strong>Forty</strong> <span>by HTML5 UP</span></a>
                <nav>

                    <form action='sresult'
method="post" >

```

```

        <input type="submit" value="Шукати" class="primary" />
        <input
type="text" size="90" name="name" id="name" value="введіть назву або
текст нотатки" />

        </form>
        <a href="#menu">Меню</a>
    </nav>
</header>

<nav id="menu">
    <ul class="links">
        <li><a href="{% url
'person' %}">Власні</a></li>
        <li><a href="{% url
'shared' %}">Підписані</a></li>
        <li><a href="{% url 'index'
%}">Доступні</a></li>
        <li><a href="{% url
'search' %}">Пошук</a></li>
        <li><a href="{% url 'tags'
%}">Теги</a></li>
    </ul>
    <ul class="actions stacked">
        <li><a href="{% url
'logout' %}" class="button primary fit">Вийти</a></li>
    </ul>
</nav>

<section id="banner">
    <div class="inner">
        <header class="major">
            <h1>Нова нотатка</h1>
        </header>
        <div class="content">
            <ul class="actions">
                <li><a href="{%
url 'record/new' %}" class="button next scrolly">Створити</a></li>
            </ul>
        </div>
    </div>
</section>

```

```

<div id="main">

    <section id="one"
class="tiles">
        {% for rec in
record_list %}

            <article
class="{{ 'art'|add: rec.color }}">

                <span
class="image">

                </span>
                <header
class="major">

                    <h3><a
href="{{ rec.get_absolute_url }}" class="link">{{ rec.name }}</a></h3>
                    <p>{% if
forloop.counter|divisibleby:2 %} rec.shortinfo[0] {% else %}
rec.shortinfo[1] {% endif %}</p>

                </header>
            </article>
            {% endfor %}

        </section>

    </div>

    <script src="{% static 'assets/js/jquery.min.js'
%}"></script>

    <script src="{% static
'assets/js/jquery.scrolly.min.js' %}"></script>
    <script src="{% static
'assets/js/jquery.scrollex.min.js' %}"></script>
    <script src="{% static 'assets/js/browser.min.js'
%}"></script>

    <script src="{% static
'assets/js/breakpoints.min.js' %}"></script>
    <script src="{% static 'assets/js/util.js'
%}"></script>

    <script src="{% static 'assets/js/main.js'
%}"></script>

    </div>

```

```

    </body>
</html>

```

Б.9 Текст файлу index.html

```

{% load static %}

<!DOCTYPE HTML>
<!--
    Forty by HTML5 UP
    html5up.net | @ajlkn
    Free for personal and commercial use under the CCA 3.0
    license (html5up.net/license)
-->
<html>
    <head>
        <title>Відкриті нотатки</title>
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width,
initial-scale=1, user-scalable=no" />
        <link rel="stylesheet" href="{% static
'myrecord/assets/css/main.css' %}" />
        <noscript><link rel="stylesheet" href="{% static
'myrecord/assets/css/noscript.css' %}" /></noscript>
    </head>

    <body class="is-preload">

        <div id="wrapper">

            <header id="header" class="alt">
                <a href="{% url 'index' %}"
class="logo"><strong>Forty</strong> <span>by HTML5 UP</span></a>

            <nav>

                <form
action='sresult2' method="post" >
                <input type="submit" value="Шукати" class="primary" />
                <input
type="text" size="90" name="name" id="name" value="введіть назву або
текст нотатки" />

```

```

        </form>
        <a href="#menu">Меню</a>
    </nav>
</header>

<nav id="menu">
    <ul class="links">
        {% if user.is_authenticated
%}
            <li><a href="{% url
'person' %}">Власні</a></li>
            <li><a href="{% url
'shared' %}">Підписані</a></li>
            {% endif %}
            <li><a href="{% url 'index'
%}">Доступні</a></li>
            {% if user.is_authenticated
%}
            <li><a href="{% url
'search' %}">Пошук</a></li>
            <li><a href="{% url 'tags'
%}">Теги</a></li>
            {% endif %}
        </ul>

        <ul class="actions stacked">
            {% if user.is_authenticated
%}
            <li><a href="{% url
'logout' %}" class="button primary fit">Вийти</a></li>
            {% else %}
            <li><a href="{% url 'login'
%}" class="button primary fit">Авторизуватись</a></li>
            {% endif %}

        </ul>
    </nav>

    <section id="banner">
        <div class="inner">
            <header class="major">
                <h1>Відкриті
нотатки</h1>

```

```

</header>

</div>
</section>

<div id="main">

    <section id="one"
class="tiles">
        {% for rec in
record_list %}

            <article
class="{{ 'art'|add: rec.color }}">

                <span
class="image">

                </span>

                <header
class="major">

                    <h3><a
href="{{ rec.get_absolute_url }}" class="link">{{ rec.name }}</a></h3>
                    <p> {% if
forloop.counter|divisibleby:2 %} rec.shortinfo2[0] {% else %}
rec.shortinfo2[1] {% endif %}</p>

                </header>
            </article>
            {% endfor %}
        </section>
    </div>

    <script src="{% static 'assets/js/jquery.min.js'
%}"></script>

    <script src="{% static
'assets/js/jquery.scrolly.min.js' %}"></script>
    <script src="{% static
'assets/js/jquery.scrollex.min.js' %}"></script>
    <script src="{% static 'assets/js/browser.min.js'
%}"></script>

    <script src="{% static
'assets/js/breakpoints.min.js' %}"></script>

```

```
%}"></script>
<script src="{% static 'assets/js/util.js'
%}"></script>
</div>

</body>
</html>
```

ДОДАТОК В
Слайди презентації

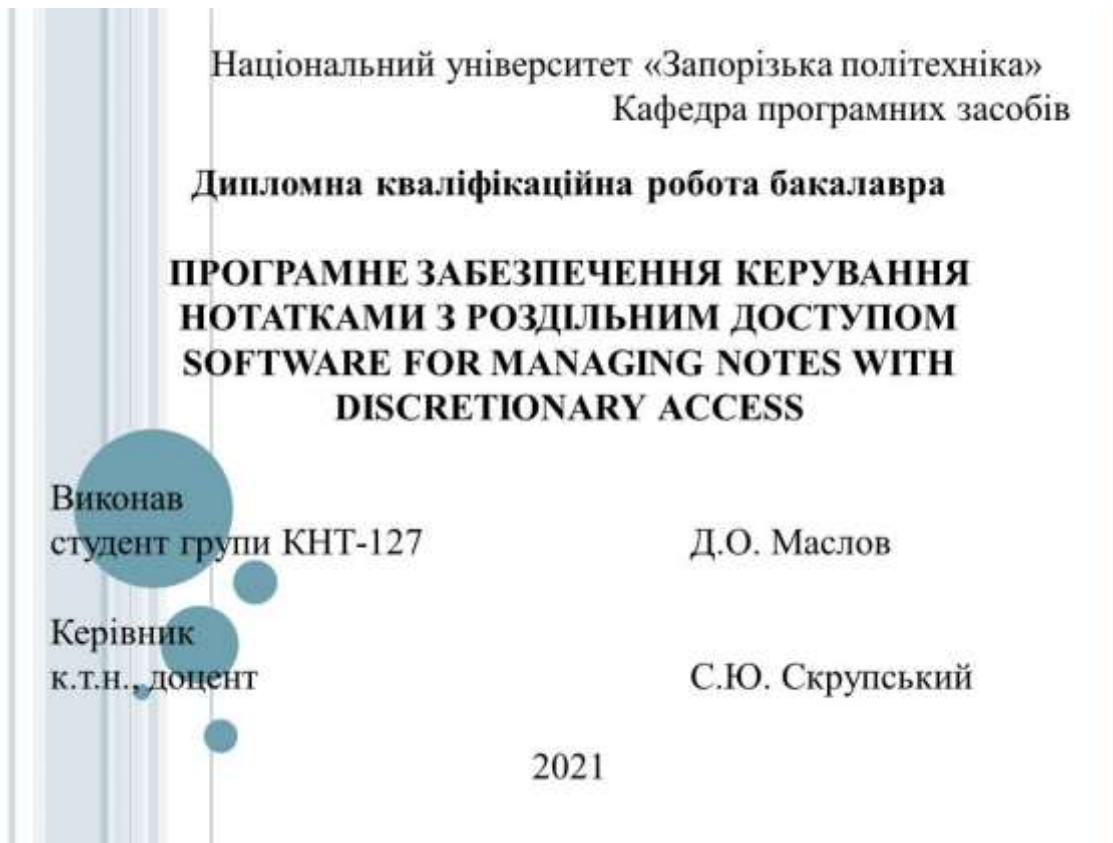


Рисунок В.1 – Слайд № 1

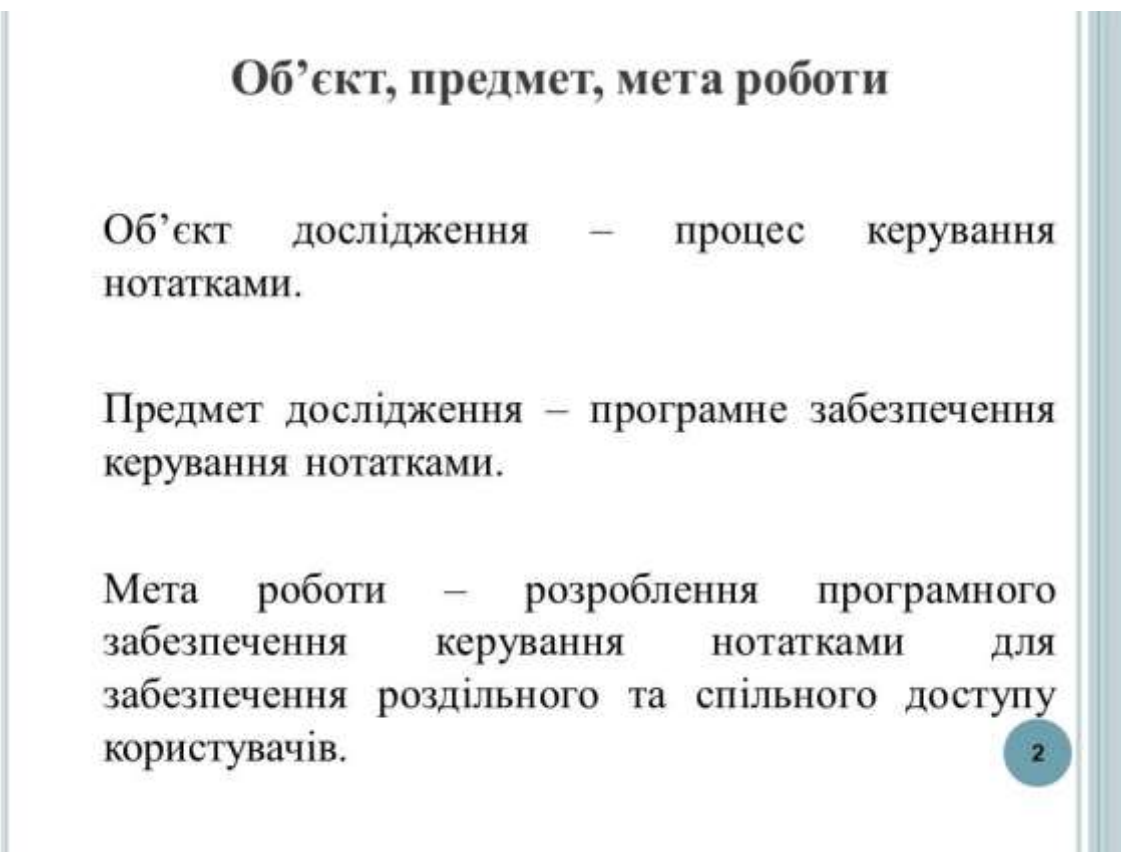


Рисунок В.2 – Слайд № 2

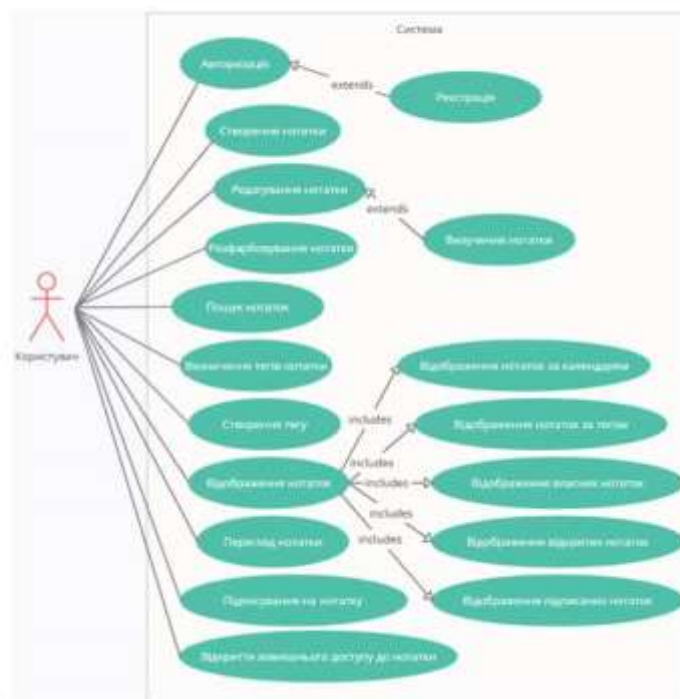
Завдання роботи

- ✓ визначення вимог до програмного забезпечення керування нотатками;
- ✓ вибір мови і засобів розробки;
- ✓ проєктування програмного забезпечення;
- ✓ реалізація програмного забезпечення;
- ✓ тестування і налагодження програмного забезпечення.

3

Рисунок В.3 – Слайд № 3

Діаграма прецедентів роботи з системою



4

Рисунок В.4 – Слайд № 4

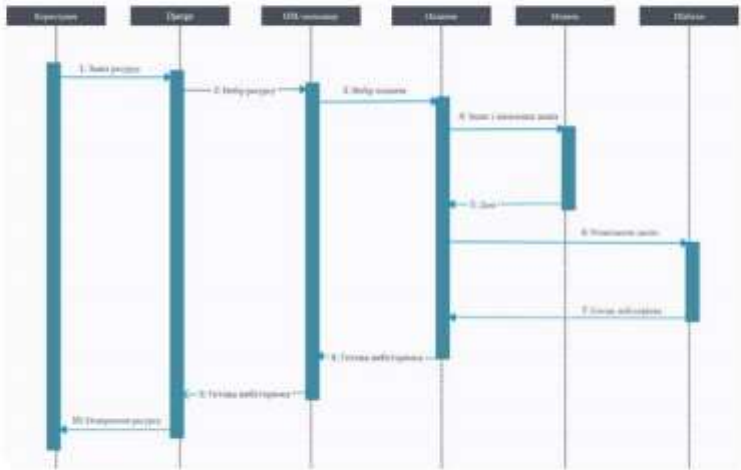
Вибір мови програмування

Показник	C#	Python
Веброзробка	+	+
Вебфреймворки	+	+
Клас	мультипарадигмальний	об'єктно-орієнтований (з можливістю використання інших парадигм)
Синтаксична простота	-	+
Підтримка	відносно невелика кількість бібліотек	велика кількість вільних бібліотек
Вартість використання	потребує сплати за бізнес-використання	безкоштовно

5

Рисунок В.5 – Слайд № 5

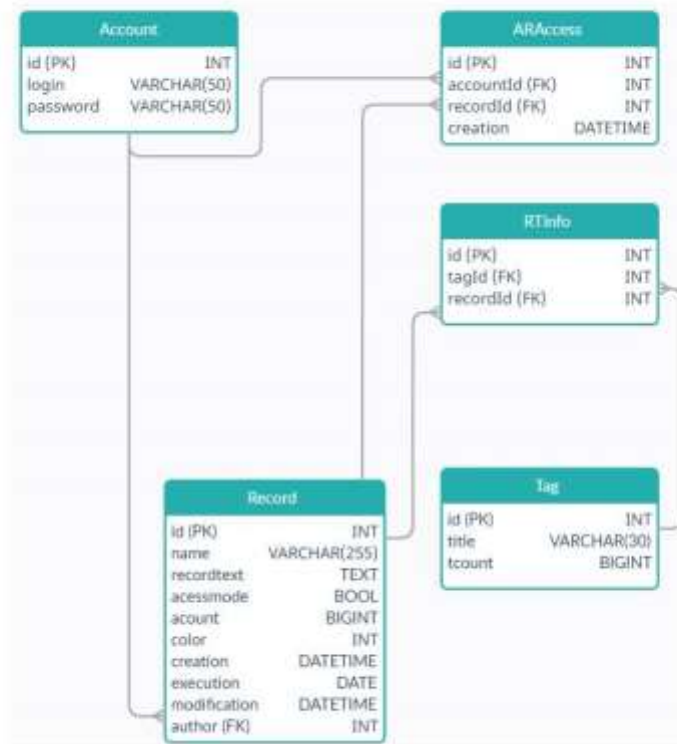
Послідовність роботи над запитами користувача у вебфреймворку Django



6

Рисунок В.6 – Слайд № 6

Схема бази даних



7

Рисунок В.7 – Слайд № 7

Діаграма класів подань



8

Рисунок В.8 – Слайд № 8

Робота з власними нотатками

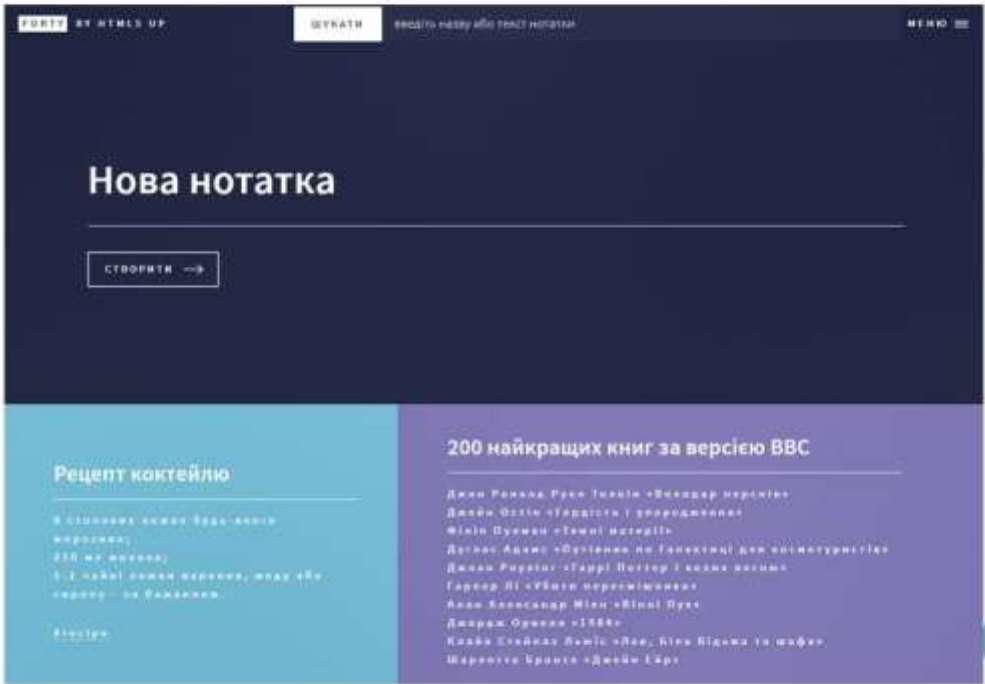


Рисунок В.9 – Слайд № 9

Перегляд та редагування нотаток

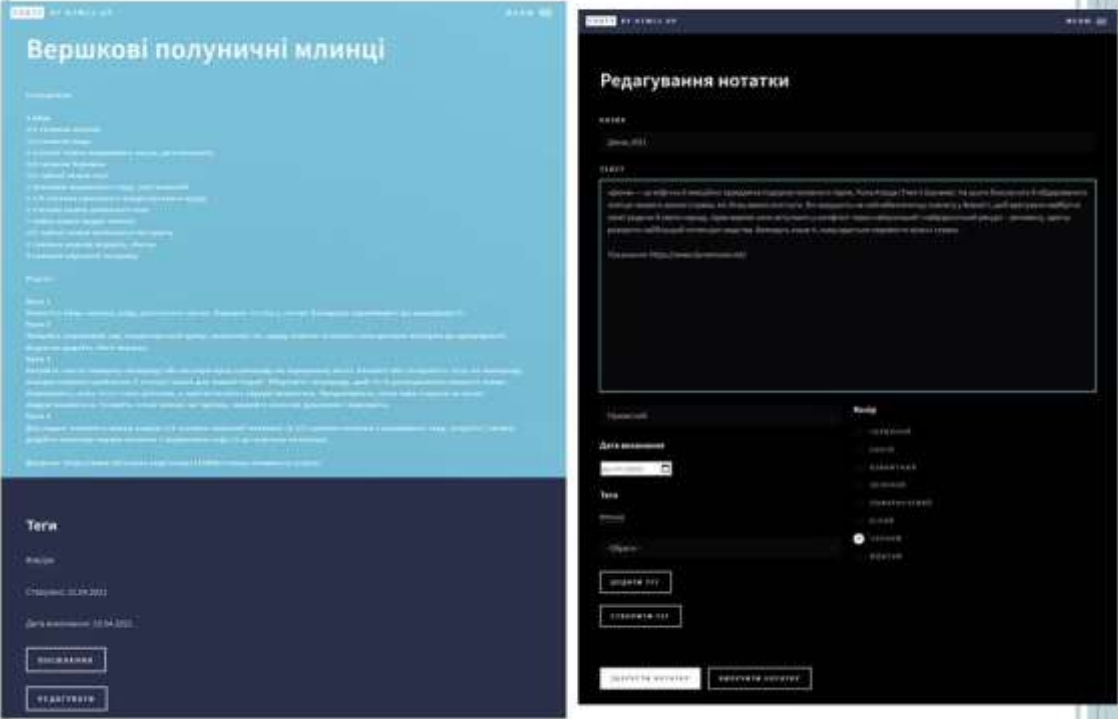
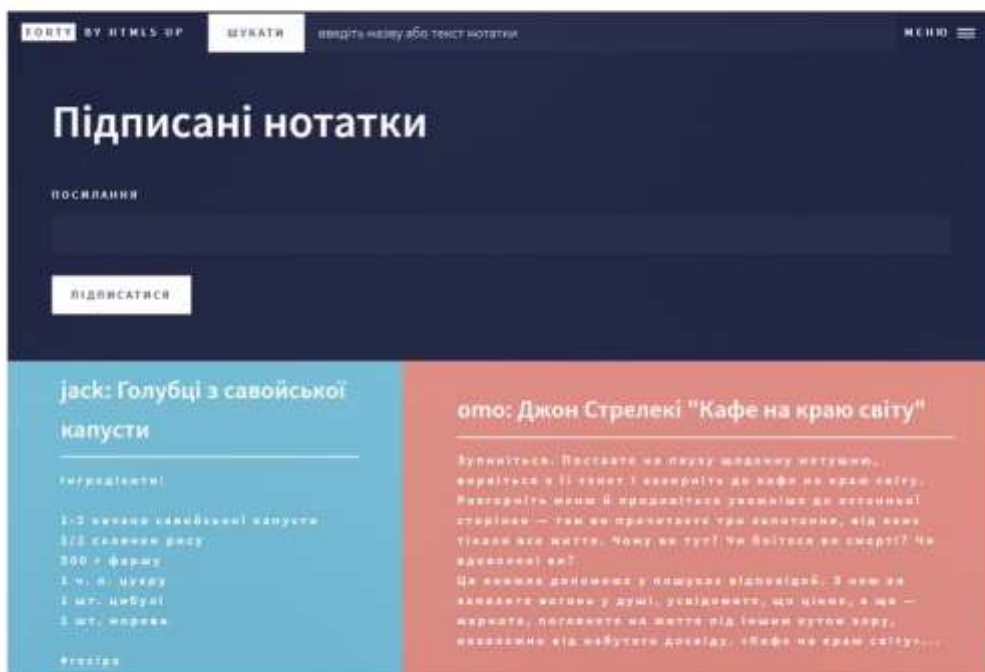


Рисунок В.10 – Слайд № 10

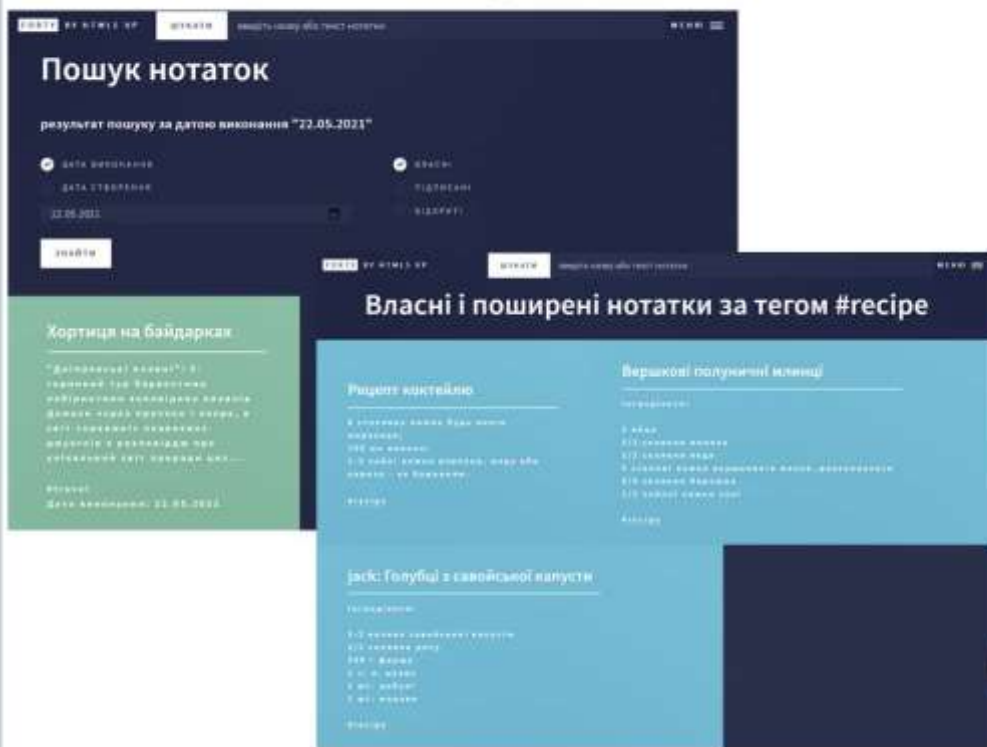
Робота з підписаними нотатками



11

Рисунок В.11 – Слайд № 11

Пошук нотаток



12

Рисунок В.12 – Слайд № 12

Висновки

У дипломній кваліфікаційній роботі розглянуто проблему керування нотатками. Для цього розглянуто основні типи поведінки користувачів при роботі з нотатками, виділено вимоги до сучасного програмного забезпечення керування нотатками, виконано огляд програмних аналогів, визначено основні позитивні та негативні риси відповідних аналогів.

Виконано проектування програмного забезпечення. Для цього визначено вимоги до програмного забезпечення з представленням відповідної діаграми прецедентів, виконано вибір мови програмування, в якості якої на основі порівняння обрано мову Python, виконано проектування вебзастосунку було виконано на основі використання вебфреймворку Django, проаналізовано архітектурний шаблон MVT та процес взаємодії в межах вебфреймворку Django для формування відповідей на запитання користувача, виконано проектування структури бази даних на основі використання MySQL в якості системи керування базою даних та побудови схеми бази даних, визначено структурну схему програмного проєкту на основі застосування шаблону MVT.

Реалізовано програмне забезпечення керування нотатками з роздільним доступом у вигляді вебзастосунку. Описано процес створення програмного забезпечення з основними поданнями, моделями, шаблонами та менеджером URL.

Усі завдання, які було визначено на дипломну кваліфікаційну роботу, повністю виконано.