МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Комп'ютерних наук, фізики та математики
Кафедра комп'ютерних наук і програмної інженерії

РОЗРОБЛЕННЯ ПРОГРАМНИХ ДОКУМЕНТІВ ДЛЯ ВИКЛАДАЧІВ ХДУ

Кваліфікаційна робота (проєкт)

на здобуття ступеня вищої освіти «бакалавр»

Виконав: здобувач 4 курс 441 група
Спеціальності 121 Програмна інженерія
Освітньо-професійної (наукової) програми 122 Комп'ютерні науки першого (бакалаврського) рівня вищої освіти
Широкопояса Андрія Олександровича

Керівник Шерман М.І.
Рецензент Лобода О.М.

Херсон – 2022

ВСТУП

Важливим елементом організації роботи навчального закладу є автоматизація роботи з документами, розглянутий у роботі В.Пайкеса «Методика складання розкладу занять у загальноосвітній установі». Раціонально автоматизовані процеси рутинної роботи сприяє ефективності НВП, зниженню і ліквідації перевантажень учнів, підвищенню працездатності студентів та викладачів. Робота з документами в освітній установі є одним із найважливіших і діючих видів планування навчально-виховної роботи, одним з важливих процесів, що визначає роботу учнівського і вчительського колективів, адміністрації і всієї школи в цілому. У роботі з даного питання необхідно керуватися практичною доцільністю. При автоматизації генерування документів важливо враховувати те, наскільки цей процес є повторюваним в освітньому житті, наприклад, якщо шаблон документа використовується кілька разів на рік, то використання даної системи для даного шаблону не є доцільним і навпаки.

Задача друку документів є надзвичайно розповсюдженою в умовах бюрократії, яка неодмінно супроводжує державні установи. У загальній постановці вона є процесом розподілу деякого кінцевого набору подій в часі в умовах ресурсних та інших обмежень.

Задача друку документів для викладачів навчальних закладів має давню історію. Вона цікавила людей, зайнятих формуванням навчального процесу з моменту появи освітніх установ масового характеру: шкіл, гімназій, коледжів, інститутів і університетів. Традиційно це завдання вирішувалося без використання технічних засобів людиною, яка на підставі певних критеріїв, документувала події та дані на папері, що задовольняє певним вимогам. При достатньо великому наборі початкових

даних і множині обмежуючих чинників, процес формування документу є досить складним і трудомістким заняттям. Слід відмітити, що у наш час загальної автоматизації і комп'ютеризації виробництва і суспільства в цілому в багатьох освітніх установах завдання формування і друку документів робиться все одно вручну.

Найпопулярнішим та визнаним інструментом друку документів є продукт від компанії Microsoft – Microsoft Word. Це текстовий процесор із широкою функціональністю для оформлення тексту в різних форматах. Однією з переваг даного інструменту є інтуїтивність інтерфейсу. Це однозначно плюс, хоча в деяких випадках занадто простий і інтуїтивний інтерфейс може спонукати користувачів не отримувати більш глибокі та спеціалізовані навички.

Метою даної роботи є скорочення трудомісткості процесу формування та друку документів використовуючи шаблони та готову базу даних.

РОЗДІЛ 1
Постановка та огляд задачі

1.1. Текстові шаблони, огляд інструменту

Шаблон - це вже оформлений (повністю або частково) документ, в який досить тільки ввести певний невеликий об’єм даних. До складу шаблону входить форматування (в тому числі стилі), текст, об'єкти Word (зображення, діаграми, схеми та ін.), Стандартні блоки, а так же, призначені для користувача вкладки, команди і макроси.
Шаблони дозволяють налаштовувати всі необхідні параметри, які користувач хоче попередньо [2] застосувати до макету документів, стилі, форматування, вкладки і т.д. Потім він може легко створити новий документ на основі цього шаблону.
Доцільним є використання централізованого сховища шаблонів, куда користувач може зайти, вибрати необхідний йому документ, заповнити форму і отримати готовий екземпляр документу. Таку систему можна створити, використовуючи веб-технології.

1.2 Огляд інсуючих аналогів

Генератор документів — це, перш за все, сайт, тому відповідно для забезпечення його працездатності необхідні веб-сервер і система управління базами даних. В даному огляді вони не розглядаються детально; досить відзначити, що в якості веб-сервера стандартом де-факто є Apache і nginx. Що стосується СУБД, то їх вибір дуже великий: починаючи від SQLite, яка підходить для невеликих проектів, і закінчуючи Oracle Database, яка застосовується в великих системах, особливо пов'язаних з фінансовими операціями.

Крім вищевказаних компонентів необхідна основа сайту - движок, який являє собою програмне забезпечення, яке працює з базою даних і відображає користувачеві необхідну інформацію.

Движки можуть бути класифіковані такими способами:

Безкоштовні та вільні. Такі програмні рішення дозволяють підприємцям використовувати системи генерації документів, не витрачаючи великі грошові суми на створення і підтримання їх інфраструктури, в чому і полягає їх основна перевага. Недоліки безкоштовних рішень - відносно висока поширеність в інтернеті, а, отже, і підвищений інтерес до них зловмисників; неадаптированность таких рішень до конкретних завдань;

Платні. Схожі з попередніми рішеннями, однак, найчастіше, мають більш високою якістю коду і мають технічну підтримаю. Такі рішення зазвичай орієнтовані на середні і великі проекти.

Унікальні. Зазвичай створюються великими компаніями на замовлення для виконання конкретних завдань.

Розглянемо деякі проекти, що реалізують функціонал генерації документів.

Salesman.pro з модулем «Генератор документів». Salesman.prop., - Система для ведення клієнтської бази та управління продажами. Містить аналітичні звіти, розсилки, завдання і календар. Гнучка настройка прав доступу, можливо додавання позаштатних співробітників з повною ізоляції від основної бази, можливість настройки полів форм. «Генератор документів» - це модуль CRM, який дозволяє створювати і / або вести облік різних документів, прикріплювати їх до Організаціям і Угодах.

1С-Бітрікс з модулем «Генератор документів». 1С-Бітрікс - професійна система управління веб-проектами: сайтами компаній, інтернет-магазинами, соціальними мережами та спільнотами,

корпоративними порталами, системами оренди веб-додатків і іншими проектами. «Генератор документів», який формує файли з розширенням .dop., cx на основі створеного користувачем шаблону, а типові дані на зразок реквізитів, прізвищ, сум або таблиць з картинками автоматично підставляються в поля шаблону після запуску бізнес-процесу. Але дана система походить з країни-агресора – російської федерації, тому використання її на території України не може бути бажаним. Тому дана система йде вслід за російським кораблем.

1.3 Постановка задачі

На підставі вищевказаного зробимо постановку задачі. Необхідно розробити інформаційну систему, генерувати документи за шаблоном. Метою такої системи є реалізація наступних функцій:

- збереження шаблонів документів
- забезпечення доступу до системи працівникам організації
- забезпечення можливості внести та скорегувати дані які система автоматично буде підставляти в шаблон
- можливість завантажити готовий документ з шаблону

Для досягнення даної мети необходіно вирішати наступні задачі:

- розробити інформаційні моделі задачі
- обрати найбільш зручну для реалізації архітектуру програмного забезпечення
- добрати інструментальні та апаратні засоби з застосуванням найбільш поширеного програмного забезпечення з локалізацією до української мови

Те саме стосується і чергування уроків з одного предмета протягом тижня. Воно має бути рівномірним за днями тижня, особливо у випадку предметів з малою кількістю навчальних годин. У педагогічній практиці шкільні навчальні предмети за ступенем складності прийнято поділяти на чотири групи.

РОЗДІЛ 2
Проектування інформаційної системи

2.1 Схема роботи інформаційної системи

Виходячи з постанови задачі виділимо типовий робочий процес інформаційної системи. В ній зберігаються шаблони документів, які являють собою dop., cx-файли. В цих файлах побудована структура документу, але замість змінних даних в них знаходяться позначки виду {{ variable }}, які суть є полями форми. Система, окрім файлів шаблонів, зберігає інформацію про назви документів, що прив`язуються до шаблонів, а також інформацію про структуру полів документів.

Користувач, який хоче отримати готовий екземпляр документу, знає його назву. Отже, першим кроком від робить пошуковий запит до системи, щоб знайти ідентифікатор необхідного шаблону. Тепер, коли відомий ідентифікатор, користувач робить другий запит, що узнати структуру питомого шаблону, а саме кількість, типи даних та назви його полів. Після цього користувач заповнює форму даними і робить третій запит – на збірку. Він передає інформаційній системі введені дані, а вона, в свою

чергу, пов`язує їх із шаблонами, замінюючи мітки-змінні дійсними даними. Далі система формує екземпляр документу і повертає його користувачеві. На цьому робочій процес закінчується.

2.2Потоки даних інформаційної системи

Виходячи із вказаної вище схеми, ми можемо зробити висновок, що основною сутністю інформаційної системи є шаблон документу.

На основі цієї сутності ми можемо виділити два основних потоки даних в системі: створення та збірка. Також виділимо дві основі групи осіб, що приймають участь у роботі інформаційної системи: користувач та адміністратор (мається на увазі не системний адміністратор, а той, хто керує інформаційними потоками).

Адміністратор створює шаблон документу, позначає в ньому змінні дані та їхні атрибути та реєструє шаблон в системі.

Користувач може виконувати пошук необхідного шаблону, вносити в нього змінні дані та створювати запити на збірку документу. Таким чином, основі процеси системи – це керування шаблонами, та їхнє використання. Відобразимо наведені процеси на DFD-діаграмі 1-го рівня на рисунку 2.1.

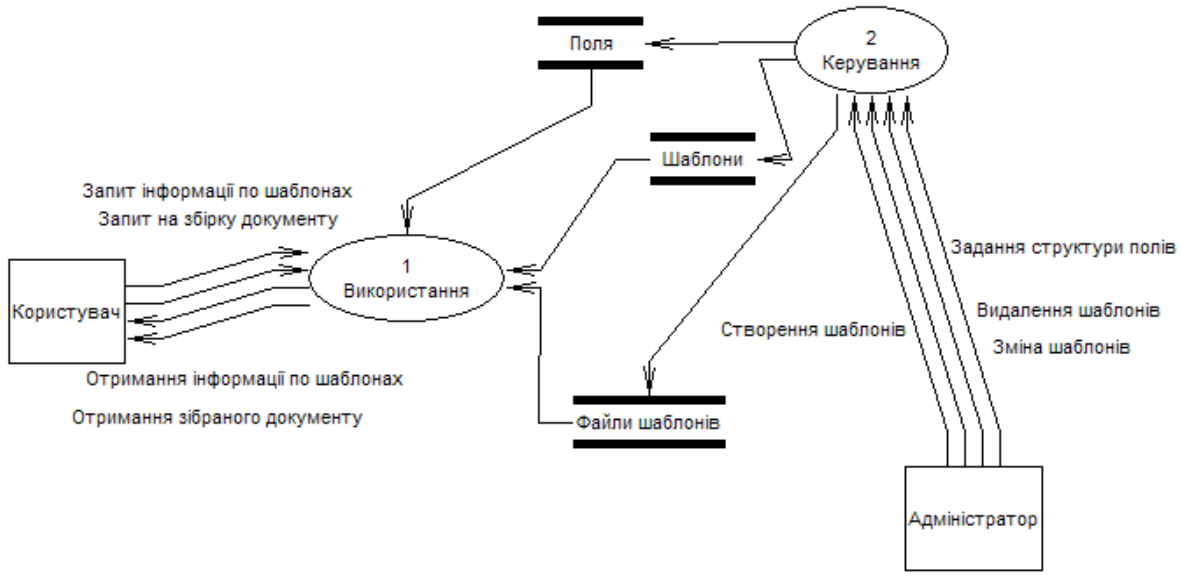

Рисунок 2.1 – DFD-діаграма 1-го рівня

2.3 Сховища даних інформаційної системи

На основі отриманих вище потоків даних виділимо 3 сховища даних: «Поля», «Шаблони», «Файли шаблонів». Нижче наведемо атрибути цих сховищ.

Шаблони (form)	
Назва	Пояснення
id	Унікальний ідентифікатор шаблону
name	Назва документу шаблону
filename	Назва файла, в якому зберігається шаблон

Таблиця 2.1 - Атрибути сховища «Шаблони»

Файли шаблонів – фізично є простим каталогом на диску	
Назва	Пояснення
filename	Назва файла, в якому зберігається шаблон

Таблиця 2.2 - Атрибути сховища «Файли шаблонів»

Поля(field)	
Назва	Пояснення
id	Унікальний ідентифікатор поля
Fop., rm_id	Ідентифікатор шаблону, до якого відноситься поле
name	Змістова назва поля
Hint	Коментарій або підказка для користувача
Field_type	Тип поля. Моживі варіанти: текст (text), число(number) та дата(date)
V ar_name	Назва змінної у файлі шаблону, яка буде замінена на значення цього поля
Is_required	Чи є поле обов`язковим

Таблиця 2.3 - Атрибути сховища «Шаблони»

Відобразимо наведені сховища, їхні атрибути та зв`язки між ними на ER-діаграмі.

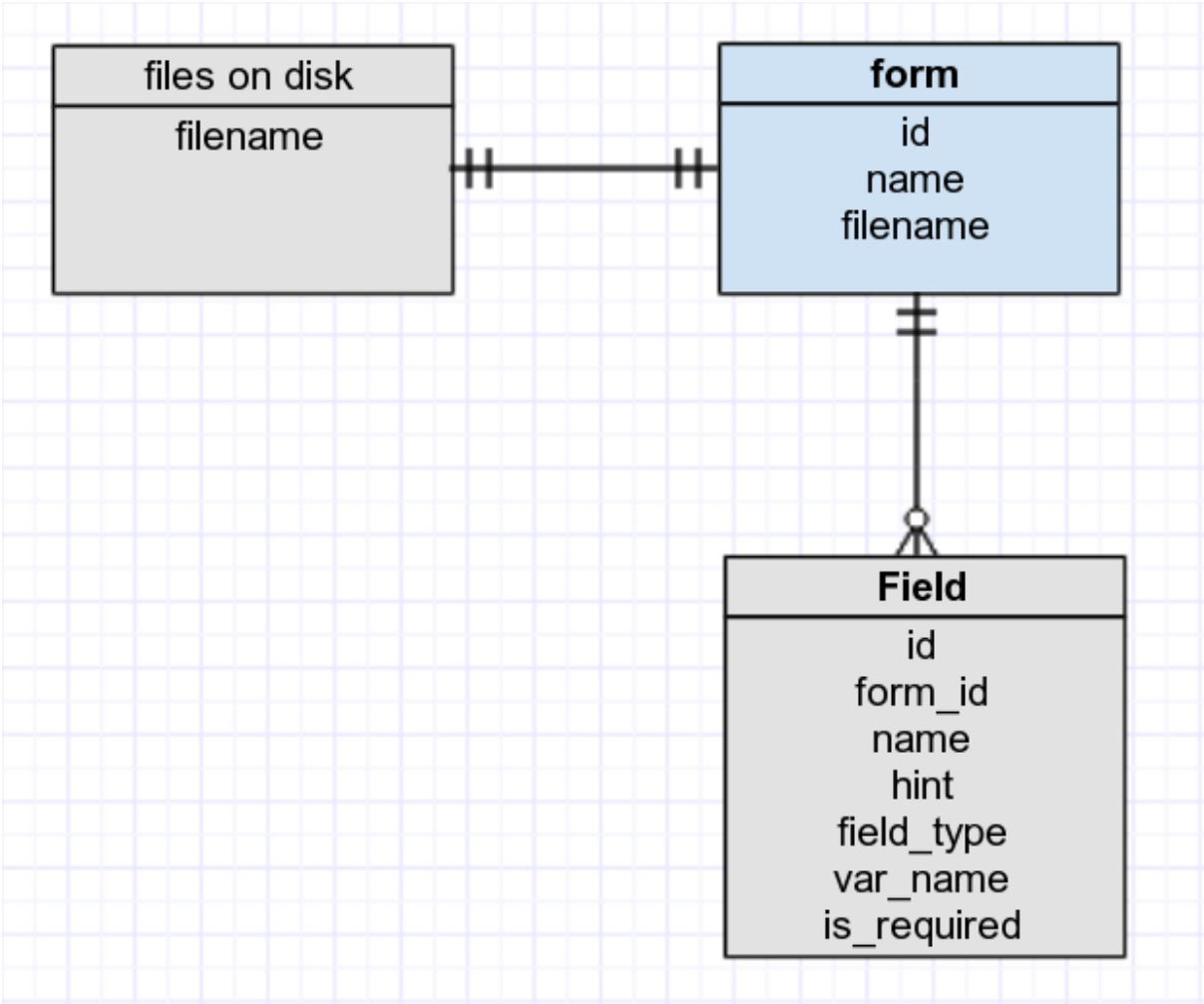

Рисунок 2.2 – ER-діаграма

2.4 Проектування бази даних

На основі отриманих вище потоків даних виділимо 3 сховища даних: «Поля», «Шаблони», «Файли шаблонів». Нижче наведемо атрибути цих сховищ.

На основі виявлених сховищ даних створимо структуру зберігання цих даних в СУБД, і яка ляже в основу фізичної реалізації.

Form					
No	Ключ	Назва	Тип даних	Null	Unique
1	PK	Id	Integer	-	+
2		Name	V archar(512)	-	-
3		fileName	V archar(64)	-	-

Таблиця 2.4 – Структура таблиці Form

Field					
No	Ключ	Назва	Тип даних	Null	Unique
1	PK	Id	Integer	-	+
2		Fop., rm_id	Integer	-	-
3	FK -> form.id	Name	Varchar(128)	-	-
4		Hint	Varchar(256)	+	-
5		Field_type	Varchar(16)	-	-
6		V ar_name	Varchar(64)	-	-
7		Is_required	boolean	-	-

Таблиця 2.5 – Структура таблиці Field

Оскільки файли шаблонів будуть зберігатися не в СУБД, то таблиця для цього сховища не створюється.

2.5 Організація інформаційної системи в мережі

Інформаційно-програмний виріб організований по архітектурі «клієнт - сервер» В цьому випадку мережеве програмне забезпечення припускає не лише спільне використання ресурсів мережі, але і обробку на сервері по запитах користувачів. Програмне забезпечення в даному випадку складається з двох частин: сервера і клієнта. Програма клієнт виконується на локальному комп'ютері користувача, вона посилає запити програмі-серверу і приймає від неї необхідну інформацію. Програма-сервер працює на комп'ютері загального доступу, здійснює обробку запитів, що поступають до неї, і повертає клієнтові необхідні результати.

В термінології веб-розробки серверну частину інформаційної системи називаються бекендом, а клієнтську, яка виконується в браузері користувача – фронтендом. Бекенд виконує задачі зберігання, обробки та видачі даних за запитом. Фронтенд організує зручний доступ даних для користувача.

В нашому випадку бекенд виконує такі функції:

1. Видача користувачеві файлів фронтенду для відображення в браузері;
2. Отримання запитів на надання інформації про шаблони та їхні структури;
3. Отримання запитів на створення екземпляру документу, разом зі змінними даними, а також видача цих екземплярів

Функція #1 являє собою видачу статичного контенту і реалізується засобами веб-серверу. Функція #2 реалізується через витяг відомостей із бази даних, формування їх у форматі JSON і передачі результату фронтенду. Функція #3 реалізується через отримання даних та передачі їх особливому компоненту системи – шаблонізатору. Він, в свою чергу, виконує основне призначення ІС та повертає питомий результат користувачу.

Для ефективної організації робочого процесу була створена схема адрес, через які клієнт може отримати від бекенда тий чи інший ресурс. Бекенд реалізує принцип REST API, тобто можливість обміну чистими даними з клієнтом, минуючи HTML-обгортку. Розглянемо адреси доступу до ресурсів, які є актуальними в бекенді:

Метод HTTP	URI	Дія
GET	/	Видача клієнту файлу index.html, який є основним для фронтенд- компонента
GET	/static/*	Видача стилів та скриптів, необхідних для роботи фронтенда
GET	/api/search/<q>	Пошук усіх шаблонів, які містять рядок q. Видає клієнту JSON, в якому містяться ідентифікатори та повні назви шаблонів
GET	/api/forms/<id>	Видає JSON, в якому містяться відомості про поля шаблону q
POST	/api/forms/<id>/compile	Отримує список пар Ключ=Значення, де ключ назва змінної, значення – бажаний зміст, на який треба замінити цю змінну. За допомогою шаблонізатора проводить відповідну заміну у шаблоні id і результат надсилає клієнтові.

Таблиця 2.5 – Організація URI-ідентифікаторів

2.6 Проектування бекенду

Первинну обробку запитів виконує компонент під назвою Роутер. Він займається маршрутизацією запитів в залежності від вказаного URI, тобто ставить в залежність деякою URI деяку дію. Якщо клієнта на деякий статичний файл, то розтер просто віддає цей файл без змін. Якщо запит на деякий ресурс, який відноситься до API, то роутер виділяє із запиту ідентифікатор конкретного ресурса, звертається до відповідної моделі даних, а вона, в свою чергу, викликає методи ORM. ORM – компонент об`єктно-реляційного зв`язку. Він дає можливість пов`язати дані із СУБД із моделями даних, і там, і там зберігаючи усі зв`язки та обмеження. Також, опосередковано роутер пов`язується із шаблонізатором, задача якого показана вище.

Виходячи з вищевказаного побудуємо модель бекенду.

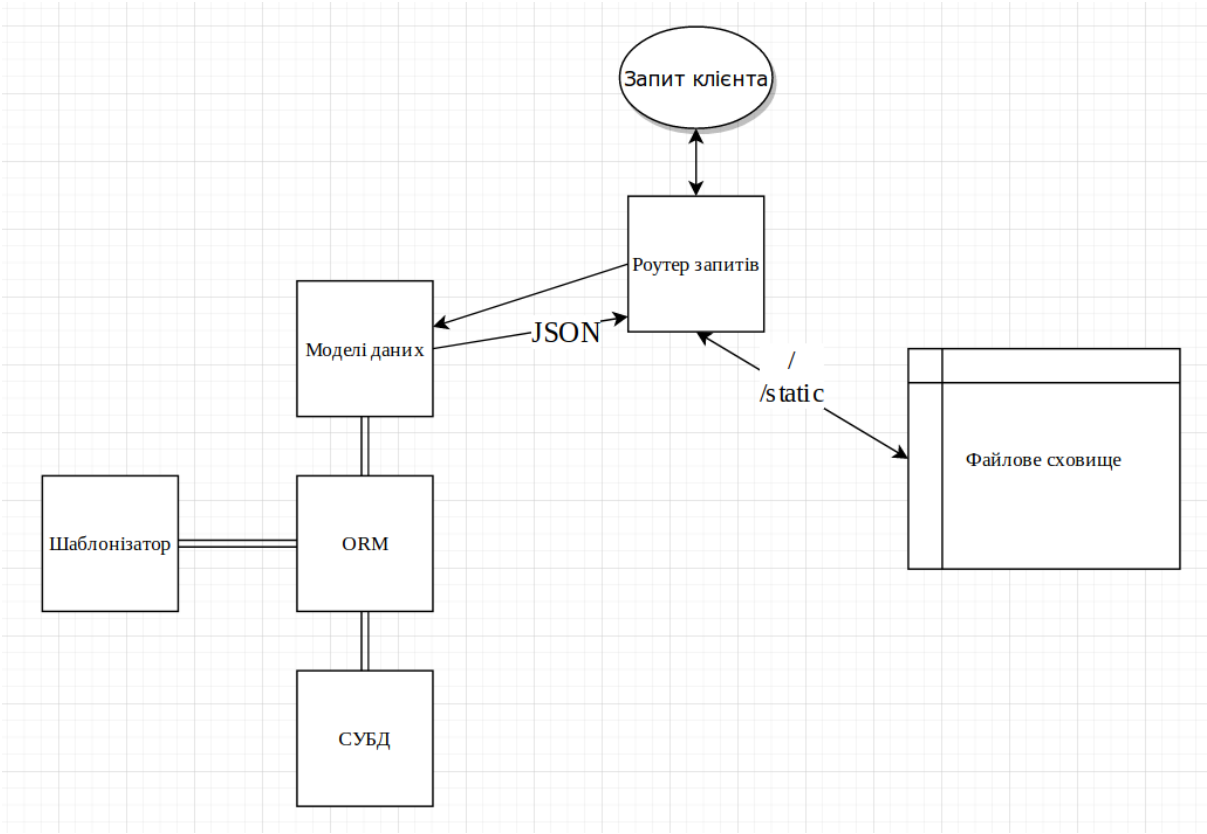

Рисунок 2.3 – Модель бекенду

2.7 Проектування фронтенду

Для фронтенду була обрана бібліотека React. React (старі назви: React.js, ReactJS) — відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту вебсторінки, з якими стикаються в розробці односторінкових застосунків. Розробляється Facebook, Instagram і спільнотою індивідуальних розробників.

React дозволяє розробникам створювати великі вебзастосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим. React обробляє тільки користувацький інтерфейс у застосунках. Це відповідає *видові* у шаблоні модель-вид-контролер (MVC), і може бути використане у поєднанні з іншими JavaScript бібліотеками або в великих фреймворках MVC, таких як AngularJS[6]. Він також може бути використаний з React на основі надбудов, щоб піклуватися про частини без користувацького інтерфейсу побудови вебзастосунків. Як бібліотеку інтерфейсу користувача React найчастіше використовують разом з іншими бібліотеками, такими як Redux.

Виходячи з постановки задачі, фронтенд буде мати наступні властивості. Він складається із однієї сторінки, на якій присутній рядок пошуку. Якщо користувач введе деяку назву, або частину назви, фронтенд негайно надішле запит бекендові на пошук. Якщо будуть знайдені будь-які шаблони, назви яких містять текст пошукового рядка, вони будуть негайно виведені під ним. Напроти кожної назви шаблону має буди відповідна кнопка редагування. Якщо її натиснути, відкриється модальне вікно, у якому будуть наявні поля вводу даних, що відповідають вказаному шаблону. Також, в цьому вікні мають виводитися підказки до полів, якщо такі наявні, а також кнопки для онулення введених даних та запиту на побудування документа. Після натискання на останню фронтенд

відправить запит до бекенда і отримає файл екземпляра документа, який буде запропоновано завантажити.

Розглянемо принципову схему головного вікна фронтенда.

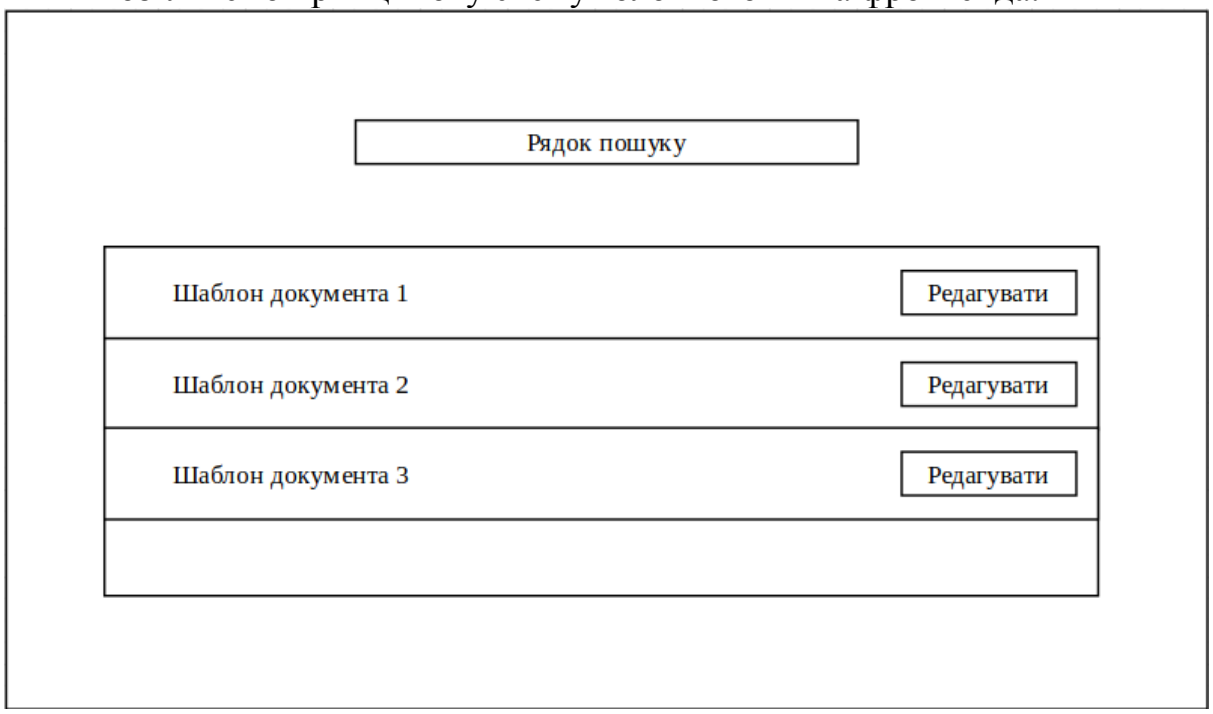

Рисунок 2.4 – Принципова схема головного вікна

Коли користувач ініціює пошук, він має побачити таке:

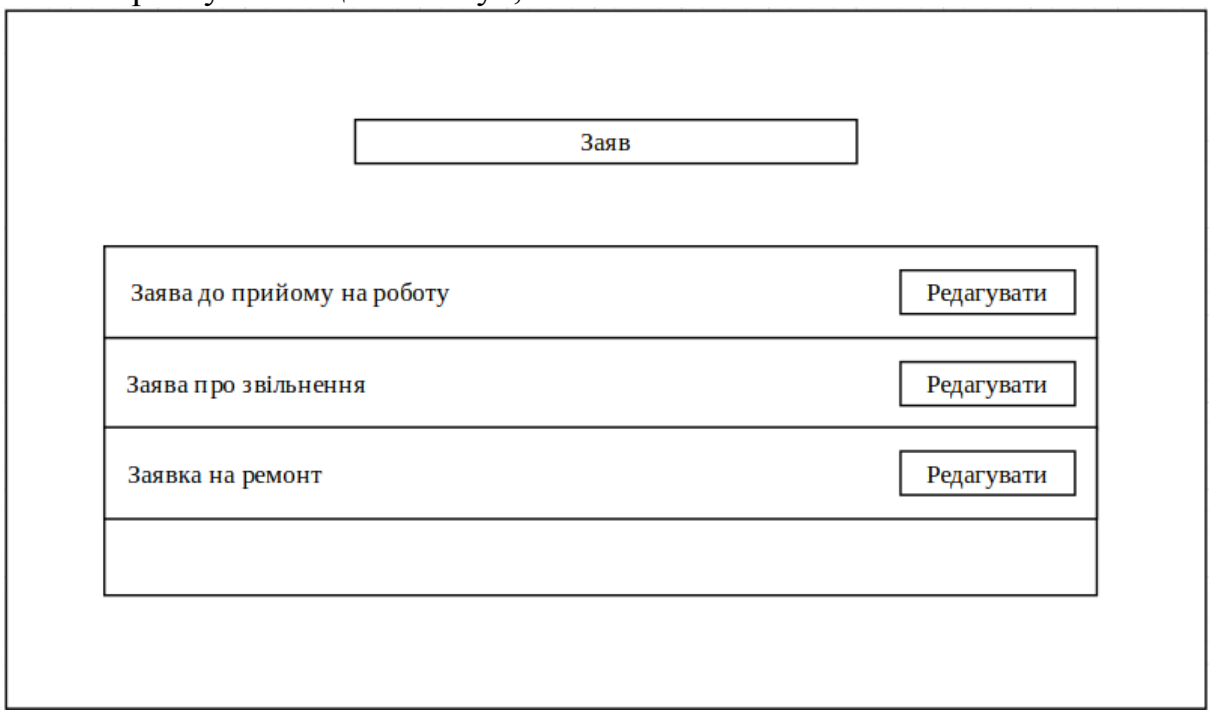

Рисунок 2.5 – Варіант вигляду головного вікна

Бачимо, що при запиті «Заяв», буде отримано усі шаблони, назви яких починаються на «Заяв».

РОЗДІЛ 3
Реалізація інформаційної системи

3.1 Обґрунтування засобів розробки

Для реалізації програмного продукту був обраний ряд ПО і технологій. При їх виборі були використані такі принципи:

- ПЗ має бути безкоштовним
- Незалежність від платформи
- ПЗ повинно дозволяти виконувати налагодження в домашніх умовах, тобто без необхідності щоразу вносити зміни в проект безпосередньо на web-сервері

- Популярність технологій, і як результат – легкість знаходження інформації по ним

З урахуванням наведених принципів був зроблений вибір. Перелічимо основні фреймворки та бібліотеки, для бекенду – це мова Python, flask, SQLAlchemy, Alembic, docxtpl, SQLite. Для фронтенду – React.js, bootstrap, axios, react-hook-form. Javascript для фронтенду є єдиною мовою. Хоча можливе і використання строгої типізації завдяки Typescript або Flow.

Python (найчастіше вживане прочитання — «**Па́йтон**», запозичено назву з британського шоу Монті Пайтон) — інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднування наявних компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується кілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.

Flask — мікрофреймворк для веб-додатків [7], створений з використанням Pythop. Його основу складає інструментарій Werkzeug та рушій шаблонів Jinja2. Flask називається мікрофреймворком, оскільки він не вимагає спеціальних засобів чи бібліотек. У ньому відсутній рівень абстракції для роботи з базою даних, перевірки форм або інші компоненти, які надають широковживані функції за допомогою сторонніх

бібліотек. Однак, Flask має підтримку розширень, які забезпечують додаткові властивості таким чином, наче вони були доступні у Flask із самого початку. Існують розширення для встановлення об'єктно-реляційних зв'язків, перевірки форм, контролю процесу завантаження, підтримки різноманітних відкритих технологій аутентифікації та декількох поширених засобів для фреймворку.

Переваги Flask:

- Містить сервер для розробки та відлагоджувач;
- Управління запитами RESTful;
- Використовує шаблони Jinja2;
- 100% відповідністьWSGI1.0;
- Підтримка Unicode;
- Гарно розроблена документація
- Наявність розширень для забезпечення бажаної поведінки

SQLAlchemy - це програмне забезпечення з відкритим вихідним кодом для роботи з базами даних за допомогою мови SQL[8]. Воно реалізує технологію програмування ORM (Object-Relatiop., nal Mapping), яка пов'язує бази даних з концепціями об'єктно-орієнтованих мов програмування. SQLAlchemy дозволяє описувати структури баз даних і способи взаємодії з ними прямо на мові Pythop., n. Найбільш видатною особливістю SQLAlchemy є можливість генерації SQL-коду на основі об`єктних моделей даних, заданих у проекті.

Alembic - це інструмент для міграції бази даних, який використовується в SQLAlchemy [9]. Міграція бази даних - це щось схоже на систему контролю версій для баз даних. При розробці програми

поширена практика зміни схеми таблиці. Тут і приходить на допомогу Alembic. Він, як і інші подібні інструменти, дозволяє змінювати схему бази даних при розвитку програми. Він також стежить за змінами самої бази, так що можна рухатися туди і назад. Якщо не використовувати Alembic, то за всіма змінами доведеться стежити вручну і міняти схему за допомогою Alter. В поточному проекті використовуються розширення flask-migrate для інтеграції Alembic у Flask.

docxtpl – пакет для Python, який дозволяє використовувати jinja2-розмітку у документах docx [10].

React.js — JavaScript-бібліотека для створення інтерфейсів користувача

Переваги бібліотеки React.js:

- Бібліотека досить проста і функціональна. Для того щоб озібратися в ній, потрібен мінімальний багаж знань.
- Вимоги до стека відсутні, тому React.JS можна використовувати на будь-якому проекті.
- Легковісна.
- Досить висока швидкість розробки. Завдяки використанню будь-яких шаблонів і доступності документації, більшість виникаючих проблем вирішуються досить швидко.

React.js дозволяє створювати функціональні додатки, які відповідають усім сучасним стандартам, при мінімальному підключенні нових ресурсів і, власне, дешевше.

Axios – HTTP-клієнт, розроблений на JavaScript для браузерів та Node.js.

Bootstrap — це безкоштовний набір інструментів з відкритим кодом, призначений для створення веб-сайтів та веб-додатків, який містить шаблони CSS та HTML для типографіки, форм, кнопок, навігації та інших

компонентів інтерфейсу, а також додаткові розширення JavaScript. Він спрощує розробку динамічних веб-сайтів і веб-додатків [12].

Bootstrap — це клієнтський фреймворк, тобто інтерфейс для користувача, на відміну від коду серверної сторони, який знаходиться на сервері. Переваги Bootstrap[13]:

- Зменшення кількості часу, що витрачається на розробку
- Адаптивність
- Кросбраузерність
- Легкість у використанні і швидкість в освоєнні
- Зрозумілий код
- Єдність стилів

SQLite — полегшена реляційна система керування базами даних. Втілена у вигляді бібліотеки, де реалізовано багато зі стандарту SQL-92c[14]. Сирцевий код SQLite поширюється як суспільне надбання, тобто може використовуватися без обмежень та безоплатно з будь-якою метою. Особливістю SQLite є те, що вона не використовує парадигму клієнт-сервер, тобто рушій SQLite не є окремим процесом, з яким взаємодіє застосунок, а надає бібліотеку, з якою програма компілюється і рушій стає складовою частиною програми. Таким чином, як протокол обміну використовуються виклики функцій (API) бібліотеки SQLite. Такий підхід зменшує накладні витрати, час відгуку і спрощує програму. SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси і дані) в єдиному стандартному файлі на тому комп'ютері, на якому виконується застосунок.

3.2 Опис файлової структури та класів

В ході розробки була створена наступна структура проекту:
app/ — основна директорія проекту, яка містить реалізацію
api/ — директорія, що містить класи, які відповідають за маршрутизацію API-запитів
forms.py — функції, які описують, як обробляти запити до «/api»
models.py — моделі даних, тобто класи, що описують сутності шаблонів та полів, і доступ до них
routes.py — функції, які описують, як обробляти запит до «/»
static/ — статичні файли проекту. В нашому випадку – все, що відноситься до фронтенду
index.html — головна сторінка фронтенду;
libs — директорія з усіма необхідними бібліотеками та стилями для фронтенду;
main.js – головний скрипт фронтенду, створює і виконує екземпляр react.js
app.db – файл бази даних SQLite
config.py – конфігураційний файл проекту
forms/ - директорія, що містить файли шаблонів
forms-crud.py – головний файл проекту, який запускає WSGI-сервер forms-
env/ — середовище Python, що містить усі необхідні для роботи бібліотеки
migrations/ — репозиторій змін у базі даних, керується Alembic

Після запуску сервера із сторони бекенда, завантажуються фреймворк Flask, який, в свою чергу, завантажує fop., rms-crud.py. Цей файл відвантажує класи та ресурси і сервер починає слухати HTTP-запити. Коли такий запит поступає, Flask визначає URI і в залежності від нього запускає

відповідний обробник. Обробники запитів визначені у файлах routes.py та forms.py. Розглянемо деякі обробники:

```
#routes.py
@app.route('/')
@app.route('/index')
    def index():
        return app.send_static_file('index.html')
```

Якщо клієнт дасть запит до «/», то буде виконана функція index(), яка на запит дасть відповідь у вигляді файлу index.html із директорії static (директорія за замовченням для всіх статичних файлів у Flask).

```
#forms.py
@bp.route('/forms/<int:id>', methods=['GET'])
    def get_form(id):
        d = {}
        fields = Field.query.filter_by(form_id=id).all()
        for f in fields:
            d[f.id] = f.repr_schema()
        return jsonify({"fields":d})
```

Якщо буде запит за URI “/forms/<id>”, то Flask вилучить із запиту значення <id>, та запустить функцію get_form(id), передавши їй цей параметр. Ця функція за допомогою ORM-методів у моделі Field вилучає із бази даних усі записи, де form_id=id. Відразу ж формуються об`єкти типу Field, для яких вилучаються значення repr_schema(). Ці значення вносяться у словник d, який потім перетворюється на JSON-відповідь і відправляється клієнтові.

Розглянемо реалізацію моделі Field:

```
class Field(db.Model):


    id = db.Column(db.Integer, primary_key=True, nullable=False,

unique=True)

    form_id = db.Column(db.Integer, db.ForeignKey('form.id'), nullable=False)
    name = db.Column(db.String(128), index=True, nullable=False)

    hint = db.Column(db.String(256))
    field_type = db.Column(db.String(16), nullable=False)

    var_name = db.Column(db.String(64), nullable=False)

    is_required = db.Column(db.Boolean, nullable=False)

    def repr_schema(self):

        schema = {

            "type": "input",

            "inputType": self.field_type,

            "label": self.name,
            "model": self.var_name,

            "inputName": self.var_name,

            "hint": self.hint,
            "required": self.is_required

        }
        return schema

    def __repr__(self):


        return '<Field {0}, {1}, {2}>'.format(self.form, self.name, self.field_type) \
```

Бачимо, що змінні класу задаються у форматі SQLAlchemy. Також, досліджуючи цей клас, Alembic може сформувати відповідну таблицю. Метод repr_schema, як було вище вказано, необхідний для формування JSON- відповіді. Такий формат словнику необхідний для подальшої інтерпретації JSON модулем react-hook-form, який створить форму з боку фронтенду для вводу даних користувачем.

Формування екземпляру документу проводиться у наступній функції:

```
@bp.route('/forms/<int:id>/compile', methods=['POST'])
def compile_form(id):
    form = Form.query.filter_by(id=id).first_or_404()
    filename = os.path.join(app.config['FORMS_DIR_PATH'], form.filename)

    doc = DocxTemplate(filename)
    doc.render(request.args.to_dict())

    stream = BytesIO()

    doc.get_docx().save(stream)

    stream.seek(0)

    return send_file(stream, as_attachment=True, attachment_filename='result.docx',
    mimetype='docx')
```

Спершу, метод по id шаблону знаходить файл, у якому той міститься. Потім, файл відкривається, із запиту вилучаються дані форми, які підставляються у шаблон. Отриманий документ зберігається у пам`яті, після чого він віправляється клієнтові.

Всі основі вихідні файли наводяться у додатку А.

3.3 Інструкція користувача

Щоб працювати із системаою користувач має зайти на відповідний веб-сайт, на головну сторінку. Він побачить наступне вікно

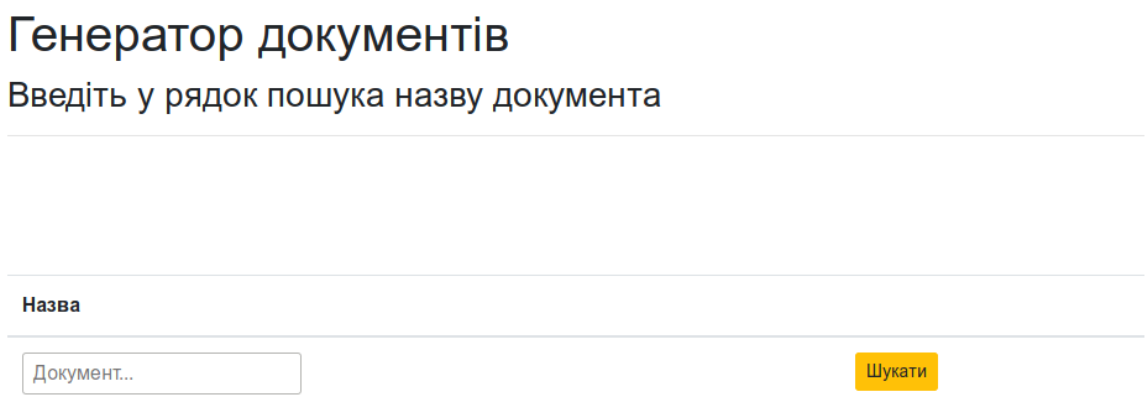

Рисунок 3.1 – Вигляд головного вікна додатку

Після цього користувач може у рядок пошуку ввести назву документа, або її частину. Якщо у користувача є проблеми із з`єднанням, він може натиснути кнопку «Шукати». На наступному скриншоті бачимо, що користувач ввів «Зая» і було знайдено документ «Заява про відгул».

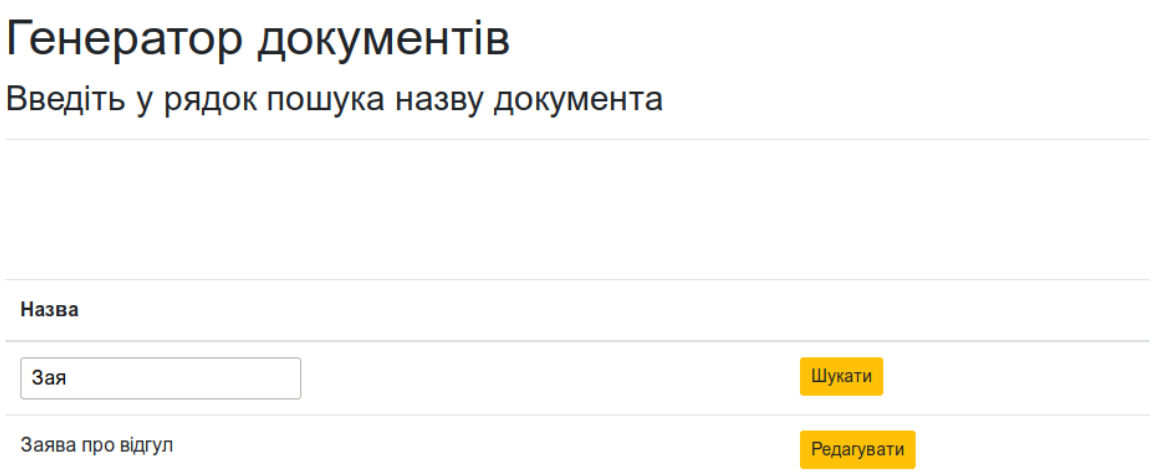

Рисунок 3.2 – Результат пошуку у головному вікні

Далі користувач має вибрати документ, який він хоче отримати і натиснути відповідну йому кнопку «Редагувати». З`явиться модальне вікно редагування форми. Далі користувач має вибрати документ, який він хоче отримати і натиснути відповідну йому кнопку «Редагувати». Далі користувач має вибрати документ, який він хоче отримати і натиснути відповідну йому кнопку «Редагувати». З’явиться модальне вікно редагування форми

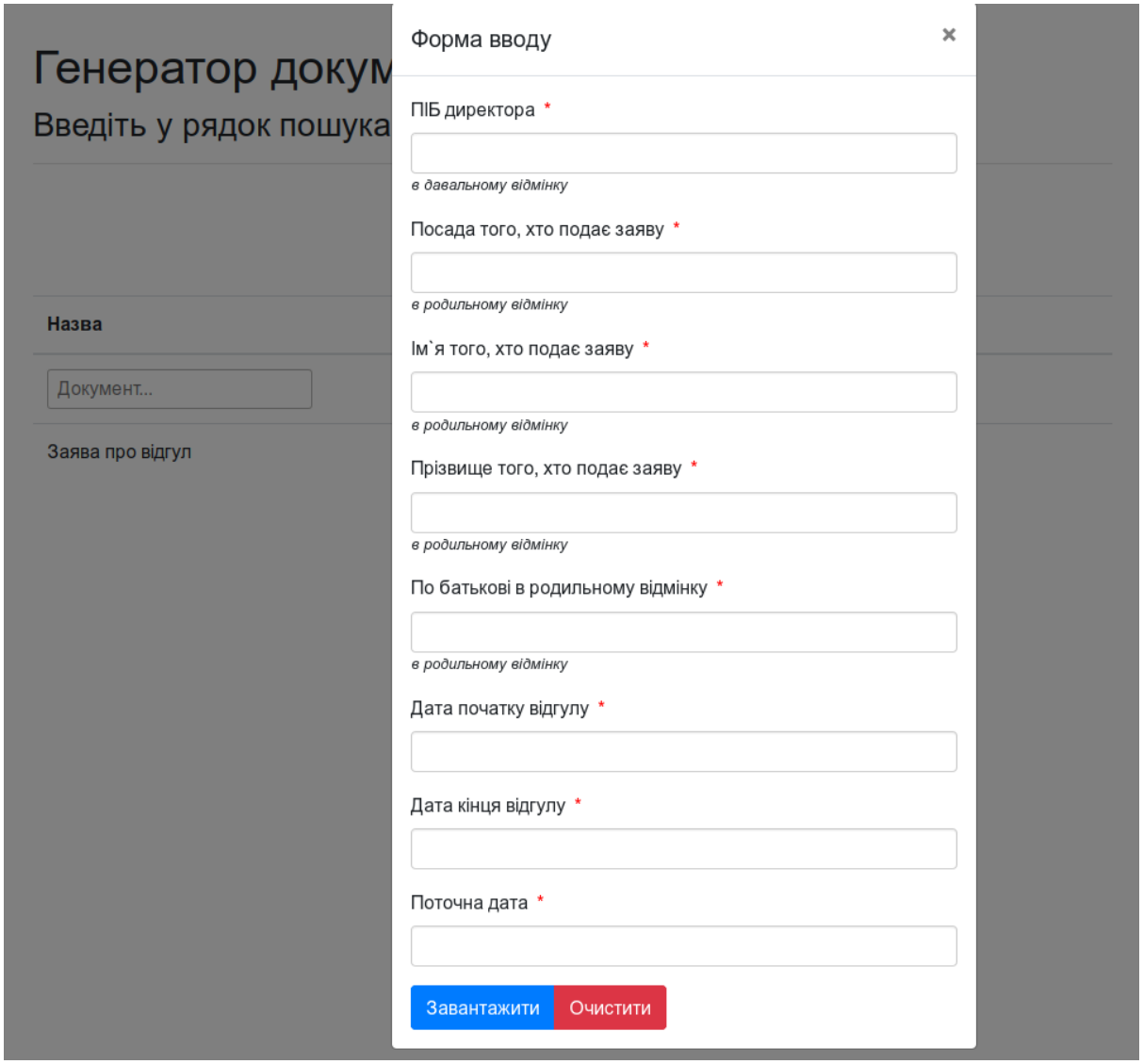

Рисунок 3.3 – Форма пошуку

Далі необхідно ввести бажані дані.

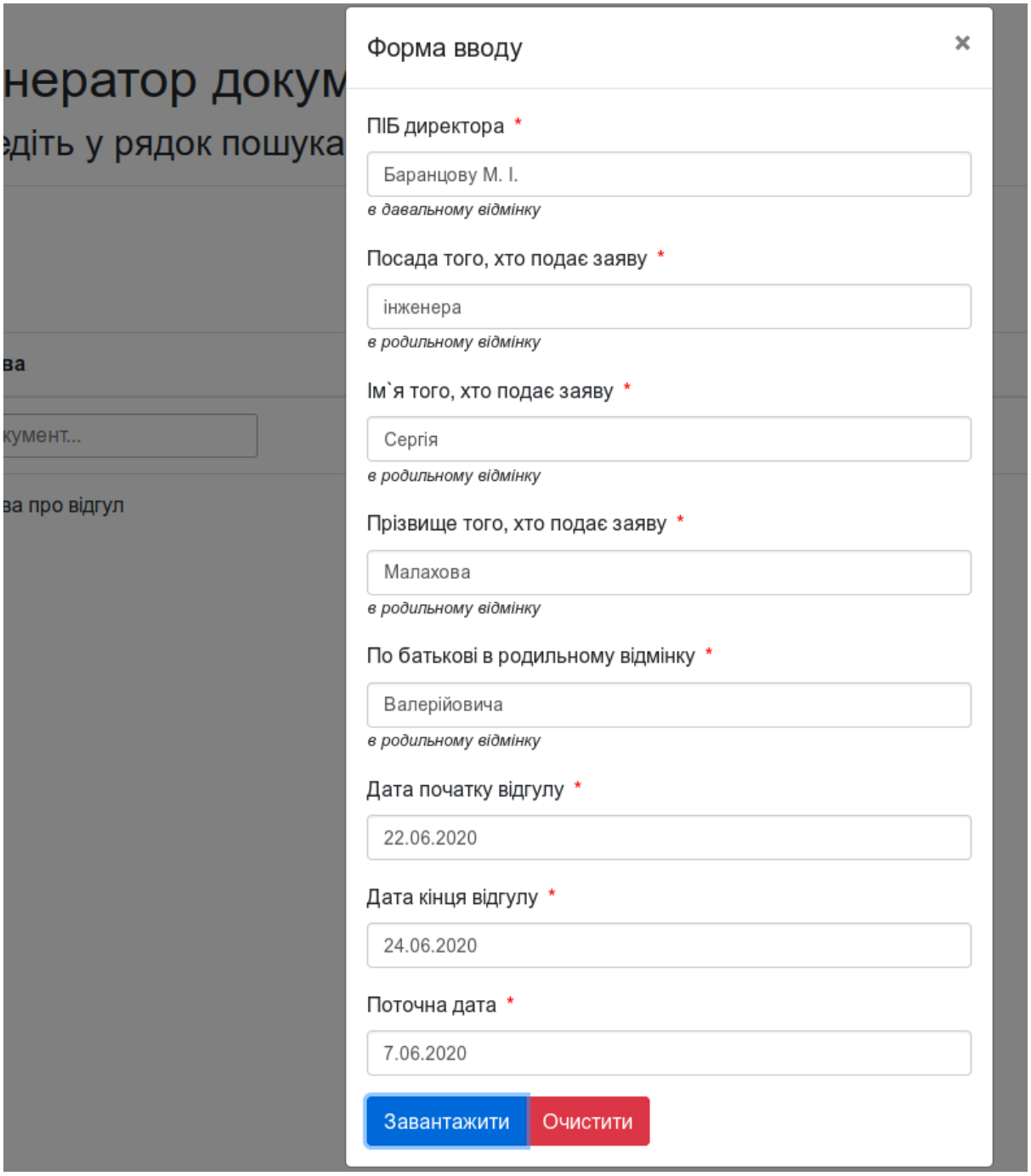

Рисунок 3.4 – Заповнена форма пошуку

Якщо треба очисти поля вводу, треба натиснути кнопку «Очистити».

Після вводу треба натиснути кнопку «Завантажити». Користувачеві буде запропоновано скачати файл.

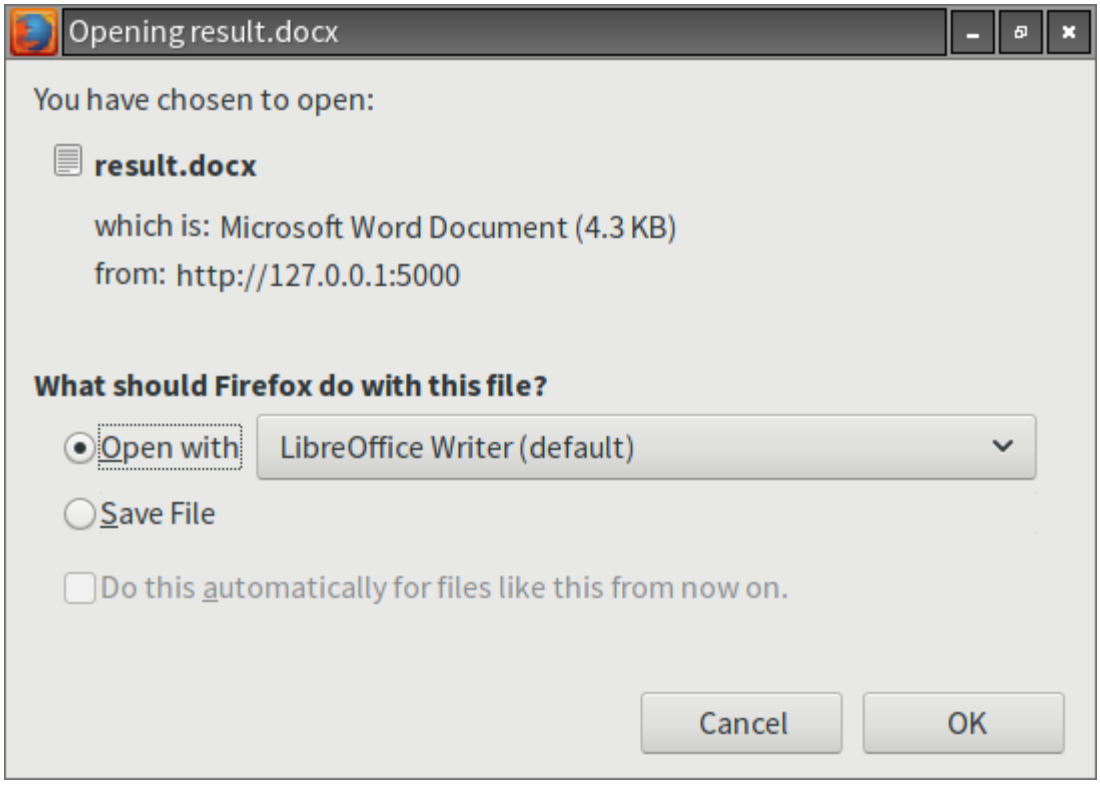

Рисунок 3.5 – Завантаження готового екземпляру документа

Приклади шаблону і результату підготовки документа наведені у додатку Б.

3.4 Інструкція адміністратора

Для роботи додатку необхідно наступне ПЗ:

Ubuntu Linux >=18.04

Пакети python3, python3-venv, python3-dev, nginx

Адміністратор має створити окремого користувача в системі для роботи додатку. В домашній каталог треба розмістити дистрибутив додатка forms-crud.

Далі, треба виконати наступні команди:

```
$ cd docx-template-form
$ python3 -m venv venv
$ source venv/bin/activate
(venv) $ pip install -r requirements.txt
```

У файлі config.py треба налаштувати наступні параметри:
SQLALCHEMY_DATABASE_URI – розташування SQLite бази даних;
FORMS_DIR_PATH – розташування файлів шаблонів.
Далі адміністратор має створити конфігурацію nginx, враховуючи усі шляхи установки. Приклад конфігурації наданий у додатку В.
Для додання нових шаблонів адміністратор має розмістити самі шаблони у каталогу FORMS_DIR_PATH. Для створення форми вводу він має будь-яким SQLite-редактором відкрити файл бази даних SQLALCHEMY_DATABASE_URI і, користуючись структурою БД із попередніх розділів, внести форми за наступним прикладом:

```
insert into form values (1, 'Заява про відгул', 'test1.docx')

insert into field values (7, 1, 'ПІБ директора', 'в давальному відмінку', 'text', 'dir_nsp', 1)

insert into field values (8, 1, 'Посада того, хто подає заяву', 'в родильному відмінку', 'text', 'position', 1)

insert into field values (9, 1, 'Ім`я того, хто подає заяву', 'в родильному відмінку', 'text', 'name', 1)

insert into field values (10, 1, 'Прізвище того, хто подає заяву', 'в родильному відмінку', 'text', 'surname', 1)

insert into field values (11, 1, 'По батькові в родильному відмінку', 'в родильному відмінку', 'text', 'patronymic', 1)

insert into field values (12, 1, 'Дата початку відгулу', '', 'date', 'date_start', 1)

insert into field values (13, 1, 'Дата кінця відгулу', '', 'date', 'date_end', 1)

insert into field values (14, 1, 'Поточна дата', '', 'date', 'date_current', 1)
```

Запуск додатка проводиться за допомогою команди flask run.

ВИСНОВКИ

Генератор документів на основі шаблонів – це веб-сервіс, який допомогає створювати неосвіченим користувачам добре сформатовані документи. Це досягається за рахунок того, что користувач не оформлює весь документ цілком – замість цього він може використати розроблену в цьому проекті інформаційну систему. В такому випадку користувачеві буде досить ввести тільки змінні дані документу, які будуть підставлені в заздалегідь відформатований шаблон.

В рамках даного дипломного проекту була розроблена така система, при цього був використаний ряд сучасних веб-технологій. А саме: бекенд-фреймворк Flask та фронтенд фрейворки Vue.js та Bop., op., tstrap. Була спроектована база даних, та реалізована за допомогою фреймворка SQLAlchemy та СУБД SQLite.

Позитивними якостями розробленого веб-сервісу є:

1. Зручний інтерфейс користувача;
2. Відсутність реєстрації, що забезпечує анонімність та економить час користувачів.
3. Можливість пошуку файлів

Негативними якостями Інтернет-магазину є:

1. Відсутність панелі адміністратора;
2. Неможливість використання складних виразів у шаблонах, замість простих змінних.

Однак, слід враховувати, що для реалізації обох цих пунктів треба реалізувати автоматичне вилучення виразів із файлу шаблону і використання більш продвинутих технік розробки. При цьому всьому розроблена система здатна покрити потреби невеликого підприємства для обслуговування електронного документообертy невеликих масштабів.

Враховуючи вище сказане можна зробити висновок, що всі поставленні завдання в роботі виконані в повному обсязі.

СПИСОК ЛІТЕРАТУРИ

1. https://officelegko.com/2017/09/26/shablonyi-v-microsoft-word/
2. https://guidepc.ru/applications/microsoft-word/kak-sozdat-shablon-v-microsoft-word/
3. https://uk.wikipedia.op., rg/wiki/Flask
4. https://ru.wikibooks.org/wiki/SQLAlchemy
5. https://uk.wikipedia. org/wiki/SQLite
6. https://reactjs.org/docs/getting-started.html

ДОДАТОК А

Лістинг файлу models.py.

```
from app import db

class Form(db.Model):

     id = db.Column(db.Integer, primary_key=True,
     nullable=False, unique=True)
     #version_id = db.Column(db.Integer, nullable=False,
     default=1)

     name = db.Column(db.String(512), index=True,
     nullable=False)

     filename = db.Column(db.String(64), index=True,
nullable=False, unique=True)

     fields = db.relationship('Field', backref='form',
lazy='dynamic')

     def __repr__(self):

          return '<Form {}>'.format(self.name)
```

Лістинг файлу routes.py

```
from app import app
import os
from io import BytesIO
from flask import request, jsonify, send_file
from api.models import Form, Field
from docxtpl import DocxTemplate

@app.route('/')
@app.route('/index')
def index():
  return app.send_static_file('index.html')

@app.route('/search/<string:q>', methods=['GET'])
def search_form(q):
  l = []
  search = "%{}%".format(q)
  forms = Form.query.filter(Form.name.like(search)).all()
  for f in forms:
    d = {}
    d['id'] = f.id
    d['name'] = f.name
```

```
    l.append(d)
  return jsonify(l)

@app.route('/forms/<int:id>', methods=['GET'])
def get_form(id):
  d = []
  fields = Field.query.filter_by(form_id=id).all()
  for f in fields:
    d.append(f.repr_schema())
  return jsonify({"fields":d})

@app.route('/forms/<int:id>/compile',
methods=['GET','POST'])
def compile_form(id):
  form = Form.query.filter_by(id=id).first_or_404()
  filename = os.path.join(app.config['FORMS_DIR_PATH'],
form.filename)
  doc = DocxTemplate(filename)
  doc.render(request.args.to_dict())
  stream = BytesIO()
  doc.get_docx().save(stream)
  stream.seek(0)
  return send_file(stream, as_attachment=True,
attachment_filename='result.docx', mimetype='docx')
```