

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ

«СИСТЕМА ОБЛІКУ ДОГОВОРІВ»

SOFTWARE IMPLEMENTATION OF THE CONTRACT RECORD

KEEPING SYSTEM WEB APPLICATION

Виконав: студент 4 курсу, групи КНТ-210сп

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерні науки

ГАРВОНА Д.С.

(ПРИЗВИЩЕ та ініціали)

Керівник ГЛАДКОВА О.М.

(ПРИЗВИЩЕ та ініціали)

Рецензент ЗЕЛЕНЬОВА І.Я.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Комп'ютерні науки
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
« » 2023 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ГАРВОНИ Данила Сергійовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Програмна реалізація вебзастосунку «Система обліку договорів». Software Implementation of the Contract Record Keeping System Web Application

керівник проєкту (роботи) к.т.н., ГЛАДКОВА Ольга Миколаївна,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від «04» квітня 2023 року № 87

2. Строк подання студентом проєкту (роботи) 01 червня 2023 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. Матеріали і методи. 3. Основні рішення щодо реалізації програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

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

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ГЛАДКОВА О.М., доцент		
Нормоконтроль	ДЕЙНЕГА Л.Ю., ст.викладач		

7. Дата видачі завдання «04» квітня 2023 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.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка структури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділи 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студент(ка)

_____ Данило ГАРВОНА
(підпис) (Імя ПРИЗВИЩЕ)

Керівник проєкту (роботи)

_____ Ольга ГЛАДКОВА
(підпис) (Імя ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
75 с., 30 рис., 3 таб., 3 дод., 16 джерел.

АВТОМАТИЗОВАНА СИСТЕМА, ДОГОВІР, ДОКУМЕНТ,
КОНТРАГЕНТ, ВЕБЗАСТОСУНОК.

Об'єкт дослідження – процес організації документообігу.

Предмет дослідження – програмні засоби для організації документообігу.

Метою роботи є програмна реалізація системи документообігу у вигляді вебзастосунку.

Матеріали, методи та технічні засоби: мова програмування Python, Framework Django, система керування базами даних MySQL.

Результати. Розроблено систему документообігу, яка дозволяє користувачу здійснити облік договорів на підприємстві.

Практична цінність роботи полягає у програмній реалізації системи документообігу у вигляді вебзастосунку. Система показує всі договори, які існують. Надає можливість створювати нові та змінювати вже існуючі. Також, додавання к договору документу Word та отримання різних версій змін цього документу.

Висновки. Реалізовано систему документообігу у вигляді вебзастосунку.

Галузь використання – користувачі підприємств.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 75 pages, 30 figures, 3 tables, 3 appendixes, 16 sources.

AUTOMATED SYSTEM, CONTRACT, DOCUMENT, CONTRACTOR, WEB APPLICATION.

The object of the study is the process of document flow implementation.

The subject of the study is software tools for the implementation of document flow.

The purpose of the work is the software implementation of the document flow system in the form of a web application.

Materials, methods and technical means: Python programming language, Django Framework, MySQL database management system.

The results. A document management system has been developed that allows the user to record contracts at the enterprise.

The practical value of the work lies in the software implementation of the document management system in the form of a web application. The system shows all contracts that exist. Provides an opportunity to create new ones and modify existing ones. Also, adding a Word document to the contract and receiving different versions of changes to this document.

Conclusions. Implemented an document management system in the form of a cross-platform web application.

Field of use – enterprise users.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ.....	9
1 Аналіз предметної області	11
1.1 Аналіз важливості галузі системи обліку договорів	11
1.2 Практичність та зручність використання застосунків для системи обліку договорів	13
1.3 Порівняння існуючих аналогів.....	15
1.3.1 Застосунок «OpenDocMan»	16
1.3.2 Сервіс «Alfresco Content Services».....	17
1.3.3 Сервіс «Kimios»	18
1.3.4 Порівняння аналогів.....	19
1.4 Постановка завдання	20
1.4.1 Функціональність системи	20
1.4.2 Вимоги до інтерфейсу.....	21
1.4.3 Вимоги до продуктивності	21
1.4.4 Експлуатаційні вимоги	21
1.4.5 Не функціональні вимоги до продукту	22
1.5 Висновки за розділом 1	22
2 Матеріали і методи	24
2.1 Вибір мови програмування	24
2.2 Вибір та обґрунтування Framework	27
2.3 Вибір СУБД.....	31
2.4 Проєктування бази даних.....	34
2.5 Висновки за розділом 2	37
3 Основні рішення щодо реалізації програми	38
3.1 Реалізація функціоналу системи	38
3.2 Проєктування інтерфейсу	42
3.3 Висновки за розділом 3	45

4	Експлуатація, тестування та експериментальне дослідження програми.....	46
4.1	Призначення й умови застосування програми	46
4.2	Вимоги до інтерфейсу	46
4.3	Умови виконання програми.....	47
4.4	Розгортання застосунку.....	47
4.4.1	Звернення до програми.....	48
4.4.2	Вхідні й вихідні дані	49
4.5	Методика та результати тестування	49
	Висновки	52
	Перелік джерел посилань	53
	Додаток А Технічне завдання	55
	Додаток Б Текст програми	59
	Додаток В Слайди презентації	69

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

ОС – операційна система;

ПЗ – програмне забезпечення;

ПК – персональний комп'ютер;

СЕД – Система Електронного Документообігу;

DMS – document management system;

GPL – General Public License;

ЕД – електронний документообіг.

ВСТУП

Ефективність керування підприємством, залежить від того, наскільки в ньому розумно організовано обіг документів. Обіг документів та управлінська діяльність тісно пов'язані одне з одним. Швидкість отримання інформації, необхідної для прийняття управлінського рішення залежить від того, наскільки оперативно здійснюється рух, опрацювання документів та їх передавання на виконання [1].

Під обігом документів враховують рух документів від моменту складання їх на конкретному підприємстві або одержання від інших підприємств до здачі в архів після опрацювання та систематизації [1].

За умов автоматизації управління і з переходом до зберігання інформації на електронних носіях обіг документів (документальне забезпечення управління) розуміють як створення інформаційної бази документів на різноманітних носіях для використання управлінським апаратом у процесі реалізації його функціоналу [1].

Електронний обіг документів працює відповідно до законодавства України або на підставі договорів. Відповідно до ст. 9 Закону України «Про електронні документи та електронний документообіг», електронний обіг документів – сукупність дій створення, оброблення, відправлення, передавання, одержання, зберігання, використання та знищення ЕД, які виконуються з перевіркою цілісності та в разі необхідності з підтвердженням факту одержання таких документів [1].

Автоматизований обіг документів дозволяє створити єдиний інформаційний простір, інтегруючи в інформаційний вузол усі документальні системи. Інтеграція відбувається без втрати якості роботи з документами. Основа подібної інтеграції є надійне сховище документів і функціонуючі з ним системи обігу документів. Всі документи зберігаються в єдиному хмарному сховищі, що забезпечує оптимальний пошук та відбір інформації при підготовці матеріалів. Однак робота з архівними документами є важливим

етапом при підготовці нових матеріалів. Інтеграція архіву автоматизованого обігу документів у єдиний інформаційний простір організації дозволить зробити доступ до створених матеріалів оперативним і ефективним. Доступ до глобальних інформаційних ресурсів, перехід на електронні технології документування, зберігання і передавання документів, тобто перехід на нові способи організації інформації і доступу до неї, ставлять перед службою документаобігу нові наукові і прикладні завдання [1].

Головне завдання автоматизованого документаобігу полягає в підвищенні ефективності та якості роботи користувачів та підприємства за рахунок впровадження системи прозорості руху документів і контролю їх виконання. Автоматизований документаобіг, в першу чергу, пов'язаний з будовою єдиного інформаційного середовища підприємства. Тому він має бути впроваджений на всіх робочих місцях пов'язаних зі створенням, редагуванням і зберіганням інформації [1].

Процес електронного документаобігу оснований на обробці обліково-звітної інформації, яка включає формування первинних ЕД, обробки інформації, автоматизована база даних. Електронний документаобіг забезпечує процеси створення, управління доступом і розповсюдження великих обсягів документів у комп'ютерних мережах, а також контроль за рухом документів в установі [1].

Основними цілями впровадження електронного документаобігу в організації:

- підвищення ефективності роботи управлінської діяльності;
- прискорення обміном документів між головними й підлеглими підрозділами у організації;
- зменшення трудомісткості роботи з документами та витрати часу на опрацювання всієї інформації, що у свою чергу дозволить підвищити оперативність й ефективність роботи діяльності всіх підрозділів [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

В першому розділі основну увагу буде приділено дослідженню предметної області системи обліку договорів. Спочатку буде розглянуто сучасний стан предметної області. Після цього буде досліджено можливості часткової автоматизації процесу обліку договорів, через використання додаткових програмних засобів. На основі попереднього дослідження буде виділено основних лідерів галузі та детально проаналізовано їх переваги та недоліки.

1.1 Аналіз важливості галузі системи обліку договорів

Від того, наскільки розвинута система обліку договорів залежить, ефективність керування підприємством. Адже, обіг документів та управлінська діяльність щільно пов'язані один з одним. Від того, наскільки швидко здійснюється рух, оброблення документів та їх переказування на виконання, залежить швидкість отримання інформації, необхідної для прийняття управлінського рішення [1].

Під обігом документів враховують рух документів від моменту складання їх на певному підприємстві або одержання від інших підприємств до здачі в архів після опрацювання та систематизації [1].

Обіг документів – це рух службових документів в установі від дати їхнього створення чи одержання до дати завершення виконання або надсилання. Цей термін обґрунтовує автоматизацію руху документів в середині установи, а також узагальнює їх шлях як невід'ємну складову робочого процесу. Процеси обігу документів розглядаються, в першу чергу, як інформаційне відображення і забезпечення управлінських дій, як система вторинних процесів, які забезпечують і відображають всі операції управління[1].

Автоматизований документообіг відбувається відповідно до законодавства України або на підставі договорів. Відповідно до ст.9 Закону України «Про електронні документи та електронний документообіг», автоматизований обіг документів – сукупність дій створення, редагування, відправлення, передавання, отримання, зберігання, використання та видалення ЕД, які виконуються із застосуванням перевірки на цілісність інформації та в разі необхідності з підтвердженням одержання таких документів [1].

Автоматизований документообіг дозволяє створити єдиний інформаційний хмарний простір, впроваджуючи в інформаційний вузол усі документальні системи. Впровадження здійснюється без втрачання якості роботи з документами, зі зберіганням всіх традицій діловодства. Основа подібної інтеграції – надійне хмарне сховище документів і вбудовані з ним системи документообігу. Всі документи зберігаються в єдиному хмарному сховищі, що дозволяє забезпечити швидкий та зручний пошук і відбір інформації для підготовки матеріалів. Однак робота з документами є важливим етапом при підготовці нових матеріалів. Інтеграція архіву електронного обігу документів у єдиний інформаційний простір організації дозволить зробити доступ до архівних матеріалів зручним та ефективним. Доступ до глобальних інформаційних ресурсів, перехід на автоматизовані технології документування, зберігання і передання документів, тобто перехід на нові способи ведення інформації і доступу до неї, ставлять перед службою діловодства нові наукові і прикладні завдання [1].

Головне завдання електронного документообігу полягає в спрощенні та підвищенні ефективності та якості роботи користувачів на підприємстві за рахунок впровадження системи прозорості руху документів і контролю їх виконання. Автоматизований обіг документів насамперед пов'язаний з будовою єдиного автоматизованого інформаційного середовища підприємства. Тому він має бути впроваджений на всіх робочих місцях пов'язаних зі створенням, редагуванням і зберіганням інформації [1].

1.2 Практичність та зручність використання застосунків для системи обліку договорів

Поява електронних документів зробила революцію в світі паперових договорів, спростила їх використання, додало швидкості пошуку інформації в них. Такі програми, як OpenDocMan, LogicalDOC Community Edition, Alfresco Content Services змінили уявлення людей про облік документації. З кожним роком вебзастосунки з обліку документації набирають популярності серед великих та вже навіть маленьких корпорацій, що сприяє збільшенню появи таких серверних вебзастосунків.

З появою програм для обліку договорів, зникла потреба у великих архівах на підприємстві, зберігання великої кількості папок в кабінеті, зменшився час на пошук потрібного документа, оскільки тепер це можна зручно знайти за пошуком у програмі, все знаходиться в одному місці і кожен працівник може знайти необхідний договір. Знижується ризик втрати або пошкодження документу через людські фактори.

Впроваджуючи таку функцію як редагування документа, що неможливо здійснити у паперовому вигляді, відбувається модернізація системи обліку в цілому.

Застарілість фізичного обліку договорів. Якщо порівнювати електронну систему з традиційною, то у останньої є досить багато недоліків, які заважають робітнику продуктивно використовувати свій робочий час та потенціал, що в цілому негативно відображається на його роботі.

Швидкість взаємодії з документами. У великих компаніях зазвичай велика картотека, архів і на пошук потрібного договору людина витрачає велику кількість часу, що також впливає на продуктивність робітників і веде до її зниження. В програмі договір можна знайти за лічені хвилини, тоді як фізично на це можуть піти години, а іноді дні, якщо у вас велике підприємство, корпорація, тощо.

Людський фактор. При зберіганні документів в єдиній системі обліку, значно знижується ризик псування документа сторонніми факторами, такими як потоп, пожежа або крадіжка. Також паперові документи на довгострокову перспективу не зручні у використанні, оскільки, папір з часом втрачає свої властивості.

Прозорість всіх етапів діяльності компанії. В системі обігу документів фіксуються кожен документ, завдання або процес, супроводжуються обліково-реєстраційною інформацією, яка спрощує нагляд за термінами виконання, моніторингу, аналітику, планування [2].

Скорочення часу на дії з документами. За допомогою системи керівник і співробітники підприємства набагато швидше справляються з такими діями над документами, як створення, пошук,, редагування, узгодження, затвердження, відправка, перевірка, видалення та інші [2].

Зручність роботи в автоматизованій системі обігу документів. У СЕД передбачені - розподіл прав доступу, автоматичне завантаження з каталогу, повний цикл договірної обліку, розпізнавання для зручнішого пошуку, використання шаблонів, редагування і контроль версій [2].

Підвищення відповідальності працівників підприємства. Завдяки системі сповіщення у системі, співробітники не зможуть забути про свої завдання, а облік робочого часу і трудовитрат дозволить керівному складу ефективніше планувати розподіл справ між працівниками [2].

Покращення якості обслуговування клієнтів. Рух обліку документів буде відбуватися значно швидше і чіткіше, ніж в оффлайн режимі, а значить звільняє час для більш уважного ставлення до кожного клієнта і проявлення індивідуального підходу [2].

Гнучкість використання. Ще один плюс веб-серверного застосунку в тому, що його не потрібно встановлювати фізично на комп'ютер, що дозволяє спростити процедуру роботи з додатком і не прив'язує працівника до робочого місця.

1.3 Порівняння існуючих аналогів

Популярні програми електронного документообігу змінили спосіб доступу людей до документів. Ці додатки пропонують зручність, надійність та простоту у використанні, але незважаючи на їх переваги в них є також ряд недоліків. Нижче наведені основні пункти на які слід звернути увагу при роботі з електронними документами.

Розглянемо загальні переваги таких застосунків.

Швидкість взаємодії з документами. Під час роботи з електронною документацією працівник економить свій час, що в цілому пришвидшує його роботу, також це сприяє пришвидшенню взаємодії з документами. Застарілі паперові документи не можуть показати таку швидку взаємодію користувача з договорами.

Підвищена захищеність даних. Самі по собі договори являються дуже цінною інформацією, яка є вразливою до зовнішніх факторів, крадіжок, тощо. Щоб підвищити надійність зберігання таких даних в електронному вигляді, кожному користувачу надаються певні права, без яких стороння людина не зможе скористатися або подивитися ту інформацію на яку в неї немає дозволу. Це і є вагомою перевагою паперу від цифровізації.

Зручність використання. Електронна система здатна в будь-який момент часу змінити той чи інший документ. Завдяки цьому працівник витрачає свій робочий ресурс, не на безглузді речі, такі як написання документа знову і знову кожен раз коли трапляється помилка, а всього лише виправляє помилку там де це потрібно.

Розглянемо загальні недоліки таких застосунків.

Можливість пошкодження документа. Незважаючи на те, що електронна система має досить високу надійність, є ризик пошкодження документа, що в цілому несе за собою негативні наслідки, бо якщо документ пошкоджений відновити його досить складно, майже неможливо.

Системи електронного документообігу мають багато переваг. Вони є досить надійними, швидкими у роботі, мають високу захищеність даних та зручні у використанні. Завдяки цьому електронна система є такою актуальною в сучасному світі. Вона дозволяє з легкістю і без зайвого напруження працювати з будь-якою документацією незалежно від місцезнаходження працівника, компанії тощо. І саме тому люди сприяють модернізації і йдуть в ногу з часом, щоб залишати продуктивність на найвищому рівні.

1.3.1 Застосунок «OpenDocMan»

OpenDocMan — це безкоштовна веб-система керування документами (DMS) із відкритим вихідним кодом, написана на PHP, розроблена відповідно до ISO 17025 і стандарту OIE для керування документами. Він має веб-доступ, детальний контроль доступу до файлів, а також автоматичне встановлення та оновлення [3].

OpenDocMan було розроблено за ліцензією GPL з відкритим вихідним кодом, яка, коротко кажучи, дозволяє вам безкоштовно використовувати програму та змінювати її будь-яким способом. Розробники заохочують відгуки користувачів, коли останні стикаються з проблемами або мають пропозиції [3].

На відміну від більшості DMS, які вимагають від користувача змінити бізнес-правила, щоб керувати своїми документами, OpenDocMan дозволяє інтегрувати поточні правила. Завдяки підтримці кількох типів файлів, для перегляду відділами, OpenDocMan дозволяє зосередитися на поточному бізнесі, керуванні документами [3].

Основи програмного забезпечення для керування документами. Програмне забезпечення для керування документами та керування документами в цілому базується на концепції централізованого зберігання документів, обмеженого доступу та відстеження змін. Ці три основи

програмного забезпечення DMS складають більшу частину ISO 17025 щодо систем керування документами [3].

Централізоване зберігання документів. Будь-яка система управління документами (DMS), будь то з відкритим кодом або комерційна, дозволить зберігати документи в якомусь централізованому місці. Це значно полегшує пошук документів у DMS, оскільки є лише одне місце для пошуку [3].

Обмежений доступ. Наявності централізованого місця для зберігання файлів недостатньо, щоб назвати ваш продукт системою керування документами. Проста папка на вашому комп'ютері може дозволити зберігати файли, але обмеження доступу до певних осіб стає більшою проблемою. Хороший DMS забезпечить точний доступ до кожного файлу [3].

Відстеження змін. Обмеження доступу до файлів DMS є хорошим кроком у правильному напрямку, але не приносить користі, якщо файли можна змінювати, додавати, видаляти без відстеження цих дій. Хороша система програмного забезпечення для керування документами додасть відстеження змін для файлів, щоб зміни можна було відзначити та скасувати, якщо потрібно [3].

1.3.2 Сервіс «Alfresco Content Services»

Alfresco Content Services включає в себе повнофункціональні можливості корпоративної системи керування контентом (ECM), які дозволяють працювати користувачу зі своїм вмістом із провідних світових пакетів і інструментів продуктивності — без перемикання програм [4].

Alfresco Content Services надає відкриті, гнучкі, високомасштабовані можливості керування корпоративним контентом (ECM) із додатковими перевагами платформи контент-сервісів — роблячи контент доступним, де б ви не працювали, завдяки легкій інтеграції з бізнес-додатками, якими користувач користується щодня [4].

Платформу Alfresco, побудовану на ядрі з відкритим вихідним кодом і підтримкою відкритих стандартів і API, легко інтегрувати та налаштовувати відповідно до потреб вашого бізнесу [4].

Програма підтримує:

- Alfresco Development Framework, який надає широкий спектр багаторазово використовуваних компонентів із можливістю конфігурації, які можна використовувати для створення сучасних адаптивних веб-додатків і програм для мобільних користувачів;
- підтримка провідних продуктивних програм, зокрема Microsoft Office, Google Docs, Microsoft Outlook тощо;
- широкий спектр готових інтеграцій і рішень від активної партнерської спільноти Alfresco, включаючи SAP, AutoCAD тощо [4].

Відкритий і модульний підхід Alfresco надає користувачу масштабовану платформу контент-сервісів, яка може обробляти зростаючий обсяг неструктурованого вмісту та мільярди документів, а також хмарну власну архітектуру, яка дозволяє повною мірою користуватися перевагами контент-сервісів [4].

Цифровий робочий простір Alfresco підвищує продуктивність завдяки швидкому й легкому доступу до файлів і робочих елементів, а також дозволяє розширювати додатки відповідно до конкретних потреб користувачів і покращувати можливості керування вмістом. Створено на основі Alfresco Developer Framework [4].

1.3.3 Сервіс «Kimios»

Kimios — це програма для керування документами з відкритим кодом, яка використовується для зберігання, редагування, створення та відстеження документів. Kimios пропонує розумні цифрові рішення для робочого процесу та контролю документів [5].

Kimios був розроблений, щоб допомогти компаніям і відділам з великим документообігом: відділ кадрів, фінансовий відділ, служба якості тощо [5].

Створення, редагування та передача документів в одному місці [5].

Програмне забезпечення для керування документами Kimios дозволяє всім користувачам отримувати доступ до нього безпосередньо з будь-якого браузера. За допомогою Kimios можна віддалено отримати доступ до всіх функцій DMS: створювати та редагувати документи, співпрацювати з іншими користувачами, завантажувати та систематизувати документи [5].

Легкий доступ – дозволяє позбутися чатів чи електронних листів.

Інтерфейс Kimios розроблено відповідно до будь-яких веб-стандартів і забезпечує повну взаємодію з користувачем у будь-якому браузері. Крім того, інтерфейс Kimios має мобільну версію, яка дозволяє користувачам смартфонів легко отримувати доступ до документів [5].

Простий пошук і зручна система індексування є одними з найбільших переваг програми керування документами Kimios. Кожен документ індексується відповідно до його вмісту та метаданих, і його можна шукати за вмістом [5].

Система керування документами Kimios підтримує всі типи контенту: текст, зображення, відео, 3D-об'єкти та багато іншого [5].

Плагіни Office передають всю потужність Kimios безпосередньо в середовище Microsoft Office [5].

1.3.4 Порівняння аналогів

Порівняння аналогів програмних продуктів документообігу наведено у таблиці 1.1 та відбувається за такими критеріями:

- безкоштовність;
- зручність інтерфейсу;
- вимоги до системи;

- контроль документів;
- швидкість навчання роботі;
- робота з хмарними технологіями.

Таблиця 1.1 – Порівняння аналогів

Критерій	Alfresco CS	OpenDocMan	Kimios	Продукт що розроблюється
Безкоштовний	-	-	-	+
Зручність інтерфейсу	+	-	+	+
Вимоги до системи	+	+	+	+
Контроль документів	+	+	+	+
Швидкість навчання роботі	-	+	+	+
Робота з хмарними технологіями	+	+	+	+

1.4 Постановка завдання

Головним завданням роботи є створення системи, яка дозволяє користувачу здійснити облік договорів на підприємстві.

Система повинна показувати всі договори, які існують. Створювати нові та змінювати вже існуючі. Також, додавання к договору документу Word та отримання різних версій змін цього документу.

1.4.1 Функціональність системи

Функціональні вимоги:

- можливість пошуку документів за деякими критеріями;
- можливість перегляду всіх документів у системі;

- можливість перегляду детальної інформації документу;
- можливість створення нового та редагування існуючого документа;
- можливість видалення документа;
- можливість завантаження до документа Word фалів;
- можливість завантаження з системи на свій ПК прикріплений файл до документу;
- можливість перегляду всіх контрагентів;
- можливість перегляду детальної інформації про контрагента та можливість змінювати інформацію про нього;
- можливість додавання нового та видалення старого контрагенту.

1.4.2 Вимоги до інтерфейсу

Вимоги до інтерфейсу системи документообігу:

- інтерфейс повинен бути в приємних кольорах;
- система повинна мати зручний інтерфейс;
- інтуїтивно зрозумілий для користувача;
- продуктивно використовувати простір екрану.

1.4.3 Вимоги до продуктивності

Система повинна забезпечувати швидкий пошук потрібного договору.

Система повинна стабільно працювати при великій кількості працюючих в ній робітників.

1.4.4 Експлуатаційні вимоги

Для експлуатації клієнтської частини необхідні такі програмно-технічні засоби:

- процесор із тактовою частотою від 1,5 GHz;
- комп'ютер з графічною ОС;
- оперативної пам'яті від 2GB;
- підключення до Internet мережі.

Також для стабільної роботи системи повинен бути як мінімум один комп'ютер-сервер та людина, що буде контролювати його коректну роботу.

1.4.5 Не функціональні вимоги до продукту

Не функціональні вимоги до продукту наступні:

- продуктивний хостинг;
- вебсервер Gunicorn;
- підтримка Python та Django.

Для експлуатації клієнтської частини необхідні такі програмно-технічні засоби:

- процесор із тактовою частотою від 1,5 GHz;
- комп'ютер з графічною ОС;
- оперативної пам'яті від 2GB;
- підключення до Internet мережі.

1.5 Висновки за розділом 1

В першому розділі основну увагу було приділено дослідженню проблемної області системи обліку договорів. Спочатку було розглянуто сучасний стан проблемної області. Після цього було досліджено можливості часткової автоматизації процесу виконання обліку договорів, через використання програмних засобів. На основі попереднього дослідження було виділено основних лідерів галузі та детально проаналізовано їх переваги та недоліки.

З рештою, це все дозволило виконати формування переліку функціональних та нефункціональних вимог до майбутнього нового рішення, що і було головною метою цього розділу.

2 МАТЕРІАЛИ І МЕТОДИ

В першому розділі основну увагу було приділено дослідженню проблемної області системи документообігу. Було розглянуто сучасний стан проблемної області. Після цього було досліджено можливості часткової автоматизації процесу документообігу.

В другому розділі буде проводитися вибір та затвердження технічних засобів розробки: технологія реалізації застосунку, мова програмування з framework, система БД MySQL . Порівняння та аналіз переваг та недоліків дозволить зробити ґрунтовний вибір.

2.1 Вибір мови програмування

Це високорівнева мова програмування, яка по праву належить до одних із найпростіших і найзрозуміліших, завдяки своєму синтаксису та читабельності. Вважається ідеальним для початківців, тому що програмувати на Python набагато швидше і простіше, ніж, наприклад, Java Script. Ви можете вивчити синтаксис Python, використовуючи звичайний підручник на офіційному сайті. Все досить добре написано [6].

Якщо ви розробник-початківець, то універсальний Python буде найкращим вибором [6].

Python – мова інтерпретується і перед запуском є просто текстовим файлом. У ньому немає етапу компіляції. Він логічний, добре спроектований та підходить для роботи практично на всіх платформах [6].

Крім того, Python є гарним для більшості проектів, таких як:

- веб-розробка;
- застосунки;
- програмне забезпечення для мобільних пристроїв;
- вбудовані системи і багато іншого.

Якщо говорити про те, де найчастіше використовується ця мова, то лідером, безперечно, буде веб-розробка. Що стосується мобільних програм, то серверна частина зазвичай пишеться на Python, а для пристроїв Android та IOS використовуються інші мови [6].

Python часто є основою для відтворення вбудованих систем, таких як Raspberry Pi [6].

Мова використовується для створення засобів регулювання різних показників, таких як температура, тиск, витрати різних ресурсів. Саме тому Python так любимо розробниками популярних систем розумного будинку [6].

Знання бібліотек Python дуже важливе. Django, JSON, знання алгоритмів та індивідуальні навички відкриють для вас двері багатьох компаній. Звичайно, компаній на ринку багато, всі різні вимоги теж можуть відрізнятися, але ця мінімальна база дасть покупцеві конкурентну перевагу перед іншими [6].

Сьогодні, коли обчислювальна потужність персональних комп'ютерів і серверів стала досить високою, існує величезний попит інтерпретовані мови програмування. Адже, крім запуску самої програми, необхідно запустити інтерпретатор, що потребує додаткових ресурсів. Python якраз така мова [7].

Популярність Python обумовлена тим, що ця мова здатна вирішувати широке коло завдань і її застосування можливе на всіх популярних сьогодні платформах, за винятком мобільного сегменту, де міцно вкоренилися специфічні мови. В останні роки Python набув популярності завдяки своїй ефективності в таких галузях розробки, як машинне навчання та наука про дані за рахунок своєї розширюваності та гнучкості, настільки необхідних у цьому сегменті програмування. При цьому Python широко використовується в WEB, в якому ця мова чудово почувається з початку 2000-х і продовжує розвивати і вдосконалювати цей напрям до сьогодні [7].

Давайте докладніше розглянемо всі переваги та недоліки цієї мови [7].
Переваги:

- гнучкість, на мій погляд, головна перевага мови, оскільки завдяки своїй гнучкості мова завоювала популярність серед багатьох розробників;
- можливість розширення — є бібліотеки та фреймворки під будь-які завдання та потреби, також величезний плюс у тому, що ми можемо використовувати С код з Python;
- простота синтаксису — із синтаксису прибрано все зайве, код чистий і зрозумілий без зайвих дужок та виразів;
- інтерпретатор Python існує для всіх популярних платформ і включений за замовчуванням у більшість дистрибутивів Linux, а це означає, що він доступний на більшості серверів «з коробки»;
- PEP — єдиний стандарт написання коду, який робить код зручним та читабельним, навіть якщо він передається від одного програміста до іншого. Це робить Python популярним;
- Open Source — код інтерпретатора Python відкритий, що дозволяє всім, хто цікавиться розвитком мови, брати участь у її розробці та покращувати її, якщо подивитись подробиці релізу однієї з версій мови, то можна побачити, що величезні частини нового функціоналу реалізовані сторонніми розробниками;
- ком'юніті — навколо Python сформувалася досить дружня і приємна спільнота, яка готова прийти на допомогу будь-якому початківцю або вже досвідченому розробнику і розібратися в його проблемі [7].

Недоліки:

- продуктивність — Python не такий швидкий, порівняно, це пов'язано з тим, що Python інтерпретується, але навіть у порівнянні з іншими інтерпретованими мовами помітно, що Python програє у продуктивності, але це легко нівелюється за допомогою реалізацій С тієї чи іншої проблемної частини коду, в умовах сьогоденних потужностей не дуже помітно;
- динамічна типізація — через динамічну типізацію Python споживає більше ресурсів, ніж міг би, але це часто компенсується внутрішнім кешуванням;

– Global Interpreter Lock, в даний час це серйозна проблема з продуктивністю в Python, і вона також викликана поганою реалізацією багатопоточності, код GIL не змінився з моменту виходу першої версії мови, що ясно вказує на те, що він застарів [7].

Порівняння мов програмування надані у таблиці 2.1.

Таблиця 2.1 – Порівняння мов програмування

Критерій	PHP	Python	ASP.NET
Надбудови для структуризації розробки	Багато	Багато	Мало
Серверна підтримка	Розповсюджена	Розповсюджена	Недостатньо розповсюджена
Масштабованість розробок	Проста	Проста	Проста
Складність підтримки	Низька	Середня	Середня

2.2 Вибір та обґрунтування Framework

Django є надзвичайно популярним та повнофункціональним серверним веб-фреймворком, написаним на Python. Даний модуль розповість про те, чому Django є одним з найпопулярніших серверних веб-фреймворків, як встановити середовище розробки, і як почати використовувати його для створення власних веб-додатків [8].

Щоб створити якісний та потужний веб-ресурс, потрібно розуміти, який функціонал він матиме. Залежно від цього програміст вибирає, який фреймворк та мову програмування використовувати. Python Django допомагає реалізувати складні проекти та створити функціональний сайт, який не поступається YouTube, Pinterest або Instagram. [9].

Створення сайтів на Django – популярна послуга. Ця платформа має гарний функціонал і призначена для створення якісних проєктів. Фреймворк використовує мову програмування Python. Багато сайтів побудовані на цій платформі, зокрема Instagram, Google, Pinterest, Mozilla [9].

Спочатку розробники планували створювати Django сайти з інформаційним ухилом, тому архітектура була відповідною. Бібліотека має великий набір інструментів, що дозволяє налаштувати адміністративну частину сайту та функціонал управління контентом. Надалі можливості розширилися, і тепер на цій платформі можна створити практично будь-який сайт [9].

Можливості платформи Django великі [9]:

- Вже готовий розроблений інтерфейс адміністративної панелі сайту із підтримкою великої кількості мов;
- архітектура додатків дуже легко підключається, тому її можна встановлювати на будь-які сайти Django;
- розроблена система кешування;
- сприятлива система фільтрів;
- вбудована документація.

І це лише мала частина того, що може запропонувати Django. Це дуже зручна та потужна платформа для створення веб-ресурсів із великим розробленим функціоналом та можливостями [9].

Розробка сайту на Django дозволяє розробити безліч завдань. Деякі розробники вважають, що це одна із CMS – систем керування контентом, але це не так, це дуже потужний інструмент, що дозволяє розробляти програмне забезпечення та сайти. Він справляється з великими навантаженнями, може виконувати велику кількість завдань [9]. Серед іншого, Django виконує такі завдання:

- за допомогою нього можна розробити повноцінну CRM-систему із потужним функціоналом;

- за допомогою Django можна розробити CMS із необхідними можливостями для веб-ресурсу;
- призначена для розробки комунікаційних платформ;
- застосовується до створення системи управління обігом документів;
- можна створювати веб-ресурси для електронної розсилки.

Фреймворк призначений для створення систем фільтрації зі складними параметрами, машинного навчання, аналізу великої кількості даних. Розробка сайтів Django дозволяє створити проєкт будь-якої по складності. Багато досвідчених розробників вважають за краще користуватися саме цією платформою [9].

Створення сайту з фреймворком Django є приємним для розробника процесом. Річ в тому, що фреймворк максимально зручний та зрозумілий. Легко вчасно виявити помилки та вирішити складні завдання. Є кілька причин, чому розробники вважають цей фреймворк чудовим та рекомендують його для розробки [9].

Дуже розвинена екосистема. Багато розробників усвідомлюють Django як систему, яка легко та зручно поєднується з іншими сторонніми програмами. Платформа вже має вбудовані блоки, із яких можна конструювати сайт. Наприклад, блок форми редагування чи створення нового об'єкту застосовується багатьох ресурсах. Програмістові не потрібно кожен раз писати новий код, достатньо створити готове рішення та довести його до ідеалу.

Готова адміністративна панель. Вона відразу ж створюється під час створення програми, що значно допомагає заощадити час на розробку сайту, оскільки програмісту не потрібно окремо розробляти адміністративну панель.

Можливість розширення. Створення сайту на Django дуже зручне, тому що функціонал цієї платформи дуже легко розширити за допомогою команд. Можна встановити додаткові плагіни, для більш зручного створення вебзастосунку.

Великий набір бібліотек. У бібліотеках зберігаються вже створені функції, класи, що полегшують вирішення певних завдань. Це дуже спрощує

роботу розробника та зменшує час створення сайту. Крім того, ймовірність помилок знижується, якщо використовувати готові та перевірені бібліотеки.

Висока швидкість роботи. Вважається, що мова програмування Python не занадто швидка, але фреймворк Django прискорює швидкість роботи і здатний витримувати великі навантаження. Тому фреймворк підходить для створення потужних сайтів із високою відвідуваністю користувачів.

Порівняння Framework надано у таблиці 2.2.

Таблиця 2.2 – Порівняння Framework

Критерії	Django	Symfony
Мова	Python	PHP
Популярність	+	+
Архітектура	MVT	MVC
Швидкодія	Відомий своєю швидкодією та ефективністю. Має добрі механізми кешування та оптимізації запитів.	Також ефективний фреймворк, проте може вимагати більше ресурсів сервера для виконання.
Розширюваність та пакети	+	+
Фокус на певних завданнях	Зосереджений на швидкій розробці веб-додатків з підтримкою адміністративного інтерфейсу.	Забезпечує розробку складних та масштабованих додатків з фокусом на розширюваність.

Перераховані вище порівняння показують, чому розробники впевнені, що розробка застосунку на Django просте та ефективне, але найголовніше – сайт виходить швидким та функціональним. Далі справа за вебдизайнерами, які повинні створити веб-ресурсу інтерфейс та доведуть до фінальної стадії розробки [9].

Створення сайтів на Django є актуальним та затребуваним на ринку. Платформа дозволяє швидке орієнтування у коді, застосовувати готові рішення з бібліотек та значно прискорювати процес розробки застосунку[9].

Зазвичай, для невеликих інтернет-магазинів і маленьких проєктів використовувати фреймворк Django не актуально, є способи розробити проєкт набагато простіше. Але для великих ресурсів цей фреймворк відмінно підходить [9].

2.3 Вибір СУБД

Бази даних (в т.ч. система MySQL) є сутністю для зберігання інформації у вигляді таблиць. Щоб чужі БД були доступні абсолютно кожному користувачеві на сервері, існує система користувачів цих баз даних. Доступ до будь-якої БД може бути призначений адміністратором (або уповноваженим користувачем) іншому користувачеві, причому він може бути повним або обмеженим. Більш конкретно цей ступінь доступу виражається в привілеях («правах» або «дозволах») [10].

База даних є структурованим набором даних. Ці дані можуть бути будь-чим — від простого списку майбутніх покупок до списку експонатів художньої галереї або величезної кількості інформації в корпоративній мережі. Для запису, отримання та обробки даних, що зберігаються в комп'ютерній базі даних, потрібна система управління базою даних, якою є програмне забезпечення MySQL [11].

Оскільки комп'ютери добре справляються з обробкою великих обсягів даних, управління базами даних грає центральну роль обчисленнях. Таке управління може бути реалізовано по-різному як у вигляді окремих утиліт, так і у вигляді коду, включеного в інші додатки [11].

MySQL – це система управління базами даних. У реляційній базі дані зберігаються не всі разом, а в окремих таблицях, завдяки чому досягається вигаш у швидкості та гнучкості. Таблиці пов'язані друг з одним відносинами,

що дозволяє об'єднувати дані з кількох таблиць під час запиту. SQL як частина системи MySQL можна охарактеризувати як мову структурованих запитів плюс найпоширенішу стандартну мову, яка використовується для доступу до баз даних [11].

MySQL це програмне забезпечення з відкритим вихідним кодом. Будь-хто може використовувати його та модифікувати. Таке програмне забезпечення можна отримати через Інтернет та використовувати безкоштовно. При цьому кожен користувач може вивчити вихідний код та змінити його відповідно до своїх потреб [11].

Чому веб-програмісти віддають перевагу СУБД MySQL? MySQL дуже швидкий, надійний та простий у використанні. Якщо вам потрібні саме ці якості, спробуйте працювати із серверними даними. MySQL також має низку зручних функцій, розроблених у тісному контакті з користувачами. Спочатку сервер MySQL розроблявся для управління великими базами даних з метою забезпечення вищої швидкості роботи в порівнянні з аналогами, що існують на той момент. І вже кілька років цей сервер успішно використовується в умовах промислової експлуатації з високими вимогами. Незважаючи на те, що MySQL постійно вдосконалюється, сьогодні він надає широкий набір корисних функцій. Завдяки своїй доступності, швидкості та безпеці MySQL дуже добре підходить для доступу до баз даних через Інтернет [11].

MySQL - це клієнт-серверна система, що містить сервер з кількома SQL, що забезпечує підтримку різних комп'ютерів баз даних, а також кілька різних клієнтських програм та бібліотек, інструменти адміністрування та широкий спектр прикладних програмних інтерфейсів (API). Ми також поставляємо сервер MySQL у вигляді багатопоточної бібліотеки, яку можна підключити до користувача додатка, щоб отримати компактний, швидкий і простий в керуванні продукт. Існує також велика кількість програмного забезпечення MySQL, більшість якого безкоштовна [11].

SQL складається з двох частин: серверної та клієнтської. Сервер MySQL постійно працює на пристрої. Клієнтські програми (наприклад,

скрипти PHP) посилають SQL-запити до сервера MySQL через механізм сокетів (тобто з використанням мережових засобів), сервер їх обробляє і запам'ятовує результат. Тобто скрипт (клієнт) вказує, яку інформацію хоче отримати від сервера з даними. Потім сервер бази даних надсилає відповідь (результат) клієнту (скрипту) [11].

Чому іноді передається не весь результат? Дуже просто: річ в тому, що розмір кінцевого набору даних може бути занадто великим, і на його передачу по мережі піде занадто багато часу. Та й мало коли буває потрібно отримувати відразу весь запит (тобто всі записи, що задовольняють заданому запиту). Наприклад, нам потрібно лише підрахувати, скільки записів задовольняє тій чи іншій умові, або ж вибрати з даних тільки перші 30 записів. Механізм застосування сокетів має на увазі технологію клієнт-сервер, а це означає, що в системі повинна бути запущена спеціальна програма - MySQL-сервер, яка приймає і обробляє запити які їй надходять. Так як весь процес відбувається в дійсності на одному комп'ютері, витрати по роботі з мережевими засобами незначні (установка і підтримка з'єднання з MySQL-сервером обходиться досить дешево) [11].

Структура MySQL трирівнева: бази даних - таблиці - записи. Бази даних і таблиці MySQL фізично представлені файлами з розширеннями frm, MYD, MYI. Таблиця являє собою сукупність записів. Записи являє собою сукупність полів різного типу. Назва бази даних MySQL унікальна в межах системи, а таблиці - в межах тільки бази даних, поля - в межах певної таблиці. Один сервер MySQL може тримати в собі відразу кілька баз даних, доступ до яких може відрізнятися логіном і паролем. Знаючи потрібний логін та пароль, можна використовувати певну базу даних. Наприклад, можна створити, редагувати або видалити в ній таблицю, додавати записи і т. д. Здебільшого логін і пароль призначаються хостинговою компанією, які і забезпечують підтримку MySQL-серверів для своїх користувачів [11].

2.4 Проектування бази даних

Загальний погляд на дані представляється на концептуальній моделі.

Є два головних підходи до проектування даних при концептуальному проектуванні:

- перший – семантичні моделі;
- другий – об'єктні моделі.

Структурі даних головну увагу приділяють семантичні моделі. Найпоширенішою семантичною моделлю є модель сутності-зв'язку (ER-модель). Модель ER складається з сутностей, відносин, атрибутів, доменів, ключів. Моделювання даних відображає логічну структуру даних так само, як блок-схеми відображають логічну структуру програми [12].

Об'єктні моделі зосереджені на поведінці об'єктів і засобах маніпулювання даними. Основним поняттям таких моделей є об'єкт, сутність, яка має стан і поведінку. Стан об'єкта визначається набором його атрибутів, а поведінка об'єкта – заданим для нього набором операцій. [12].

ER-моделювання — це низхідний підхід до проектування бази даних, зображений на рисунку 2.1, який починається з визначення найважливіших даних, які називаються об'єктами, і зв'язків між даними, які мають бути представлені в моделі. Потім модель доповнюється властивостями сутності та зв'язку, які називаються атрибутами, а також обмеженнями, пов'язаними з сутностями, зв'язками та атрибутами. Модель ER надає графічне представлення логічних об'єктів та їх зв'язків у структурі бази даних через діаграму [12].



Рисунок 2.1 – Послідовність проведення ER-моделювання [12]

Сутність дозволяє моделювати клас об'єктів одного типу. Сутність має унікальне ім'я в даній системі. Передбачається, що в системі є багато екземплярів цієї сутності. Об'єкт, якому відповідає сутність, має набір атрибутів, що характеризують його властивості. Цей набір має бути таким, щоб можна було розрізняти конкретні екземпляри сутності. Набір атрибутів, який однозначно ідентифікує конкретний екземпляр сутності, називається ключовим атрибутом [12].

Предметна область системи документообігу включає в себе сутності:

- «Contracts»;
- «Customers»;
- «StorageDoc»;
- «Users».

Сутність «Contracts» служить для збереження даних договорів. Має в собі такі атрибути:

- id – ключовий атрибут;
- id_customer – атрибут зв'язку між сутностями «Contracts» та «Customers»;

- id_user – атрибут зв'язку між «Contracts» та «Users»;
- contract_number – атрибут збереження інформації о номері договору;
- contract_number_registration – атрибут збереження інформації о реєстраційному номері договору;
- contract_amount – атрибут зберігає вартість;
- contract_end_date – атрибут зберігає дату закінчення договору;
- contract_date_signed – атрибут зберігає дату підписання;
- contract_date_customer_signed – атрибут зберігає дату підписання контрагентом.

Сутність «StorageDoc» служить для зберігання Word документів кожного договору. Має в собі такі атрибути:

- id – ключовий атрибут;
- name – зберігає ім'я документу;
- id_contract – атрибут зв'язку між сутностями «Contracts» та «StorageDoc»;
- document_file – атрибут зберігає Word документ у БД;
- date_version – атрибут зберігає дату завантаження документу.

Сутність «Customers» служить для зберігання інформації про контрагентів. Сутність має в собі такі атрибути:

- id – ключовий атрибут;
- okpo – атрибут зберігає інформацію про ЄДРПОУ контрагента;
- name – атрибут зберігає найменування контрагента;
- phone – атрибут зберігає телефон контрагента;
- email – атрибут зберігає електронну пошту контрагента.

Сутність «Users» служить для зберігання інформації про користувачів. Сутність має в собі такі атрибути:

- id – ключовий атрибут;
- username – атрибут зберігає ім'я для входу в систему користувача;

- password – атрибут зберігає пароль користувача;
- first_name – атрибут зберігає ім'я користувача;
- last_name – атрибут зберігає прізвище користувача;
- email – атрибут зберігає пошту користувача.

Під час концептуального проєктування була розроблена модель бази даних. Розроблена БД зображена на ER-діаграмі (рис. 2.2).

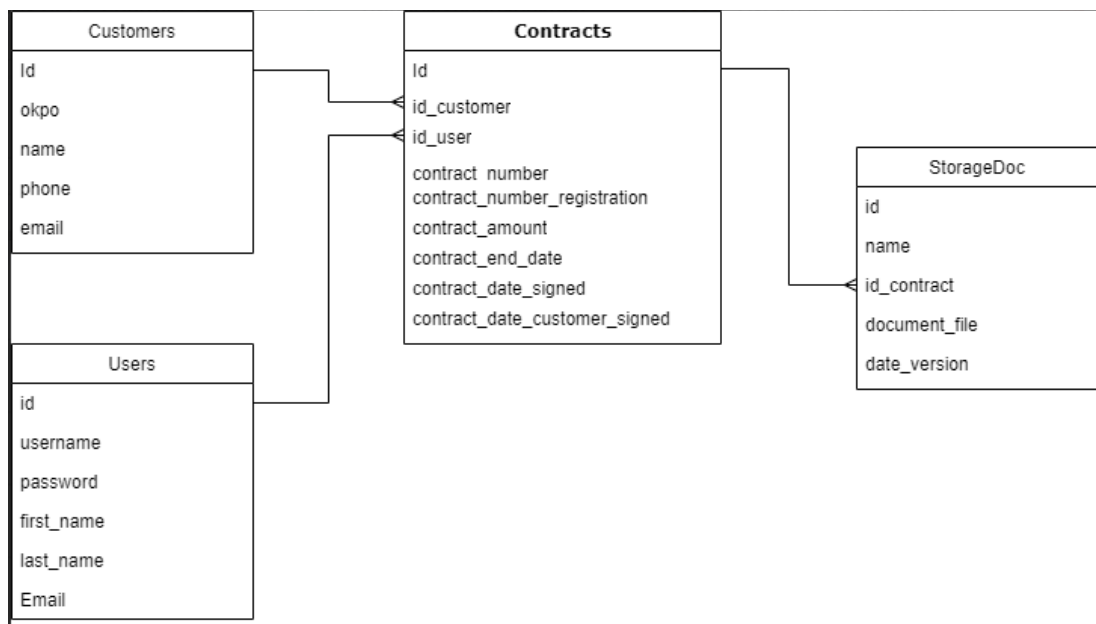


Рисунок 2.2 – ER-діаграма

2.5 Висновки за розділом 2

В другому розділі було виконано роботу з вибору та затвердження технічних засобів розробки: технологія реалізації застосунку, мова програмування, надбудови для організації розробки. Аналіз переваг та недоліків дозволили зробити ґрунтовний вибір: мова Python, Framework Django.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ ПРОГРАМИ

3.1 Реалізація функціоналу системи

Функціональна вимога - це формулювання того, як повинна поводитися система. Він визначає, що потрібно робити системі, щоб задовольняти потребам або очікуванням користувача. Функціональні вимоги треба розглядати як дії, які хоче виконати користувач. Нефункціональні вимоги визначають, як система повинна всередині працювати [13].

Функціональні вимоги мають дві частини: функції та поведінки. Функція – це те, що повинна виконувати розроблена система. Поведінка буде залежить від того, як система виконує свої функції [13].

Архітектура застосунку вказана на рисунку 3.1.

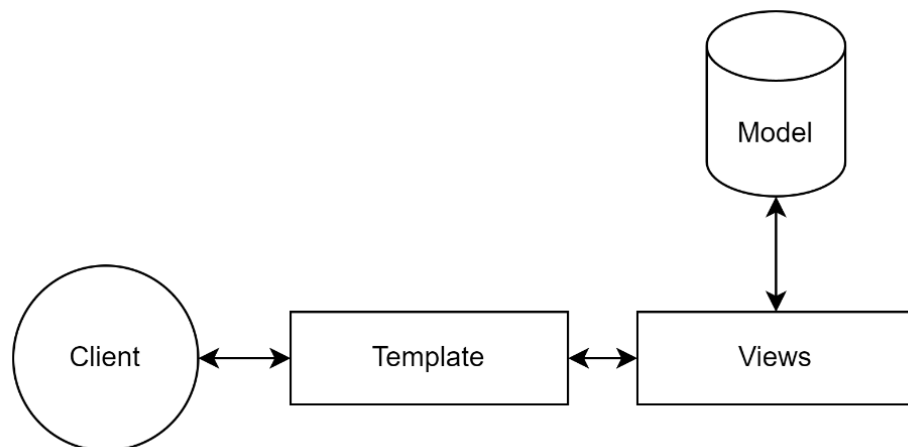


Рисунок 3.1 – Архітектура застосунку

В цій системі було розроблено форми, які виконують кожна свою функцію.

Перша форма відповідає за авторизацію користувача в систему, та зображена на рисунку 3.2.

Увійдіть у систему

Введіть логін та пароль

Логін

Пароль

Увійти

Рисунок 3.2 – Форма авторизації

Друга форма – це головна форма (рис. 3.3), яка відображає договори, які є в системі.

Документообіг					Вихід	About
- Головна - Контрагенти - Користувачі	Створити		1 Наступна >			Search...
	№ документа	Контрагент	Користувач	Регістраційний номер	Ціна	
	№4320-453	ООО "Провайдер"	Сергей Главатских	№263	1518.15	
	№458834-3456	ООО "Побутовий магазин"	Зоряна Главатских	№364	18563.57	
	№75834-543	ООО "Хостингова компанія"	Сергей Главатских	№589	5748.54	
	№75834-463	ФОП "Довженю"	Данило Гарвона	№1283	5435.43	
	№114345-43	ФОП "Чайковський"	Сергей Главатских	№54	4234.59	

Рисунок 3.3 – Головна форма

У верхньому меню розташована кнопка створити. За її допомогою ми відкриваємо третю форму, яка відповідає за створення нового договору (рис. 3.4).

Документообіг [Вихід](#) [About](#)

Головна
Контрагенти
Користувачі

Контрагент:

Користувач:

Номер документу:

Номер реєстрації:

Вартість:

Термін закінчення договору:

Дата підписання:

Дата підписання контрагентом:

Документ: файл не вибрано

Рисунок 3.4 – Форма створення нового договору

Також на головній формі, ми можемо вибрати договір та клацнути по номеру документа, за рахунок цього ми перейдемо на іншу форму, яка відображає повну інформацію про обраний документ (рис. 3.5).

Документообіг [Вихід](#) [About](#)

Головна
Контрагенти
Користувачі

№ контракту:	№4320-453
Контрагент	ООО "Провайдер"
Користувач	Сергей Главатских
№ реєстрації	№263
Ціна	1518.15
Строк дії	30-06-2023
Дата підписання	29-05-2023
Дата підписання контрагентом	27-05-2023
Документ	DocProvider.docx

Рисунок 3.5 – Форма інформації про обраний документ

Під інформацією договору є чотири кнопки, які відповідають за дії, які можна виконати з цим договором.

Якщо вибрати редагування ми потрапимо в форму редагування договору (рис. 3.6). Змінивши інформацію вибираємо зберегти, так ми зберігаємо оновлену інформацію, та повертаємося на сторінку з договором.

Документообіг

Вихід About

Головна
Контрагенти
Користувачі

Контрагент: ООО "Провайдер"

Користувач: Sergei

Номер документу: №4320-453

Номер реєстрації: №263

Вартість: 1518,15

Термін закінчення договору: 30.06.2023

Дата підписання: 29.05.2023

Дата підписання контрагентом: 27.05.2023

Документ: Вибрати файл | Файл не вибрано

Зберегти
Відміна

Рисунок 3.6 – Форма редагування обраного документа.

На форму, яка відображає контрагентів, ми можемо перейти за допомогою меню ліворуч. Форма відображає весь список контрагентів (рис. 3.7).

Документообіг

Вихід About

Головна
Контрагенти
Користувачі

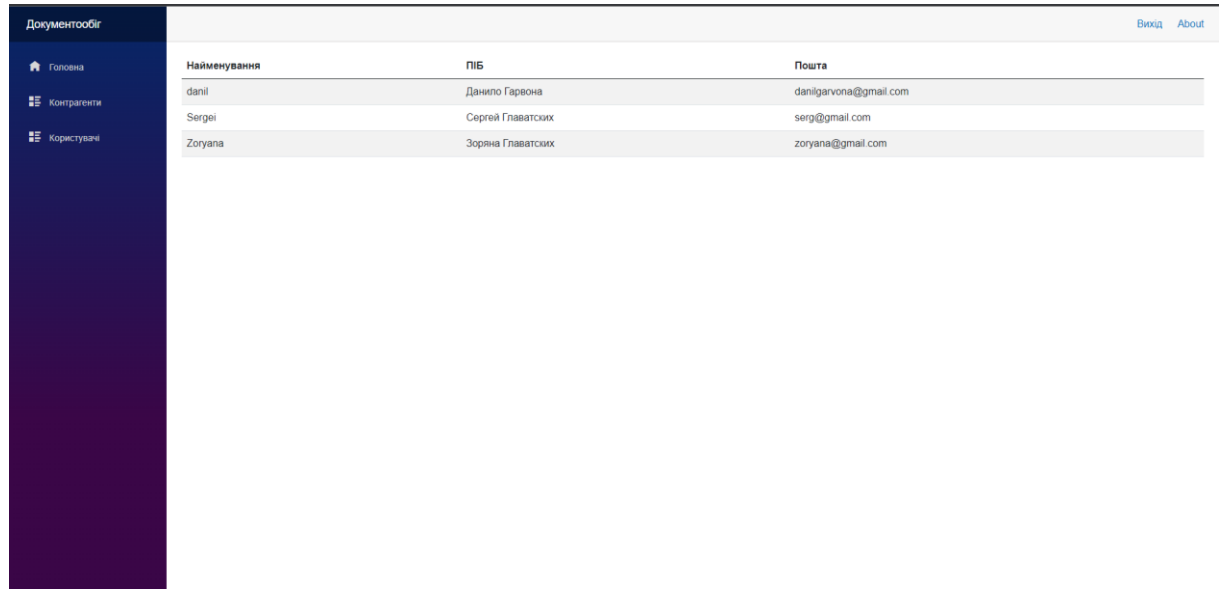
Створити

ІД	Найменування	Телефон	Пошта
38503254	ООО "Провайдер"	+3801785501	provider@gmail.com
56783294	ООО "Хостингова компанія"	+3802541881	host@gmail.com
475883489	ФОП "Чайковський"	+38001651778	chaikov@gmail.com
54385340	ФОП "Довженко"	+380164533	dovzhenko@gmail.com
542347	ФОП "Іванів"	+3801861351	ivanov@gmail.com
54385798	ООО "Товарний магазин"	+3801681651	magazin@gmail.com
5435654	ООО "Соняш"	+380105414	sun@gmail.com
4532545	ФОП "Тем'яченко"	+38015416	tem@gmail.com

Рисунок 3.7 – Форма з контрагентами

Функціонал з контрагентами аналогічний з контрактами.

У меню ліворуч ми можемо вибрати користувачі. У цій формі відображається інформація про користувачів системи (рис. 3.8).



Найменування	ПІБ	Пошта
danil	Данило Гарвона	danilgarvona@gmail.com
Sergei	Сергей Главатских	serg@gmail.com
Zoryana	Зоряна Главатских	zoryana@gmail.com

Рисунок 3.8 – Форма з користувачами

3.2 Проєктування інтерфейсу

Розробники інтерфейсів завжди повинні враховувати як будуть працюватимуть з їх застосунком, які фізичні і розумові здібності користувачей цієї системи. Люди, нажаль, можуть запам'ятати вельми обмежений об'єм інформації на короткий час і робити багато помилок при введенні великої кількості даних або при роботі у стресових умовах. Часто, фізичні можливості людей істотно можуть різнитися [14].

Перерахуємо й опишемо головні принципи проєктування інтерфейсу користувача [14]:

- врахування знань користувача — у інтерфейсі необхідно використовувати терміни, які будуть зрозумілі користувачам. Інакше кажучи, інтерфейс має бути настільки зрозумілим, щоб користувач звикав до нього без особливих старань, водночас потрібно використовувати терміни, зрозумілі

користувачеві, а предмети, керовані системою, мають бути обов'язково пов'язані з робочим середовищем користувача;

- узгодженість — однакові операції повинні виконуватись однаковим способом;

- мінімум несподіванок — поведінка системи має бути передбачуваною.

Розробнику інтерфейсу користувача необхідно вирішити дві основні проблеми: як користувач буде вводити дані в систему і як дані будуть представлені користувачеві [14].

Усі сучасні інтерактивні системи, незалежно від призначення, використовують кольори. Правильне використання кольорів полегшує сприйняття інтерфейсу користувача. В іншому випадку (при неправильному використанні кольорів) буде створено візуально непривабливий інтерфейс, що спровокує створенню помилок користувачем [14].

В цілому процес створення інтерфейсу можна поділити на такі етапи [15]:

- аналіз;
- проєктування;
- дизайн;
- впровадження та налагодження.

Вони зображені на рисунку 3.9.

Скільки часу займає кожен етап і що конкретно на ньому відбувається, залежить від багатьох обставин: складності проєкту, комерційних міркувань, розміру команди, досвіду її учасників [15]. Для невеликого проєкту в злагодженій команді терміни можуть бути приблизно такі як зображено на рисунку 3.9.

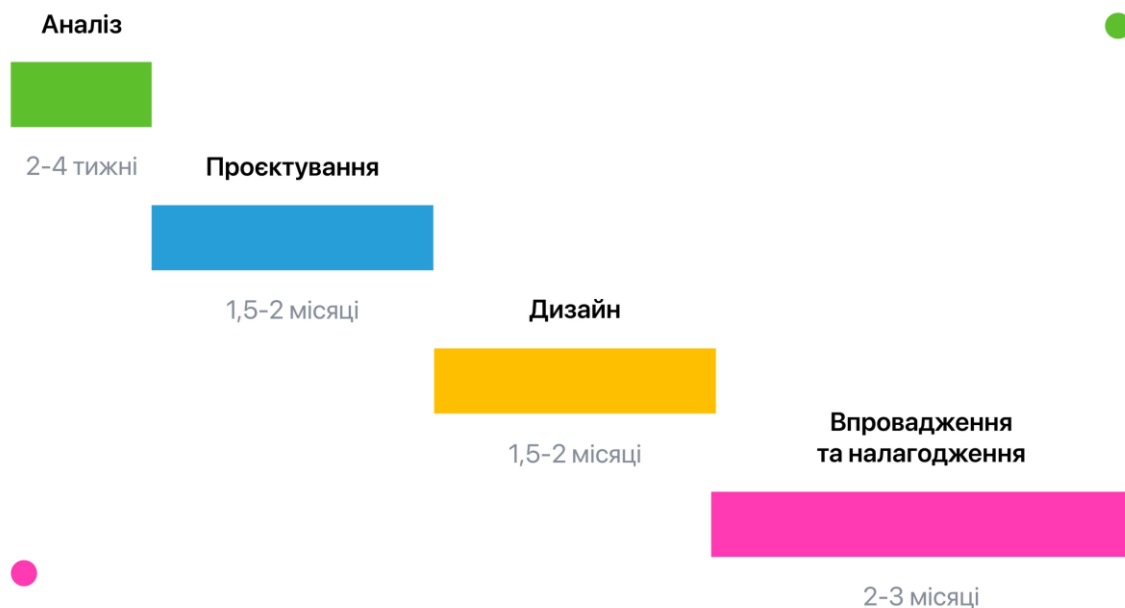


Рисунок 3.9 – Час на виконання кожного етапу [15]

В результаті виконаного дослідження було побудовано приблизний макет інтерфейсу майбутнього програмного забезпечення, який зображено на рисунку 3.10.

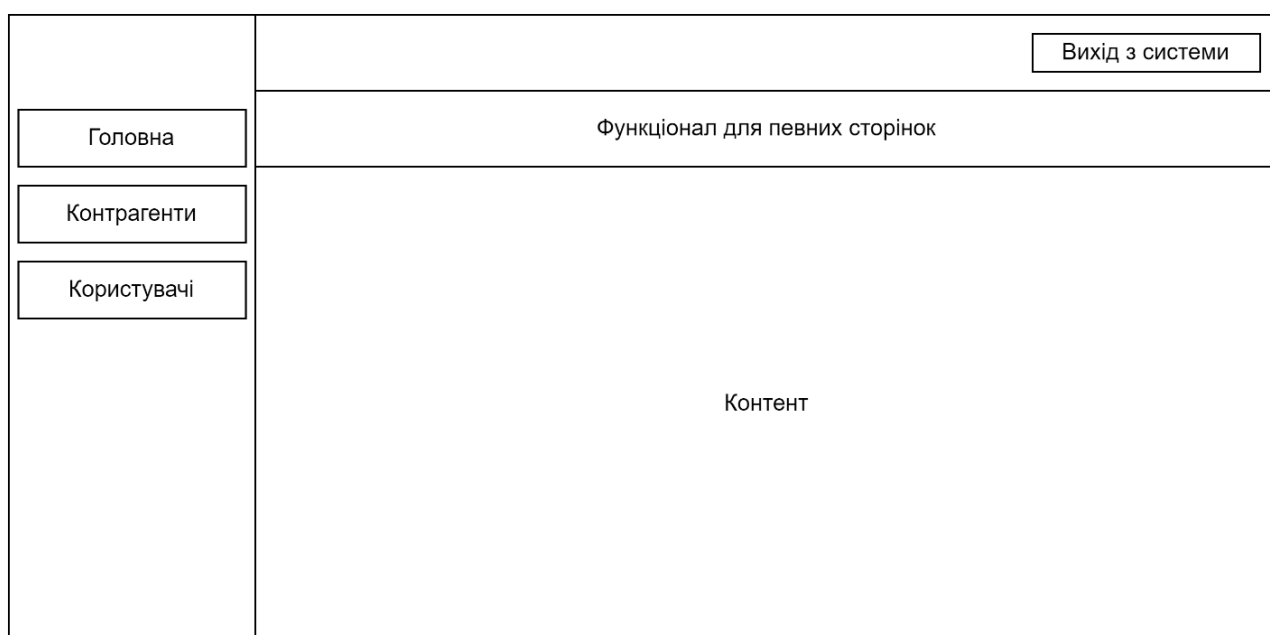


Рисунок 3.10 – Макет інтерфейсу

3.3 Висновки за розділом 3

Використовуючи засоби проєктування, було зроблене фізичне та логічне проєктування вебзастосунку. Спираючись на визначені принципи розробки графічного інтерфейсу було прийнято рішення, щодо подальшої реалізації ПЗ.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Головним завданням роботи є створення системи документообігу [16].

Система автоматизації документообігу – інформаційна система, що забезпечує процес управління і поширення електронних документів, а також забезпечує пильний контроль над рухом документів в організації [16].

Автоматизована система (АС) (англ. automated system) — це система в якій частину функцій виконує користувач, а іншу, сукупність керованих об'єктів й автоматичних керуючих пристроїв. АС забезпечує вироблення рішень на основі автоматизації майже всіх процесів у різних сферах діяльності (управління, проєктування, виробництво) або їх поєднаннях [16].

Функціональні вимоги:

- можливість пошуку документів за деякими критеріями;
- можливість перегляду всіх документів у системі;
- можливість перегляду детальної інформації документу;
- можливість створювання нового та редагування існуючого документа;
- можливість видалення документа;
- можливість завантаження до документа Word фалів;
- можливість завантаження з системи на свій ПК прикріплений файл до документа;
- можливість перегляду всіх контрагентів;
- можливість перегляду детальної інформації про контрагента та можливість змінювати інформацію про нього;
- можливість додавання нового та видалення старого контрагенту.

4.2 Вимоги до інтерфейсу

Вимоги до інтерфейсу системи документообігу:

- інтерфейс повинен бути в приємних кольорах;
- система повинна мати зручний інтерфейс;
- інтуїтивно зрозумілий для користувача;
- продуктивно використовувати простір екрану.

4.3 Умови виконання програми

Мінімальні системні вимоги та програмне забезпечення для функціонування серверної частини системи документообігу:

- продуктивний хостинг;
- вебсервер Gunicorn;
- підтримка Python та Django.

Для експлуатації клієнтської частини необхідні такі програмно-технічні засоби:

- процесор із тактовою частотою від 1,5 GHz;
- комп'ютер з графічною ОС;
- оперативної пам'яті від 2GB;
- підключення до Internet мережі.

4.4 Розгортання застосунку

Структура вебзастосунка на основі використання обраного Framework Django.

Архітектура серверу, на якому розгортається застосунок:

- операційна система Linux;
- веб-сервер Gunicorn;
- сервер бази даних MySQL;
- версія Python 3.11.2.

Структура вебзастосунка наведена на рис.4.1.

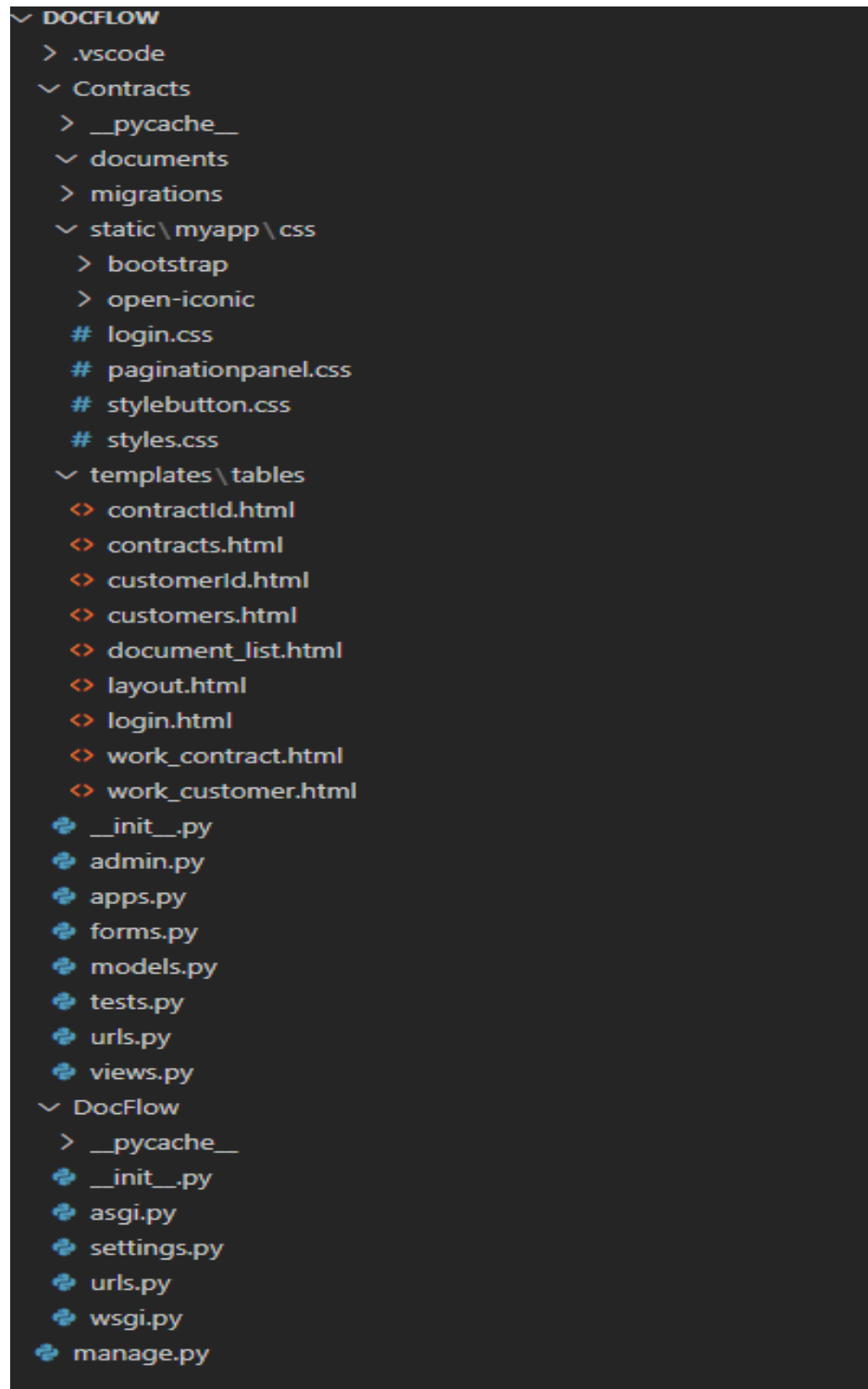


Рисунок 4.1 – Структура вебзастосунку

4.4.1 Звернення до програми

Звертання до програми від користувача організовано через відправлення запиту про перехід на сайт, використовуючи адресний рядок у веббраузері. Також звертання до програми відбувається, коли адміністратор змінює налаштування або виконує операції з користувачами.

4.4.2 Вхідні й вихідні дані

В якості вхідних даних виступає БД. Бд наведена на рис 4.2.

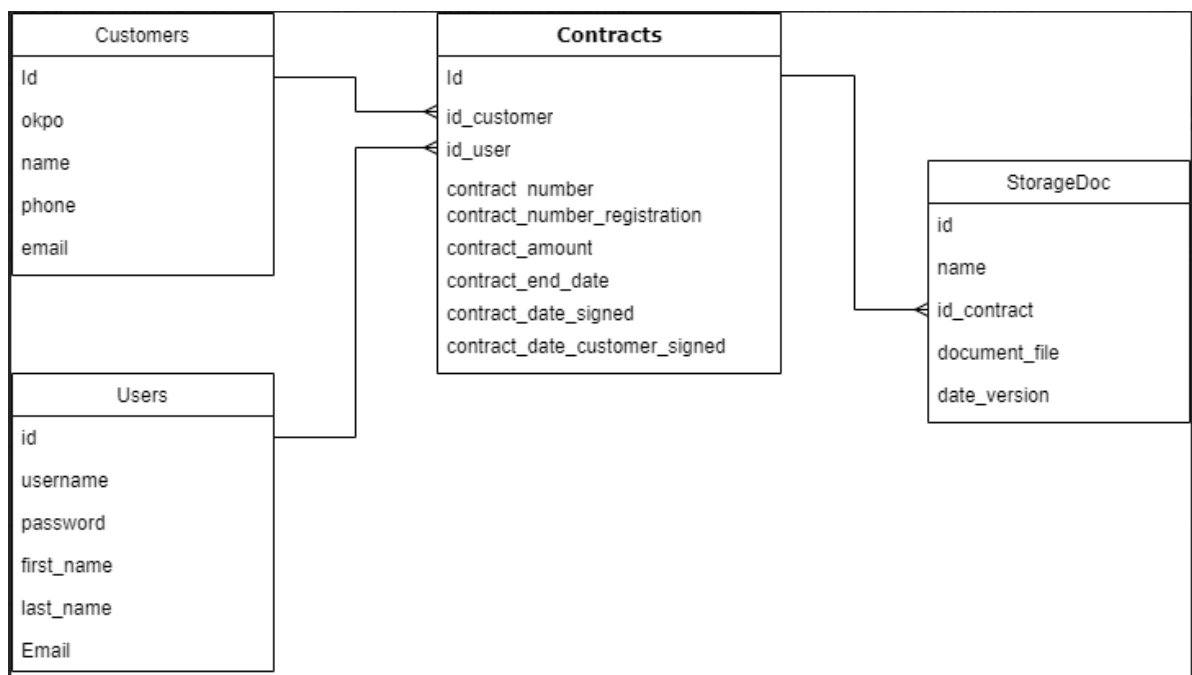


Рисунок 4.2 – База даних системи документообігу

4.5 Методика та результати тестування

Під час тестування було обрано стратегію поетапного виконання алгоритму користування програмою.

На рис. 4.3 наведено зовнішній вигляд головної сторінки.

№ контракту	Контрагент	Користувач	Реєстраційний номер	Ціна
№4320-453	ООО "Провайдер"	Сергей Главатских	№263	1518.15
№458934-3456	ООО "Побутовий магазин"	Зоряна Главатских	№364	18563.57
№75834-543	ООО "Хостингова компанія"	Сергей Главатских	№589	5748.54
№75934-463	ФОП "Довженко"	Данило Гарвона	№1283	5435.43
№114345-43	ФОП "Чайковский"	Сергей Главатских	№54	4234.59

Рисунок 4.3 – Зовнішній вигляд головної сторінки

На рис. 4.4 наведено сторінку детальної інформації договору.

№ контракту:	№4320-453
Контрагент	ООО "Провайдер"
Користувач	Сергей Главатских
№ реєстрації	№263
Ціна	1518.15
Строк дії	30-06-2023
Дата підписання	29-05-2023
Дата підписання контрагентом	27-05-2023
Документ	DocProvider.docx

[До переліку договорів](#)
[Редагувати](#)
[Видалити](#)
[Отримати весь список документів](#)

Рисунок 4.4 – Сторінку детальної інформації договору

На рис. 4.5 наведено зовнішній вигляд сторінки контрагентів.

Документообіг				Вихід	About
Головна Контрагенти Користувачі					
Створити					
ЄДРПОУ	Найменування	Телефон	Пошта		
38503254	ООО "Провайдер"	+3801788501	provaider@gmail.com		
56783294	ООО "Хостингова компанія"	+3802541881	host@gmail.com		
475983489	ФОП "Чайковський"	+38001651778	chaikov@gmail.com		
54385340	ФОП "Довженко"	+380164533	dovzhenko@gmail.com		
542347	ФОП "Іванов"	+3801861351	ivanov@gmail.com		
54385798	ООО "Побутовий магазин"	+3801681851	magazin@gmail.com		
5435654	ООО "Сонечко"	+380105414	sun@gmail.com		
4532545	ФОП "Дем'яненко"	+38015416	dem@gmail.com		

Рисунок 4.5 – Сторінка Контрагентів

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи бакалавра було реалізовано головну мету – програмну реалізацію системи документообігу. Для цього було виконано ряд завдань:

- виділити переваги та недоліки;
- сформулювати список вимог;
- виконати проєктування нового застосунку;
- виконати розробку та тестування застосунку.

Структура вебзастосунку була реалізована із використанням Python з Framework Django.

Був розроблений сценарій та проведено тестування вебзастосунку. Сценарій тестування був на основі основної логіки використання системи. Отриманий результат свідчить про позитивний результат.

В результаті виконання дипломної роботи було отримано автоматизовану систему, що відповідає всім поставленим вимогам, які були визначені на початку розробки та закріпленим в технічному завданні. Розроблена система вийшла зручною для використання користувачем та виконує всі поставлені функції, які були визначені на початку роботи .

Усі завдання дипломної кваліфікаційної роботи бакалавра повністю виконано.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Сутність та призначення електронного документообігу [Електрон. ресурс]. – Режим доступу: <https://sites.google.com/site/elektrdokumentoobig/sutnist-ta-priznacenna-elektronного-dokumentoobigu>.
2. Системи електронного документообігу в Україні – приклади СЕД систем [Електрон. ресурс]. – Режим доступу: <https://expresssoft.com.ua/uk/sistemi-elektronного-dokumentoobigu-vidi-prikladi-gotovi-rishennja/>.
3. Free Document Management Software [Electronic resource]. – Access mode: <https://www.opendocman.com/>.
4. About Alfresco Software [Electronic resource]. – Access mode: <https://www.alfresco.com/company/about-alfresco>.
5. KimiosDMS [Electronic resource]. – Access mode: <https://www.kimios.com/>.
6. Чому варто обрати Python [Електрон. ресурс]. – Режим доступу: <https://wezom.academy/ua/pochemu-stoit-vybrat-python/>.
7. Переваги і недоліки мови Python [Електрон. ресурс]. – Режим доступу: <https://blog.ithillel.ua/articles/perevagi-i-nedoliki-movi-python>.
8. Веб-фреймворк Django (Python) [Електрон. ресурс]. – Режим доступу: https://developer-mozilla-org.translate.goog/ru/docs/Learn/Server-side/Django?_x_tr_sl=ru&_x_tr_tl=uk&_x_tr_hl=uk&_x_tr_pto=sc.
9. Розробка сайтів на Python Django [Електрон. ресурс]. – Режим доступу: <https://wezom.com.ua/ua/blog/razrabotka-sajtov-na-python-django>.
10. Права та привілеї користувачів СУБД MySQL [Електрон. ресурс]. – Режим доступу: <https://vps.ua/wiki/ukr/mysql-privileges/>.
11. База даних MySQL [Електрон. ресурс]. – Режим доступу: <https://promoter.net.ua/articles/baza-danix-mysql.html>.

12. Концептуальне проектування баз даних [Електрон. ресурс]. – Режим доступу: <https://naurok.com.ua/konceptualne-proektuvannya-baz-danih-92407.html>.

13. Функціональні вимоги: приклади та шаблони [Електрон. ресурс]. – Режим доступу: <https://visuresolutions.com/uk/посібник-з-відстеження-управління-вимогами/функціональні-вимоги>.

14. Проектування інтерфейсу користувача, продуктивний дизайн, розроблення прототипу [Електрон. ресурс]. – Режим доступу: <http://kievoit.ippo.kubg.edu.ua/kievoit/2016/83/index1.html>.

15. Огляд процесу створення інтерфейсу [Електрон. ресурс]. – Режим доступу: <https://cases.media/article/oglyad-procesu-stvorenniya-interfeisu>.

16. Автоматизована система [Електрон. ресурс]. – Режим доступу: https://www.wik.uk-ua.nina.az/Автоматизована_система.html#cite_note-1.

ДОДАТОК А
Технічне завдання

Вступ

Область застосування програмного забезпечення – будь-яке підприємство, що потребує систему документообігу.

Тема для проєктування системи – програмна реалізація вебзастосунку «Система обліку договорів».

Умовне позначення теми розробки – система документообігу.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Програмна реалізація вебзастосунку «Система обліку договорів», затверджене наказом Національного університету «Запорізька політехніка» № 87 від 04 квітня 2022 р.

A.2 Основні вимоги до програми, що розробляється

Головним завданням роботи є створення системи, яка дозволяє користувачу вести документообіг у будь-якому підприємстві.

Система повинна інформувати користувача по існуючим документам, надавати можливість створювати нові, редагувати існуючі.

A.2.1 Вимоги до функціональних характеристик

Функціональні вимоги:

- можливість пошуку документів за деякими критеріями;
- можливість перегляду всіх документів у системі;
- можливість перегляду детальної інформації документу;
- можливість створення нового та редагування існуючого документа;

- можливість видалення документа;
- можливість завантаження до документа Word фалів;
- можливість завантаження з системи на свій ПК прикріплений файл до документу;
- можливість перегляду всіх контрагентів;
- можливість перегляду детальної інформації про контрагента та можливість змінювати інформацію про нього;
- можливість додавання нового та видалення старого контрагенту.

A.2.2 Вимоги до інтерфейсу програми

Вимоги до інтерфейсу системи документообігу:

- інтерфейс повинен бути в приємних кольорах;
- система повинна мати зручний інтерфейс;
- інтуїтивно зрозумілий для користувача;
- продуктивно використовувати простір екрану.

Макет інтерфейсу наведено на рисунку A.1.

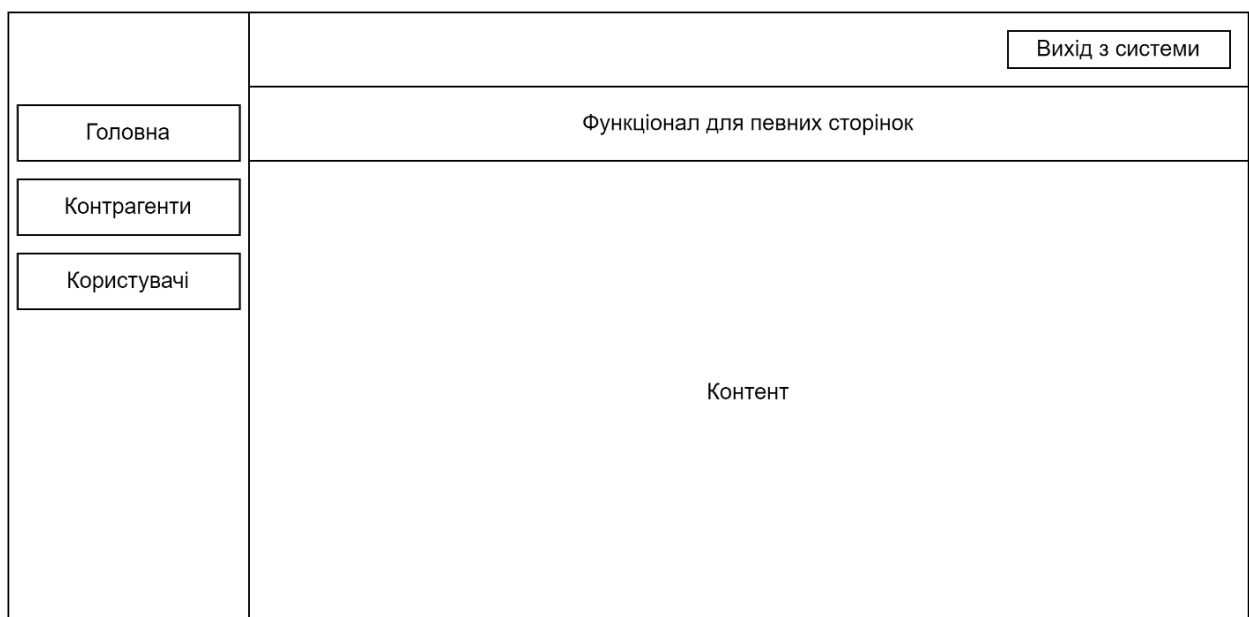


Рисунок A.1 – Макет інтерфейсу

A.2.3 Вимоги до складу та параметрів технічний засобів

Для експлуатації клієнтської частини необхідні такі програмно-технічні засоби:

- процесор із тактовою частотою від 1,5 GHz;
- комп'ютер з графічною ОС;
- оперативної пам'яті від 2GB;
- підключення до Internet мережі.

A.3 Вхідні дані до роботи

В якості вхідних даних виступають дані, які надходять від віддаленого клієнта про запитувану сторінку або про виконання деяких дій, а вихідними даними є результати обробки запитів клієнта і формування вебсторінки.

ДОДАТОК Б
Текст програми

Б.1 Перший файл views.py

```

from django.contrib.auth import authenticate, login, logout
from django.http import HttpResponseRedirect,
FileResponse
from django.shortcuts import render, get_object_or_404,
redirect
from .models import Contracts, Customers, StorageDoc
from django.contrib.auth.models import User
from .forms import CustomerForm, ContractForm,
StorageDocForm
from django.urls import reverse
from django.core.paginator import Paginator
from django.db.models import Q

# Create your views here.
def login_view(request):
    if request.method == "POST":
        username = request.POST["username"]
        password = request.POST["password"]
        user = authenticate(request, username=username,
password=password)
        if user is not None:
            login(request, user)
            return HttpResponseRedirect(reverse("home"))
        else:
            return render(request, "tables/login.html", {
                "message": "Invalid credentials."
            })
    return render(request, "tables/login.html")

def logout_view(request):
    logout(request)
    return render(request, "tables/login.html", {
        "message": "Logged out."
    })

```

```

    })

def contracts(request):
    if not request.user.is_authenticated:
        return HttpResponseRedirect(reverse("login"))
    rows_count = request.GET.get('p', 5)
    query = request.GET.get("q")
    if query:
        post_list=Contracts.objects.filter(
            Q(id_customer__name__contains=query) |
            Q(contract_number__contains=query ))
        paginator = Paginator(post_list, rows_count)
    else:
        post_list = Contracts.objects.all()
        paginator = Paginator(post_list, rows_count)
    page_number = request.GET.get('page')
    page_obj = paginator.get_page(page_number)
    return render(request, "tables/contracts.html", {
        "contracts": page_obj, "search": bool(query),
        "query": query
    })

def customers(request):
    return render(request, "tables/customers.html", {
        "customers": Customers.objects.all()
    })

def users(request):
    return render(request, "tables/users.html", {
        "users": User.objects.all()
    })

def contractId(request, id):
    contract = get_object_or_404(Contracts, id=id)
    file = StorageDoc.objects.filter(id_contract=id).order_by("-
    date_version")

```

```

        if file.exists():
            return render(request, 'tables/contractId.html', {
                'contract': contract, 'file': file[0],
                'file_exists': True})
        else:
            # file_document =
            file.objects.annotate(days_diff=F('date_version' -
            date.today()).filter(days_diff__gte=0).order_by('days_diff').first()

            return render(request, 'tables/contractId.html', {
                'contract': contract, 'file': file, 'file_exists':
False
            })
    def list_of_documents(request, id):
        document_list =
StorageDoc.objects.filter(id_contract=id)
        return render(request, "tables/document_list.html", {
            "document_list": document_list, 'contract_id':id
        })
    def download_file(request, id):
        # Отримайте файл, який потрібно завантажити
        document = get_object_or_404(StorageDoc, id=id)
        file = document.document_file
        # Створіть HTTP-відповідь з файлом
        response =
HttpResponse(content_type='application/vnd.openxmlformats-
officedocument.wordprocessingml.document')
        response['Content-Disposition'] = 'attachment;
filename=' + document.name
        response.write(file)
        return response
    def customerId(request, id):
        customer = get_object_or_404(Customers, id=id)
        return render(request, 'tables/customerId.html', {
            'customer': customer

```

```

    })

def edit_customer(request, id):
    customer = get_object_or_404(Customers, id=id)
    if request.method == 'POST':
        form = CustomerForm(request.POST, instance=customer)
        if form.is_valid():
            form.save()
            return redirect('customerId', id=customer.id)
    else:
        form = CustomerForm(instance=customer)
        value = False
    return render(request, 'tables/work_customer.html',
{'form': form, 'value': value, 'custmid': id})

def new_customer(request):
    if request.method == 'POST':
        form = CustomerForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('customers')
    else:
        form = CustomerForm()
        value = True
    return render(request, 'tables/work_customer.html',
{'form': form, 'value': value})

def delete_customer(request, id):
    record = Customers.objects.get(id=id)
    # Удаляем запись из базы данных
    record.delete()
    # Перенаправляем пользователя на другую страницу
    return redirect('customers')

def new_contract(request):
    if request.method == 'POST':
        form_contract = ContractForm(request.POST)

```

```

        form_file = StorageDocForm(request.POST,
request.FILES)
        if form_contract.is_valid() and
form_file.is_valid():
            contract = form_contract.save()
            # file = form_file.save(commit=False)
            if form_file.instance.document_file != None:
                file =
StorageDoc.objects.create(document_file=request.FILES["document_f
ile"].read())
                file.id_contract = contract
                file.name =
form_file.instance.document_file.name
                file.save()
            return redirect('contracts')
        else:
            form_contract = ContractForm()
            form_file = StorageDocForm()
            value=True
            return render(request, 'tables/work_contract.html',
{'form_contract': form_contract, 'form_file': form_file, 'value':
value})

def edit_contract(request, id):
    contractId = get_object_or_404(Contracts, id=id)
    # fileId = get_object_or_404(StorageDoc, id_contract=id)
    fileId =
StorageDoc.objects.filter(id_contract=id).order_by("date_version"
).first()
    if request.method == 'POST':
        form_contract = ContractForm(request.POST,
instance=contractId)
        form_file = StorageDocForm(request.POST,
request.FILES, instance=fileId)
        if form_contract.is_valid() and
form_file.is_valid():

```

```

        contract = form_contract.save()
        if "document_file" in form_file.changed_data:
            file =
StorageDoc.objects.create(document_file=request.FILES["document_f
ile"].read())

            file.id_contract = contract
            file.name =
form_file.instance.document_file.name
            file.save()
            return redirect('contractId', id=contractId.id)
    else:
        form_contract = ContractForm(instance=contractId)
        if fileId != None:
            form_file = StorageDocForm(instance=fileId)
        else:
            form_file = StorageDocForm()
        value=False
        return render(request, 'tables/work_contract.html',
{'form_contract': form_contract, 'form_file': form_file, 'value':
value, 'contrid': id})

def delete_contract(request, id):
    record = Contracts.objects.get(id=id)
    # Удаляем запись из базы данных
    record.delete()
    # Перенаправляем пользователя на другую страницу
    return redirect('contracts')

```

Б.2 Другой файл forms.py

```

from django import forms
from .models import Customers, Contracts, StorageDoc
from django.contrib.auth.models import User

class CustomerForm(forms.ModelForm):

```

```

        okpo =
forms.CharField(widget=forms.TextInput(attrs={"class":"form-
control"}), label = 'ЄДРПОУ')

        name =
forms.CharField(widget=forms.TextInput(attrs={"class":"form-
control"}), label = 'Найменування')

        phone =
forms.CharField(widget=forms.TextInput(attrs={"class":"form-
control"}), label = 'Телефон')

        email =
forms.EmailField(widget=forms.TextInput(attrs={"class":"form-
control"}), label = 'Пошта')

        class Meta:
            model = Customers
            fields = ['okpo', 'name', 'phone', 'email']

class ContractForm(forms.ModelForm):
    id_customer =
forms.ModelChoiceField(queryset=Customers.objects.all(), label =
'Контрагент')
    id_user = forms.ModelChoiceField(queryset=User.objects.all(),
label = 'Користувач')
    contract_number =
forms.CharField(widget=forms.TextInput(attrs={"class":"form-
control"}), label = 'Номер документа')
    contract_number_registration =
forms.CharField(widget=forms.TextInput(attrs={"class":"form-
control"}), label = 'Номер реєстрації')
    contract_amount = forms.FloatField(label = 'Вартість')
    contract_end_date =
forms.DateField(widget=forms.DateInput(attrs={'type': 'date'}),
label = 'Термін закінчення договору')
    contract_date_signed =
forms.DateField(widget=forms.DateInput(attrs={'type': 'date'}),
label = 'Дата підписання')

```

```

        contract_date_customer_signed =
forms.DateField(widget=forms.DateInput(attrs={'type': 'date'}),
label = 'Дата підписання контрагентом')

class Meta:
    model = Contracts
    fields = ['id_customer', 'id_user', 'contract_number',
'contract_number_registration', 'contract_amount',
'contract_end_date',
            'contract_date_signed',
'contract_date_customer_signed']

class StorageDocForm(forms.ModelForm):
    document_file = forms.FileField(label = 'Документ',
required=False)
    class Meta:
        model = StorageDoc
        fields = ['document_file']

```

Б.3 Третій файл models.py

```

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

# Create your models here.

#Table Customer
class Customers(models.Model):
    okpo = models.IntegerField(null=True, blank=False)
    name = models.CharField(max_length=100, null=True,
blank=False)
    phone = models.CharField(max_length=100, null=True,
blank=False)
    email = models.EmailField(null=True, blank=True)

```

```

    def __str__(self):
        return self.name

def getpath_upload_to(instance, filename):
    return "documents/{}/{}".format(instance.id, filename)

#Table Contracts
class Contracts(models.Model):
    id_customer = models.ForeignKey(Customers,
on_delete=models.CASCADE, null=True, blank=False)
    id_user = models.ForeignKey(User, on_delete=models.CASCADE,
null=True, blank=False)
    contract_number = models.CharField(max_length=100, null=True,
blank=False)
    contract_number_registration =
models.CharField(max_length=100, null=True, blank=False)
    contract_amount = models.FloatField(null=True, blank=True)
    contract_end_date = models.DateField(null=True, blank=True)
    contract_date_signed = models.DateField(null=True, blank=True)
    contract_date_customer_signed = models.DateField(null=True,
blank=True)

    def __str__(self):
        return self.contract_number

#Table StorageDoc
class StorageDoc(models.Model):
    name = models.CharField(max_length=100, null=True,
blank=False)
    id_contract = models.ForeignKey(Contracts,
on_delete=models.CASCADE, null=True, blank=True)
    document_file = models.BinaryField(editable = True,
null=True)
    date_version = models.DateField(null=True, auto_now=True)

```

ДОДАТОК В
Слайди презентації

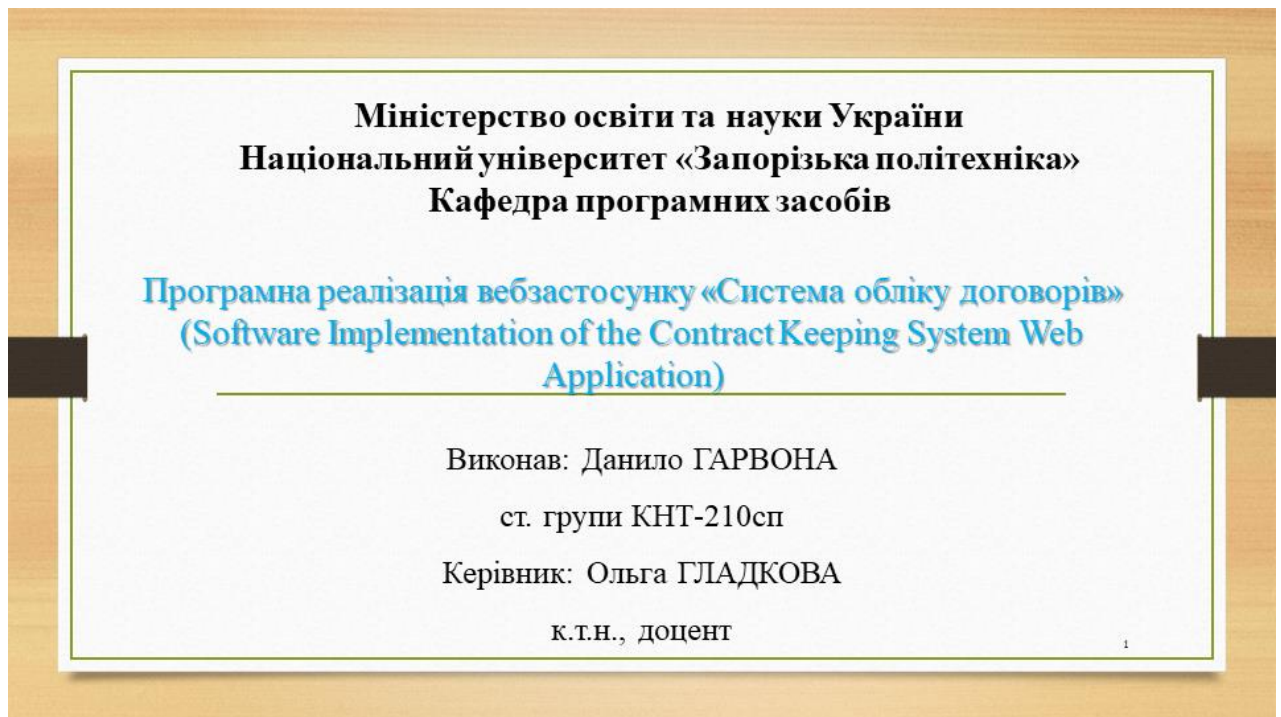


Рисунок В.1 – Слайд 1

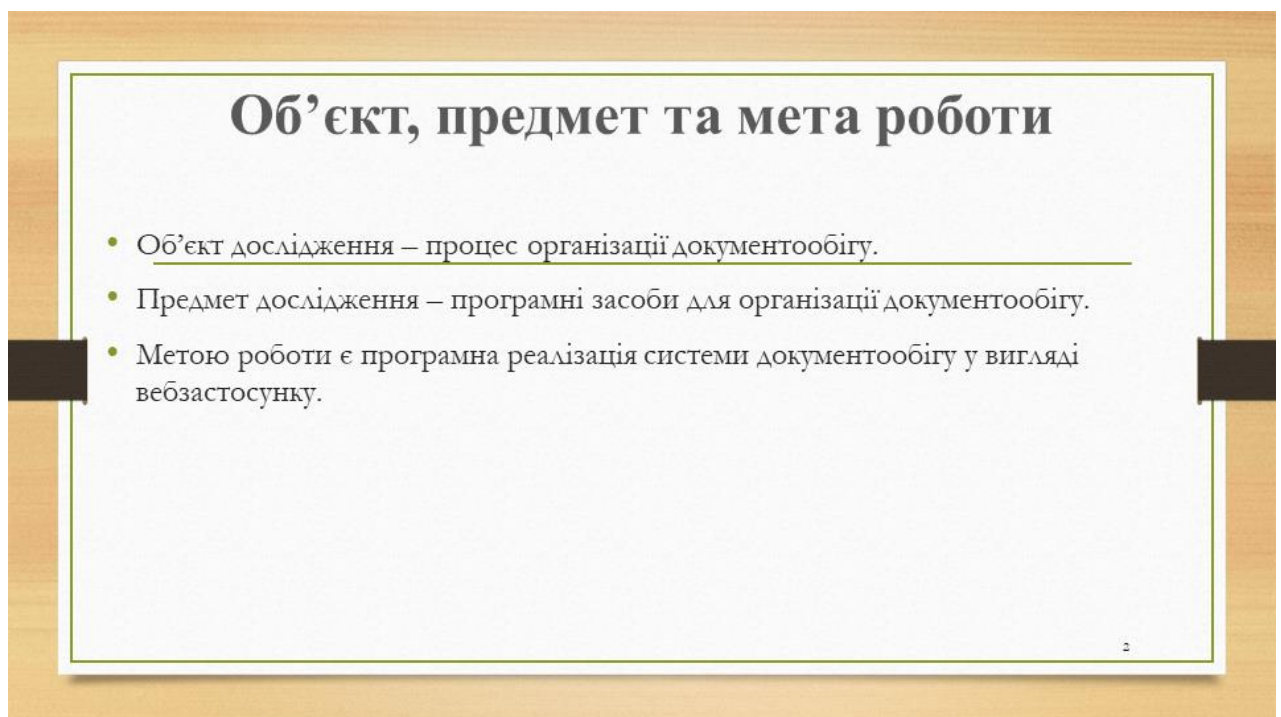


Рисунок В.2 – Слайд 2

Порівняння існуючих аналогів

Критерій	Alfresco CS	OpenDocMan	Kimios	Розроблюваний продукт
Безкоштовний	-	-	-	+
Зручність інтерфейсу	+	-	+	+
Вимоги до системи	+	+	+	+
Контроль документів	+	+	+	+
Швидкість навчання роботи	-	+	+	+
Робота з хмарними технологіями	+	+	+	+

3

Рисунок В.3 – Слайд 3

Постановка завдання

Головним завданням роботи є створення системи, яка дозволяє користувачу здійснити облік договорів на підприємстві.

Система повинна показувати всі договори, які існують. Створювати нові та змінювати вже існуючі. Також, додавання к договору документу Word та отримання різних версій змін цього документу.

4

Рисунок В.4 – Слайд 4

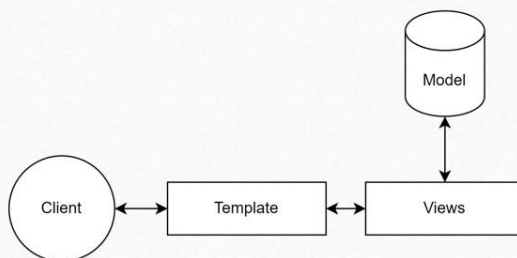
Порівняння мов програмування			
Критерій	PHP	Python	ASP.NET
Надбудови для структуризації розробки	Багато	Багато	Мало
Серверна підтримка	Розповсюджена	Розповсюджена	Недостатньо розповсюджена
Масштабованість розробок	Проста	Проста	Проста
Складність підтримки	Низька	Середня	Середня

Рисунок В.5 – Слайд 5

Порівняння Framework		
Критерії	Django	Symfony
Мова програмування	Python	PHP
Популярність	+	+
Архітектура	MVT	MVC
Швидкодія	Відомий своєю швидкістю та ефективністю. Має добрі механізми кешування та оптимізації запитів.	Також ефективний фреймворк, проте може вимагати більше ресурсів сервера для виконання.
Розширюваність та пакети	+	+
Фокус на певних завданнях	Зосереджений на швидкій розробці веб-додатків з підтримкою адміністративного інтерфейсу.	Забезпечує розробку складних та масштабованих додатків з фокусом на розширюваність.

Рисунок В.6 – Слайд 6

Загальна схема застосунку



7

Рисунок В.7 – Слайд 7

Просто почніть вводити
тут текст, щоб
переглянути функції,
знайти потрібний вміст
або отримати довідку.

Докладніше

Структура веб-застосунку

```

DocFlow
├── .vscode
├── Contracts
├── __pycache__
├── documents
├── migrations
├── static\myapp\css
├── bootstrap
├── open-iconic
├── login.css
├── paginationpanel.css
├── stylebutton.css
├── styles.css
├── templates\tables
├── contractid.html
├── contracts.html
├── customerid.html
├── customers.html
├── document_list.html
├── layout.html
├── login.html
├── work_contract.html
├── work_customer.html
├── __init__.py
├── admin.py
├── apps.py
├── forms.py
├── models.py
├── tests.py
├── urls.py
├── views.py
├── __pycache__
├── __init__.py
├── asgi.py
├── settings.py
├── urls.py
├── wsgi.py
└── manage.py
  
```

8

Рисунок В.8 – Слайд 8



Рисунок В.9 – Слайд 9

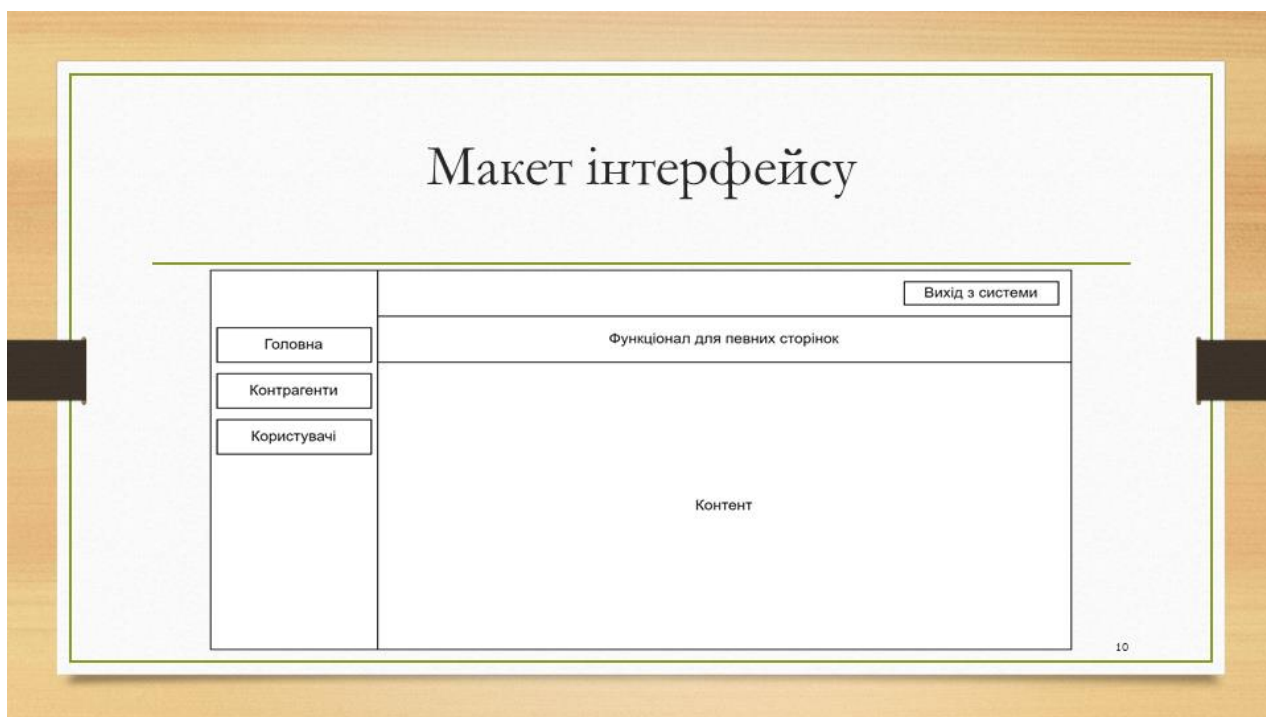


Рисунок В.10 – Слайд 10

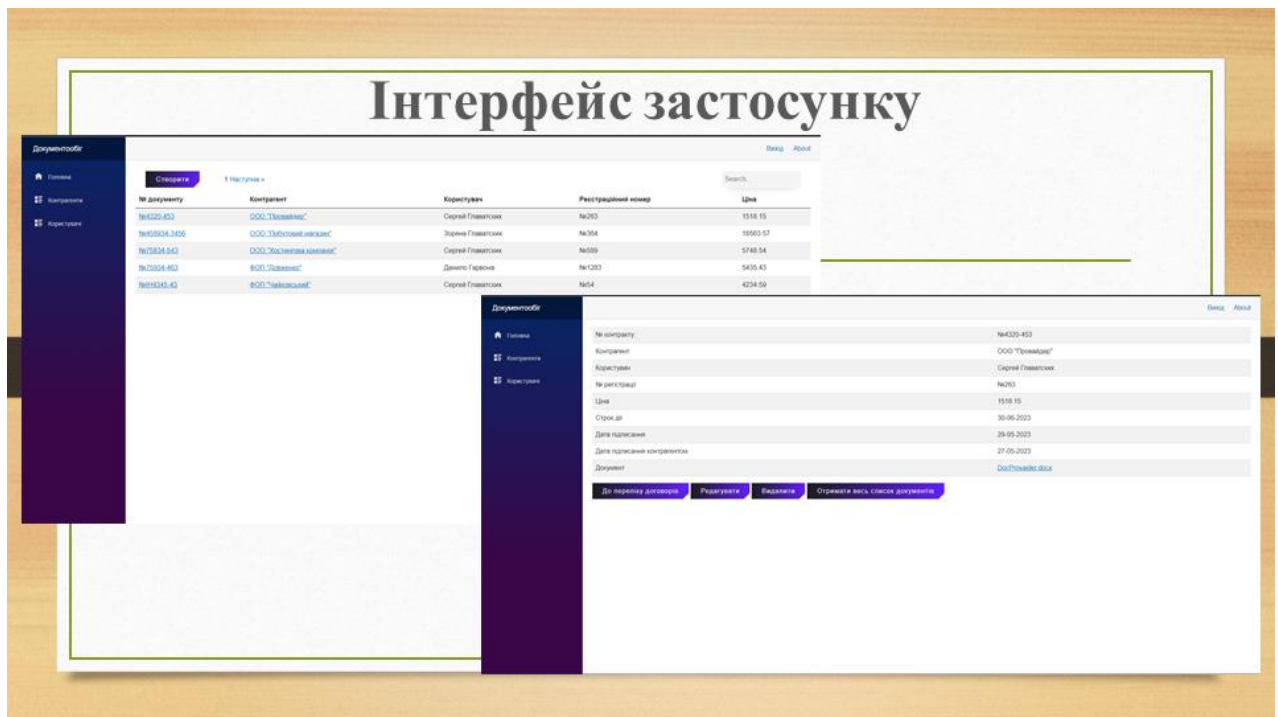


Рисунок В.11 – Слайд 11

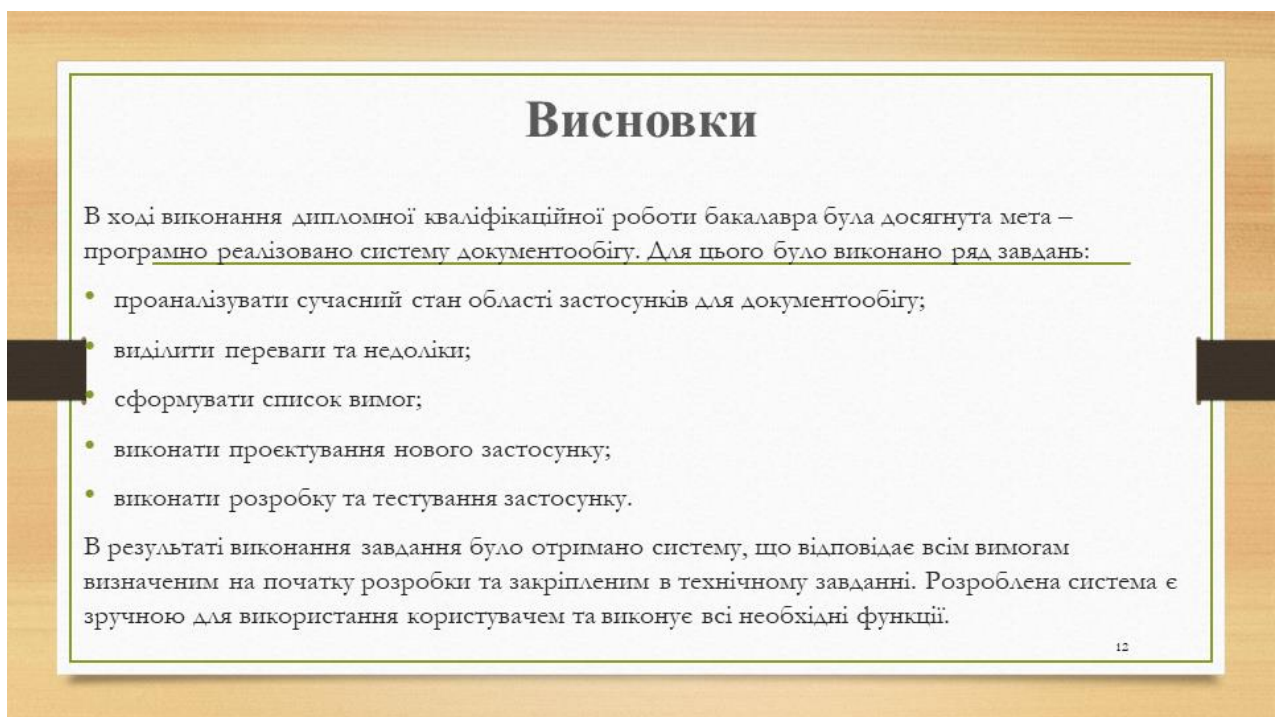


Рисунок В.12 – Слайд 12