Київський національний торговельно-економічний університет

Кафедра комп'ютерних наук та інформаційних систем

ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЕКТ

на тему:

«Автоматизація управління документообігом на підприємстві»

Студентки 4 курсу, 9 групи,

спеціальності
122 «Комп'ютерні науки»

_____
підпис студента

Кожухар
Олександри
Володимирівни

Науковий керівник
Доктор технічних наук, професор

_____
підпис керівника

Краскевич Валерій
Євгенович

Гарант освітньої програми
кандидат технічних наук, доцент

_____
підпис керівника

Демідов Павло
Георгійович

Київ 2020

Київський національний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра комп'ютерних наук та інформаційних систем
Спеціальність 122 «Комп'ютерні науки»

Затверджую
Зав. кафедри _____Пурський О.І.
«20» грудня 2019р.

Завдання
на випускний кваліфікаційний проект (роботу) студентці

Кожухар Олександрі Володимирівні
(прізвище, ім'я, по батькові)

1. Тема випускного кваліфікаційного проекту (роботи)
«Автоматизація управління документообігом на підприємстві»
Затверджена наказом ректора від *«04» грудня 2019 р. № 4111*
2. Строк здачі студентом закінченого проекту *12 червня 2020 року*
3. Цільова установка та вихідні дані до проекту
Мета проекту: аналіз та розробка системи для автоматизації подачі звітності на умовному підприємстві.
Об'єкт дослідження: автоматизація подачі звітів на умовному підприємстві.
Предмет дослідження: є внутрішня організація документообігу та засоби автоматизації його процедур.
4. Перелік графічного матеріалу_____

_____

_____

5. Консультанти по проекту із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Пурський О.І.	15.12.2019 р.	15.12.2019 р.
2	Пурський О.І..	15.12.2019 р.	15.12.2019 р.
3	Пурський О.І.	15.12.2019 р.	15.12.2019 р.

6. Зміст випускного кваліфікаційного проекту (роботи) (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. Теоретичні аспекти розробки сучасних систем електронного документообігу

1.1.Поняття і сутність системи електронного документообігу

1.2. Види систем електронного документообігу

1.3. Сучасні підходи до автоматизації документообігу на підприємстві

1.4. Нормативно-правове регулювання діловодства та документообігу в Україні

РОЗДІЛ 2. Огляд технологій створення систем для електронного документообігу

2.1. Метод технології Python Django

2.2. Метод технології Python Django REST Framework

2.3. Метод технології JavaScript Angular

2.4. Метод технології JavaScript React

2.5. Метод технології MongoDB

2.6. Метод технології PostgreSQL

РОЗДІЛ 3. Створення системи автоматизації документообігу

3.1. Вимоги до структури та функціонування системи

3.2. Основні модулі системи та їх прототипи

3.3. Розробка серверної частини системи

3.4. Розробка користувацького інтерфейсу

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

7. Календарний план виконання проекту

№ Пор.	Назва етапів випускного кваліфікаційного проекту	Строк виконання етапів проекту	
		За планом	фактично
1	2	3	4
1	*Вибір теми випускного кваліфікаційного проекту*	*01.10.2019*	*01.10.2019*
2	*Розробка та затвердження завдання на випускний кваліфікаційний проект*	*15.12.2019*	*15.12.2019*
3	*Вступ*	*03.02.2020*	
4	*РОЗДІЛ 1. Теоретичні аспекти розробки сучасних систем електронного документообігу*	*28.02.2020*	
5	*РОЗДІЛ 2. Огляд технологій створення системи для електронного документообігу*	*06.04.2020*	
6	*РОЗДІЛ 3. Створення системи автоматизації документообігу*	*12.05.2020*	
7	*Висновки*	*15.05.2020*	
8	*Здача випускного кваліфікаційного проекту на кафедру науковому керівнику*	*20.05.2020*	
9	*Попередній захист випускного кваліфікаційного проекту*	*03.06.2020*	
11	*Виправлення зауважень, зовнішнє рецензування випускного кваліфікаційного проекту*	*09.06.2020*	
12	*Представлення готового зшитого випускного кваліфікаційного проекту на кафедру*	*12.06.2020*	
13	*Публічний захист випускного кваліфікаційного проекту*	*За розкладом роботи ЕК*	

8. Дата видачі завдання «15» грудня 2019 р.

Керівник випускного кваліфікаційного проекту (роботи)
Краскевич В.Є.
(прізвище, ініціали, підпис)

Гарант освітньої програми Демідов П.Г.
(прізвище, ініціали, підпис)

Завдання прийняв до виконання студент-дипломник
Кожухар О.В.
(прізвище, ініціали, підпис)

9. Відгук керівника випускного кваліфікаційного проекту (роботи)

____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____

Керівник випускного кваліфікаційного проекту (роботи)

________________ ________ р.

(підпис, дата)

10. Висновок про випускний кваліфікаційний проект (роботу)

Випускний кваліфікаційний проект (робота) студента ____________________
(прізвище, ініціали)
може бути допущена до захисту в екзаменаційній комісії.

Гарант освітньої програми________________________Демідов П.Г._____
(підпис, прізвище, ініціали)

Завідувач кафедри ______________________________Пурський О.І.____
(підпис, прізвище, ініціали)

«_____»________________2020 р.

Анотація

У випускному кваліфікаційному проекті розглянуто поняття електронного документу, електронного документообігу та його нормативно-правову базу. Проведений аналіз сучасних підходів до автоматизації документообігу на підприємстві, наведено порівняльну характеристику систем електронного документообігу на українському ринку. Розглянуто основні технології для розробки комплексних інтеграційних та модульних веб-додатків. Узагальнено вимоги до систем автоматизації документообігу та принципи їх побудови. Розроблено архітектуру, серверну та клієнтську частини системи автоматизації документообігу на підприємстві.

Ключові слова: електронний документ**,** електронний документообіг, система електронного документообігу, прикладний програмний інтерфейс.

Annotation

The graduation qualification project is devoted to consider the concept of electronic document, electronic workflows and its legal framework. The analysis of modern approaches to automation of electronic workflows at the enterprise is carried out, the comparative characteristic of electronic document management systems in the Ukrainian market is resulted. The main technologies for the development of complex integration and modular web-applications are considered. The requirements to document automation systems and the principles of their construction are generalized. The architecture, server and client parts of the document automation system at the enterprise have been developed.

Keywords: electronic document, electronic workflows, electronic document management system, application programming interface

ЗМІСТ

ВСТУП

Сучасний світ важко уявити без інформаційних технологій, які використовуються у всіх сферах діяльності людини: освіта і наука, сфера торгівлі і послуг, промисловості і охороні здоров'я тощо.

Завдяки швидкому розвитку інформаційних технологій сфера бухгалтерського обліку зазнає чимало позитивних змін під впливом створення електронного документообігу та активному використанню електронних документів для реєстрації господарських операцій.

Електронний документообіг є однією з найважливіших складових будь-якої системи, яка має за основу автоматизацію підприємства, оскільки ефективність будь-якого підприємства, в більшій мірі, залежить від інформації. Сьогодні для кожної організації є пріоритетом накопичення інформації і швидке прийняття управлінських рішень. Саме тому доступ до інформації має бути зручним і оперативним.

В сучасному світі з ринковою економікою, напевно, не знайдеться жодної компанії, яка не працює з документами. Документи використовуються скрізь, починаючи зі статуту компанії і закінчуючи накладними та ордерами. При цьому використовуються, в більшій мірі, паперові носії, незважаючи на те, що всі тексти і форми попередньо друкуються на ПК. Це призводить до додаткових витрат компанії на папір, оргтехніку і канцелярію.

В Україні, малий бізнес, як правило, користується традиційним документообігом, без залучення засобів автоматизації. Це є логічним, оскільки при малому обороті документів в компанії не виникає необхідності в автоматизації, так як розробка або покупка системи автоматизації документообігу коштуватиме досить дорого.

Але з ростом і розвитком компанії, коли процес документообігу стає складнішим і інтенсивнішим, стає складно вчасно заповнювати і підписувати документи. До того ж накопичення паперових носіїв може призвести до їх втрати або псування.

Таким чином, керівництво компанії починає задумуватись про

систематизацію існуючих документів і створення ефективного механізму їх обороту в компанії. Саме тоді вони починають аналізувати і порівнювати багаточисельні системи автоматизації управління, які є на українському та світовому ринку.

Тому, **метою даного дипломного проекту** є аналіз та розробка системи для автоматизації подачі звітності на умовному підприємстві. Реалізація поставленої мети передбачає необхідність вирішення наступних **завдань**:

- Аналіз існуючих систем електронного документообігу, їх видів та властивостей;
- Дослідження нормативно-правової бази документообігу в Україні;
- Визначення функцій, які необхідно автоматизувати на підприємстві;
- Визначення конкретних вимог до автоматизованої системи;
- Вибір технологій та проектування архітектури;
- Розробка серверної архітектури системи;
- Розробка інтерфейсу користувача;
- Висновки по результатам виконаного проекту.

Об'єктом дослідження дипломного проекту є автоматизація подачі звітів на умовному підприємстві.

Предметом дослідження є внутрішня організація документообігу та засоби автоматизації його процедур.

Методи дослідження є обумовленими об'єктом і предметом дипломного проекту. Для успішного досягнення мети та завдань дипломного проекту застосовано комплекс таких методів, як:

- Методи системного аналізу;
- Методи узагальнення;
- Методи порівняння;
- Методи прямого структурного аналізу;
- Методи причинно-наслідкового аналізу;
- Індукція і дедукція.

Практичне значення даного дипломного проекту, перш за все, полягає в тому, що в дослідженні викладений всесторонньо проаналізований та систематизований матеріал про сучасний стан електронного документообігу в Україні, основні характеристики, властивості та вимоги до систем автоматизації. Окрім того, представлений список літератури може бути використаний для подальшого дослідження по даній темі.

РОЗДІЛ 1.
ТЕОРЕТИЧНІ АСПЕКТИ РОЗРОБКИ СУЧАСНИХ СИСТЕМ ЕЛЕКТРОННОГО ДОКУМЕНТООБІГУ

1.1.Поняття і сутність системи електронного документообігу

Документ є основним носієм інформації, на основі якого приймаються рішення щодо подальшого функціонування та напрямку розвитку підприємства, здійснюється контроль операцій та рух цінностей. Для кожної компанії характерним є постійний рух документів, незалежно від її розмірів.

Статтею 1 Закону України “Про інформацію” [1] визначено, що **документ** – це матеріальний носій, що містить інформацію, основними функціями якого є її збереження та передавання у часі та просторі.

На сьогоднішній день, очевидними є такі проблеми пов’язані з документами на підприємстві [2]:

- документи губляться;
- накопичується безліч документів, призначення і джерела яких незрозумілі;
- до інформації можуть мати доступ сторонні особи;
- значні витрати часу на пошук потрібних документів та формування їх у тематичні добірки;
- створення декількох копій одного і того ж документа – на папір та копіювання витрачаються додаткові кошти;
- значні витрати часу на підготовку та узгодження документів, а також їх доставку адресатам тощо.

Впровадження практики електронних документів та електронного документообігу на підприємстві дозволяє вирішити майже всі вищевказаній проблеми або звести ймовірність їх виникнення до мінімуму.

Відповідно до ст. 5 Закону України “Про електронні документи та електронний документообіг” [3] визначено, що **електронний документ** (ЕД) –

це документ, інформація в якому зафіксована у вигляді електронних даних, включаючи обов'язкові реквізити документа, склад та порядок розміщення яких визначається законодавством.

У статті 9 Закону України "Про електронні документи та електронний документообіг" [3] вказано, що **електронний документообіг** (обіг електронних документів) - сукупність процесів створення, оброблення, відправлення, передавання, одержання, зберігання, використання та знищення електронних документів, які виконуються із застосуванням перевірки цілісності та у разі необхідності з підтвердженням факту одержання таких документів.

Основні принципи електронного документообігу [4]:

- однократна реєстрація документа, що дозволяє однозначно його ідентифікувати у будь-якій підсистемі;
- можливість паралельного виконання різних операцій з метою скорочення часу руху документів і підвищення оперативності їх виконання;
- безперервність руху документа;
- єдина база документної інформації для централізованого зберігання документів і виключення можливості дублювання документів;
- ефективно організована система пошуку документа.

Слід взяти до уваги, що поняття "система електронного документообігу" має різні тлумачення в українській науковій літературі, європейських стандартах та англомовній ІТ-літературі, тобто не є "загальновизначеним".

Натомість, в Законах України ми можемо знайти унормоване визначення таких понять, як "електронний документ" та "електронний документообіг" [2].

У межах даного дипломного проекту, ми будемо використовувати таке визначення: "Система електронного документообігу (СЕД) – організаційно-технічна система, яка забезпечує процес створення, управління доступом і розповсюдження електронних документів у комп'ютерних мережах, а також забезпечує контроль над потоками документів в організації" [5].

Головною метою СЕД є створення механізму збереження електронних документів, зручного доступу до них, пошуку по атрибутам і змісту. СЕД

повинна відслідковувати створення і внесення змін в документи, дотримання регламентів подачі документів, збереження ланцюгу погодження кожного документа, забезпечувати систематизоване зберігання інформації, розмежовувати права доступу користувачів за організаційно-штатною структурою, а також давати можливість масштабування системи в подальшому.

До основних вимог щодо СЕД відносять:

- Масштабованість. Система повинна підтримувати необхідну кількість користувачів, без проблем інтегруватися з корпоративним ПО підприємства.
- Розподіленість. Архітектура СЕД повинна підтримувати роботу з документами в організаціях розподілених територіально.
- Модульність. Система має складатись з незалежних модулів, які повністю інтегровані між собою, оскільки часто користувачу не потрібен весь функціонал системи.
- Відкритість. В системі повинні бути відкриті інтерфейси для подальшого доопрацювання і інтеграції з іншими системами.

1.2. Види систем електронного документообігу

Завдання автоматизації документообігу на підприємстві можна вирішити за допомогою таких програмних засобів:

- Засобів групової роботи;
- Систем управління особливими видами документів (зокрема, PDM-систем);
- Спеціальних модулів управління документообігом;
- Універсальних систем електронного документообігу.

Засоби групової роботи (Lotus Notes, MS Exchange, Novell GroupWise) націлені на забезпечення ефективної взаємодії користувачів. Як правило, це розширення програм електронної пошти і часто не підходить для роботи з крупними архівами документів.

Системи управління особливими групами документів розділяють на дві умовні підгрупи:

- Вузькопрофільне програмне забезпечення, що не має засобів інтеграції з іншими корпоративними та зовнішніми системами. Як правило, такі системи представлені місцевими розробниками підприємства. Витрати на підтримку таких систем часто не виправдовують себе.
- Спеціалізовані системи, що призначені для управління виробничою інформацією і що мають засоби інтеграції з іншими системами.

Спеціальні модулі управління документообігом входять до складу більшості корпоративних інформаційних систем (ІС). Як правило, модулі таких систем достатньо обмежені, оскільки майже неможливо створити універсальну ІС. До того ж їх ціна дуже велика, тому далеко не кожне підприємство може собі це дозволити.

Універсальні системи електронного документообігу надають можливість комплексні рішення великої кількості завдань з управління документами. Більш того, їх впровадження для компаній буде набагато дешевшим і ефективнішим.

Український ринок систем електронного документообігу представлений широким спектром програмних засобів. Серед найпопулярніших можемо виділити: «M.E.Doc», «СОТА», «FREDO ДокМен», «Вчасно», «FlyDoc». Саме їх ми обрали для подальшого порівняльного аналізу (табл. 1.1).

Усі системи електронного документообігу, які порівнювались, належать до класу систем електронного управління документами. В усіх системах присутні можливості працювати в одному обліковому записі одразу з декількома підприємствами, підтримка колективної роботи користувачів, імпорту документів з облікових систем, створення первинних документів за існуючими шаблонам безпосередньо в програмі, автоматичної перевірки наявності помилок у документах, контролю статусу документів (відправлено, прийнято, відхиллено, заблоковано), пошуку документів за встановленими фільтрами (параметрами), збереження документів на хмарі, підтримка змін

законодавства, інтеграції з іншими програмами, електронно-цифровий підпису (ЕЦП).

Таблиця 1.1 Порівняльна характеристика СЕД [6-10]

Програма/критерії	M.E.Doc	СОТА	FREDO ДокМен	Вчасно	FlyDoc
Можливість працювати в одному обліковому записі одразу з декількома підприємствами	+	+	+	+	+
Підтримка колективної роботи користувачів	+	+	+	+	+
Імпорт документів з облікових систем	+	+	+	+	+
Створення первинних документів за існуючими шаблонам безпосередньо в програмі	+	+	+	+	+
Можливість створення власних шаблонів первинних документів	+	+	+	-	+
Автоматична перевірка наявності помилок у документах	+	+	+	+	+
Контроль статусу документів (відправлено, прийнято, відхиллено, заблоковано)	+	+	+	+	+
Реєстрація податкових накладних та розрахунку коригування	+	+	-	+	+
Можливість ведення книги обліку доходів та витрат	-	+	-	-	-
Пошук документів за встановленими фільтрами (параметрами)	+	+	+	+	+
Наявність вбудованого поштового серверу	+	-	+	+	-
Можливість збереження документів на хмарі	+	+	+	+	+
Підтримка змін законодавства	+	+	+	+	+
Інтеграція з іншими програмами	+	+	+	+	+
Підтримка електронно-цифрового підпису	+	+	+	+	+
Наявність модуля «Довільне підписання»	-	-	+	-	-

Тільки в “FREDO ДокМен” наявний модуль “Довільне підписання”, але відсутні можливість ведення книги обліку доходів та витрат і реєстрація податкових накладних та розрахунку коригування.

З усього переліку, тільки “СОТА” надає можливість ведення книги обліку доходів та витрат, що є достатньою перевагою серед інших систем.

Отже, кожне підприємство самостійно обирає програмне забезпечення, для подальшого використання. Оптимальний вибір системи електронного документообігу (ЕДО) залежить не тільки від функціональних можливостей конкретної системи, а й від порядку ліцензування, вартості, надання навчання для працівників, якість послуг з технічного обслуговування.

1.3. Сучасні підходи до автоматизації документообігу на підприємстві

На сьогоднішній день виділяють наступні підходи до автоматизації документообігу:

- Document processing (обробка електронних версій документів);
- Document management (управління зберіганням і пошуком документів);
- Workflow management (управління процесами роботи);
- Workgroup management (групова робота з документами);
- Groupware (технологія групового доступу);
- Records management (управління записами);
- Digital asset management, DAM (управління цифровими активами);
- Business performance management, BPM (управління ефективністю бізнесу).

Ефективність та комфорт використання комп’ютерів для роботи з текстом призвело до появи великої кількості програм. Саме покращення текстових та табличних модулів призвело до виникнення document processing (обробка електронних версій документів).

Document management (управління зберіганням і пошуком документів) дає можливість автоматизувати процеси створення, відслідковування, зберігання, спільного використання і розподілу документів. Ця технологія додає до документів метадані та теги, за допомогою яких зв’язує їх з бізнес-процесами і забезпечує контроль на всіх етапах життя документа.

Технології workflow management (управління процесами роботи) , workgroup management (групова робота з документами), groupware (технологія групового доступу) дають можливість організувати технологію управління робочими процесами з врахуванням принципів та правил автоматизації документообігу.

Використання технології workflow management (управління процесами роботи) дозволяє уникнути необхідності зайвих контактів, оскільки документ передається по чітко визначеному маршруту.

Технологія workground management (групова робота з документами); автоматизує процес створення ланцюга погодження документу. Наприклад, особа, якій відправили документ на підпис сама обирає, хто наступний має його погодити.

Технологія groupware (технологія групового доступу) вирішує проблеми одночасної роботи з документом великої кількості осіб і інтеграції інформації з різних ресурсів в корпоративний додаток. Надання групового доступу дозволяє управляти віртуальними робочими групами, налаштовувати їх комунікацію тощо.

Технологія records management (управління записами) дозволяє ідентифікувати, зберігати і здійснювати управління документами, забезпечує роботу з архівами.

На виникнення технології digital asset management (управління цифровими активами) вплинуло наявність в організаціях аудіо, відео та фото інформації, яка необхідна для успішного управління компанією. DAM дозволяє не тільки зберігати та знаходити інформацію, а й зберігати метадані до цифрових активів, відслідковувати версіонність та права доступу до них інших користувачів.

Технологія business performance management (управління ефективністю бізнесу) підтримує управлінську парадигму процесного управління підприємством, яка розглядає неструктуровані бізнес-процеси, як особливі ресурси, які динамічно змінюються під впливом зовнішніх умов.

Існування вищевказаних технологій обумовлено правилами ведення документообігу і тенденціями розвитку інформаційно-комунікаційних технологій, підходів до управління організацією.

Кожне підприємство на свій розсуд обирає який підхід до автоматизації документообігу використовувати в своїй діяльності. Найоптимальніше рішення буде прийнято на основі врахування специфіки компанії, технічних можливостей, складності процесу документообігу, бюджету та ін. Але найкращий результат може надати тільки поєднання всіх підходів.

1.4. Нормативно-правове регулювання електронного діловодства та документообігу в Україні

З метою покращення функціонування електронного документообігу в Україні, постійно розвивається нормативно-правова база, яка описує вимоги, які висуваються до нього органами влади. Нормативно-правова база повинна відповідати за організацію здійснення відповідних процесів та дотримання вимог до оформлення документів, розроблення єдиної уніфікованої системи діловодства тощо.

Відносини, пов'язані з електронним документообігом та використанням електронних документів, регулюються Конституцією України, Цивільним кодексом України, законами України “Про інформацію”, “Про захист інформації в автоматизованих системах”, “Про державну таємницю”, “Про зв'язок”, “Про обов'язковий примірник документів”, “Про Національний архівний фонд та архівні установи”, цим Законом, а також іншими нормативно-правовими актами. Кабінет Міністрів України та інші органи виконавчої влади в межах повноважень, визначених законом, реалізують державну політику електронного документообігу.

Державне регулювання електронного документообігу спрямовано на:

- реалізацію єдиної державної політики електронного документообігу;

- забезпечення прав і законних інтересів суб'єктів електронного документообігу;
- нормативно-правове забезпечення технології оброблення, створення, передавання, одержання, зберігання, використання та знищення електронних документів.

Основоположними документами у сфері регулювання електронного документообігу в Україні є Закони України "Про електронні документи та електронний документообіг" (№ 851 від 22.05.2003 р.) та "Про електронний цифровий підпис" (№ 852 від 22.05.2003 р.).

Закон України "Про електронні документи та електронний документообіг" поширюється на відносини, що виникають у процесі створення, відправлення, передавання, одержання, зберігання, оброблення, використання та знищення електронних документів [3]. Закон регулює, що відправлення або передавання електронного документу здійснюється автором або посередником в електронному вигляді інформаційно-комунікаційними засобами або безпосереднім відправленням електронних носіїв з даним документом. Зокрема електронний документ вважається одержаним, якщо автору надійшло повідомлення в електронній формі від адресата про його одержання, якщо інше не передбачено чинним законодавством або попередньою домовленістю.

Цілісність електронного документа визначається справжністю накладеного на нього ЕЦП. У Законі надано визначення електронного документу. Це документ, інформація в якому зафіксована у вигляді електронних даних, включаючи обов'язкові реквізити документа.

Закон України "Про електронні документи та електронний документообіг" регламентує, що електронно-цифровий підпис є обов'язковим реквізитом електронного документу, який є ідентифікатором автора або підписувача електронного документу іншими суб'єктами. Накладанням ЕЦП завершується створення електронного документу. Кожен із примірників електронного документу вважається оригіналом і має однакову юридичну силу, у разі його надсилання декільком адресатам або зберігання на кількох носіях.

Електронний документ не може бути оригіналом у таких випадках:

- Свідоцтва про право на спадщину;
- Документа, який відповідно до законодавства може бути створений лише в одному оригінальному примірнику, крім випадків існування централізованого сховища оригіналів електронних документів;
- В інших випадках, передбачених законом.

Закон України “Про електронний цифровий підпис ” визначає правовий статус електронного цифрового підпису та регулює відносини, що виникають при використанні електронного цифрового підпису [11]. З прийняттям цього закону, за умови дотримання певних вимог, електронний підпис за його правовим статусом було прирівняно до власноручного підпису або печатки.

Постановою Кабінету Міністрів України від 06.05.2005 № 324 «Про заходи щодо виконання у 2005 році Програми діяльності Кабінету Міністрів України “Назустріч людям”» [12] було акцентовано увагу на необхідності ухвалення цільової програми впровадження й розвитку електронного документообігу з використанням ЕЦП.

У 2008 році Державна податкова адміністрація України (нині Державна фіскальна служба України) видала Наказ “Про подання електронної податкової звітності” (№ 233 від 10.04.2008), яким було затверджено “Інструкцію з підготовки і подання податкових документів в електронному вигляді засобами телекомунікаційного зв’язку”, яка визначає загальні принципи організації інформаційного обміну під час подання платниками податків податкової звітності до органів державної податкової служби України в електронній формі із використанням електронного цифрового підпису [13]. Згідно цього Наказу організації мають можливість подавати податкові документи до органів Державної фіскальної служби з використанням інформаційних систем. Для цього платники податків повинні мати: доступ до Інтернету, посиленні сертифікати ЕЦП, програмні продукти, які формують податкові документи в електронному вигляді та засіб криптографічного захисту інформації.

Електронний документообіг регулюється і іншими нормативно-правовими актами. Серед них:

- “Про затвердження Порядку засвідчення наявності електронного документа (електронних даних) на певний момент часу” [14];
- “Про затвердження Порядку акредитації центру сертифікації ключів” [15];
- “Про затвердження Положення про центральний засвідчувальний орган” [16];
- “Про затвердження Порядку застосування електронного цифрового підпису органами державної влади, органами місцевого самоврядування, підприємствами, установами та організаціями державної форми власності” [17];
- “Про затвердження Типового порядку здійснення електронного документообігу в органах виконавчої влади” [18];
- “Про затвердження Порядку обов’язкової передачі документованої інформації” [19].

Оновлення і розвиток нормативно-правової бази в Україні та створення спеціальних юридичних норм створюють позитивне середовище для подальшого впровадження та удосконалення електронного підпису та електронного документообігу.

ЦКУ та закони вказанні вище, стали правовою базою для широкого застосування новітніх технологій у цивільних правовідносинах, поширюючи їх на необмежену сферу правовідносин органів державної влади, суб’єктів господарювання та громадян [20].

РОЗДІЛ 2.
ОГЛЯД ТЕХНОЛОГІЙ СТВОРЕННЯ СИСТЕМ ДЛЯ ЕЛЕКТРОННОГО ДОКУМЕНТООБІГУ

2.1. Метод технології Python Django

Django - це високорівневий Python веб-фреймворк, який підтримує швидку розробку і чисту та прагматичну архітектуру. Створений досвідченими розробниками, він вирішує більшу частину проблем веб-розробки [21].

Django з'явився на ринку в 2005 році, і поступово став одним з кращих фреймворків, який допомагає тисячам розробників виконувати роботу протягом декількох хвилин. Django суттєво спростив чимало складнощів в розробці веб-додатків, надавши розробці більш зрозумілий підхід.

Використання технології Django має такі переваги[22]:

- Тісний зв'язок між компонентами. Всі інтегровані компоненти Django були створені групою розробників. Перш за все Django був розроблений в якості внутрішньої основи для управління сайтами з новинами. Через це компоненти були призначені для інтеграції, багатократного використання і швидкості з самого початку;
- ORM(Об'єктно-реляційна проекція). Компонент бази даних ORM забезпечує зв'язок між моделлю бази даних і її движком. Він підтримує великий набір баз даних і переключення з одного движку на інший, це залежить лише від конфігурації в файлі. Це дає розробнику свободу зміни движку бази даних;
- Зрозуміла URL-архітектура. Система URL в Django дуже гнучка і потужна, вона дозволяє визначити шаблони для URL-адрес в веб-додатку і задати функції для кожного шаблону;
- Вбудований інтерфейс адміністратора. Django після встановлення є готовим до використання с інтерфейсом адміністратора. Цей інтерфейс

допомагає оперувати даними, які є в вашому додатку. Він є дуже гнучким і легко налаштовується;

- Передове середовище розробки. Окрім того, Django забезпечує чудове середовище розробки, до складу якого входить легкий веб-сервер для розробки і тестування. Коли режим налагодження включений, Django надає дуже детальні інструкції та повідомлення про помилки. Все це робить процес знайдення і виправлення помилок дуже простим;
- Багатомовна підтримка. Django підтримує багатомовні сайти завдяки своїй інтернаціональній системі, яка робить переклад інтерфейсу дуже легкою задачею.

З вищесказаного можна зробити висновок, що Django розроблена з особливою підтримкою “слабоз’язаності”(незалежності) і чітким розмежуванням компонентів додатку. Якщо підтримувати цю філософію, то буде легко вносити зміни в одну частину програми, не чіпаючи інші.

На схемі (рис 2.1) зображено послідовність виконання запиту на різних рівнях серверу Django:

- На рівні додатку відбувається виведення HTML сторінки і контроль відображення її користувачу;
- Логіка доступу до даних, логіка виконання і логіка представлення - разом ці частини є концепцією, яку називають патерном модель-представлення-контролер (англ. Model-View-Controller, скорочено MVC). В даній схемі “Модель” - включає в себе визначення класів для зберігання інформації, “Представлення” - контроль доступу і фільтрація інформації, в тому порядку, в якому вона необхідна на рівні відображення, “Контролер” - отримує і опрацьовує матеріали для оновлення рівня моделі, окрім того, контролер оновлює елементи для рівня представлення при необхідності [23];
- На рівні бази даних моделі зберігаються в таблицях обраної бази даних.

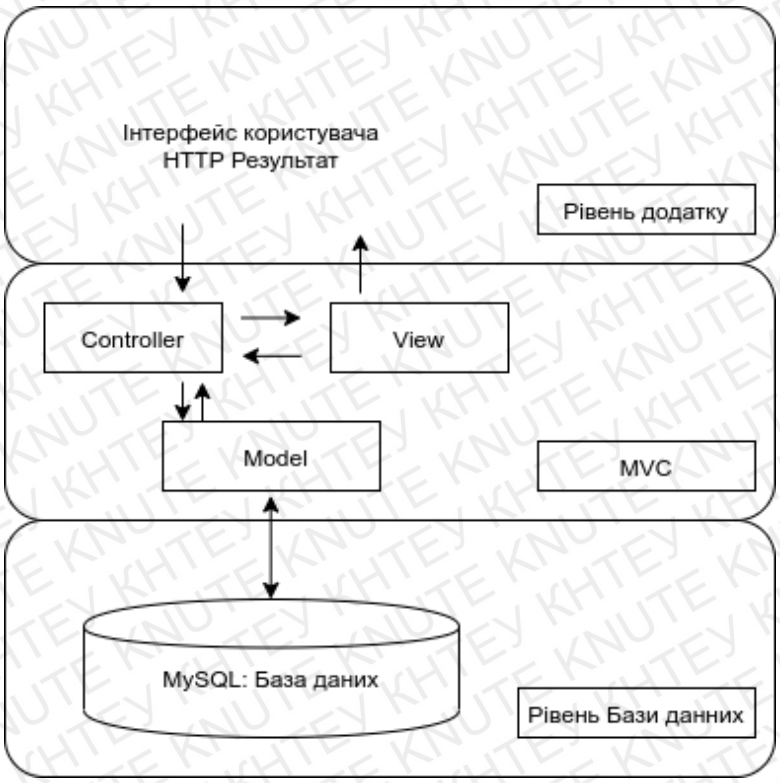

Рис. 2.1. Схема роботи серверу Django

Незважаючи на велику кількість плюсів використання Django, є й певні мінуси, такі як:

- необхідність використання певного шаблону маршрутизації;
- монолітність фреймворка;
- все базується на Django ORM;
- для початку роботи необхідно знати всю систему.

Існує багато переваг та недоліків Django, але коли ми маємо проект з визначеним терміном, використання Django для цього проекту є оптимальним рішенням.

2.2. Метод технології Python Django REST Framework

Django REST Framework (DRF) - це потужний і гнучкий Python веб-фреймворк для створення веб-API, який працює зі стандартними Django-моделями.

REST API використовується в веб-додатках або програмах для зв’язку серверної частини і інтерфейсу користувача.

Прикладний програмний інтерфейс (англ. Application Programming Interface, скорочено API) - це набір означень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення. API надає розробнику засоби для швидкої розробки програмного забезпечення оскільки можна скористуватися готовими об’єктами (функціями) іншого програмного забезпечення через означені в останньому правила взаємодії [24].

Чимало фреймворків дозволяють легко розробляти API для додатків, але в межах даного проекту ми розглянемо Django REST. Його використання в багатьох випадках надає широкий спектр переваг [25]:

- REST API доступний для перегляду в веб-браузері;
- Політика аутентифікації включає пакети для OAuth1 і OAuth2;
- Серіалізація підтримує як джерела даних ORM, так і джерела, які не пов'язанні з ORM;
- Можливість повного налаштування. Якщо немає необхідності в потужних модулях і функціях, то можна використовувати звичайні функції;
- Модуль розмежування доступу;
- Модуль пагінації;
- Детальна документація і величезна спільнота розробників;
- Використовується і користується довірою міжнародно визнаних компаній, включаючи Mozilla, Red Hat, Heroku и Eventbrite;
- Можливість використання всіх переваг Django.

Архітектура Django REST Framework складається з таких рівнів (рис 2.2):

- Сериалізатор - форматує інформацію, яка зберігається в базі даних і визначену моделями Django, в формат, який ефективно і легко передається через API;
- Представлення(Views) - визначає функції(читання, створення, оновлення, видалення), які будуть доступні через API;
- Маршрутизатор(URL Patterns) - визначає URL-адреси, які будуть надавати доступ до кожного представлення.

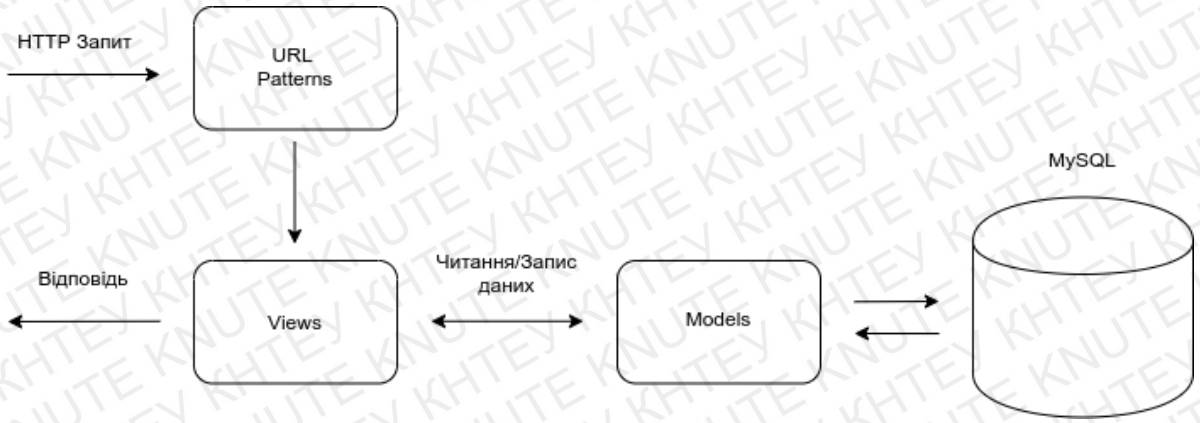

Рис. 2.2. Архітектура Django REST Framework

Використання Django REST API є доцільним у випадках, коли потрібно:

- Створити додаток з фреймворком для клієнтської частини, наприклад, React або Angular;
- Розробити серверну частину для мобільного додатку;
- Написати систему, до якої можна отримати доступ з різних платформ;
- Відображати динамічну зміну даних.

Для побудови складних додатків, Django REST API є більш потужним і гнучким інструментом, оскільки дає можливість спільного використання даних на різних сервісах і додатках, що є важливою у мовою подальшої масштабованості і інтеграції з різними системами.

2.3. Метод технології JavaScript Angular

Angular - це веб-фреймворк на базі TypeScript з відкритим вихідним кодом, який був розроблений в 2009 році. Angular використовується в основному для створення односторінкових додатків. Це один з найпопулярніших JavaScript фреймворків.

Angular має багато функцій, наприклад двостороння прив’язка даних, шаблонізація, маршрутизація, компоненти тощо.

Перевагами використання Angular є [26]:

- Кросплатформність. Angular може бути використаний для побудови високо продуктивних веб-додатків, створення мобільних додатків з

використанням технологій Cordova, Ionic або NativeScript, розробки програмного забезпечення для ПК;

- Генерація коду. Angular перетворює шаблони в оптимізований код для сучасних JavaScript віртуальних машин;
- Універсальність. Можливість використання серверів на різних мовах для миттєвого рендеренгу на чистому HTML і CSS, також наявна опція SEO-оптимізації посилань на сайті;
- Розмежування коду. Angular-додатки швидко загружаються з новою маршрутизацією компонента, який підтримує автоматичне розмежування коду, тому користувачу загружається тільки той код, який необхідний для перегляду;
- Анімація. Створення високопродуктивних і складних анімацій з невеликою кількістю коду через інтуїтивний Angular API.

Використання Angular є необхідним в таких випадках [27]:

- Створення великомасштабного односторінкового додатку;
- В додатку буде багато функцій і він буде включати динамічний контент;
- Проект є довгостроковим;
- Необхідність розробки гібридного або прогресивного додатку .

На жаль, як більшість схожих фреймворків, Angular досить важкий для початкового розуміння через велику кількість концепцій необхідних для роботи з ним.

2.4. Метод технології JavaScript React

React - це JavaScript бібліотека для побудови користувацького інтерфейсу. Його головною задачею є забезпечення виводу на екрані того, що можна бачити на веб-сторінках. React значно спрощує створення інтерфейсів, завдяки поділу кожної сторінки на невеликі фрагменти. В React ці фрагменти називаються компонентами.

React компонент - це частина коду, який представляє частину веб-сторінки. Кожен компонент - це JavaScript-функція, яка повертає кусок коду, який представляє фрагмент сторінки.

Для формування сторінки ми викликаємо компоненти в певному порядку, збираємо разом результати викликів і показуємо їх користувачу.

Перевагами React є [28]:

- Декларативність. React робить простим процес створення інтерактивних користувацьких інтерфейсів;
- Оснований на компонентах. Так як логіка компоненту написана на JavaScript, а не на шаблонах, досить легко передавати дані через додаток і тримати стан поза DOM;
- Синтаксис JSX, який схожий на HTML;
- Одностороння прив’язка даних;
- Відсутність обов’язкової прив’язки до класів, що спрощує код.

Вивчення React на початковому етапі дається набагато легше, ніж Angular, оскільки для створення компонентів React використовує функції, а не класи.

При виборі інструменту для розробки користувацького інтерфейсу, перевагу варто надати React, якщо [27]:

- Є необхідність в модульному інтерфейсі;
- Потрібне рішення з можливістю повного налаштування, яке залишає місце для подальшого зростання;
- Додаток має підтримувати SEO;
- В подальших планах створити кросплатформений додаток на основі React Native.

Динамічна бібліотека React, є надійною технологією, яка широко використовується в індустрії розробки, навіть для проектів зі складною архітектурою.

2.5. Метод технології MongoDB

MongoDB- це кросплатформна документорієнтована база даних (БД) з відкритим вихідним кодом, що зберігає дані в JSON-подібних документах [29]. Це одна з найпопулярніших NoSQL-систем.

NoSQL-система - бази даних, які являються базами даних наступного покоління, які в основному характеризуються [30]:

- Нерелятивністю;
- Розподіленістю;
- Відкритим вихідним кодом;
- Горизонтальною масштабованістю.

MongoDB підтримує зберігання документів в JSON-подібному форматі, має дуже гнучку мову для формування запитів, може створювати індекси для різних атрибутів, які зберігаються, ефективно забезпечує зберігання великих бінарних об’єктів, підтримує запис операцій по зміненню і додаванню даних в БД, може працювати у відповідності з парадигмою Map/Reduce, підтримує реплікацію і побудову відмовостійких конфігурацій.

Особливості MongoDB:

- Документоорієнтованість. MongoDB зберігає бізнес-об’єкт в мінімальній кількості документів;
- Спеціальні запроси. Підтримання пошуку по діапазону, регулярному виразу;
- Індексація. Будь-яке поле в документі може бути проіндексоване;
- Реплікації. MongoDB забезпечує високу доступність реплік. Репліка - набор із двох або більше копій даних;
- Балансування навантаження. MongoDB може працювати на декількох серверах;
- Зберігання файлів. MongoDB може бути використаний в якості файлової системи, використовуючи балансування і реплікацію даних;
- Агрегація. Map/Reduce може бути використаний для пакетної обробки даних операцій агрегації;

- Серверне виконання JavaScript. JavaScript може бути використаний в запитах, функціях агрегації(такі як Map/Reduce) і направлені напряму в базу даних, яка буде виконана;
- Блокування колекцій. MongoDB підтримує колекції фіксованого розміру.

Використання MongoDB дає переваги в тому, що ми маємо гнучкий JSON-формат документів. Для деяких задач це набагато зручніше, ніж додавати колонки в SQL-базах даних.

2.6. Метод технології PostgreSQL

PostgreSQL - це об'єктно-реляційна система управління базами даних с акцентом на розширення і відповідність стандартів. PostgeSQL сумісний з транзакційністю, має обновлювальні і матеріалізовані представлення, тригери і іноземні ключі. Вона підтримує функції і збереження процедур.

PostgreSQL використовує таблиці, обмеження, тригери, ролі, збереження процедур і зображення в якості ключових компонентів. Таблиця складається з рядків і кожен рядок складається з одного і того ж набору стовбців. PostgreSQL використовує первинні ключі, щоб однозначно ідентифікувати кожен рядок в таблиці, і зовнішні ключі, щоб забезпечити посилальну цілісність між двома суміжними таблицями.

PostgreSQL підтримує багато функцій NoSQL. Також вона підтримує частину стандарту SQL і пропонує більшість сучасних функцій:

- Складні запроси;
- Зовнішні ключі;
- Тригери;
- Представлення, які можуть змінюватись;
- Транзакційна цільність;
- Мультиверсійність .

PostgreSQL можна інтегрувати з будь-якою мовою програмування, такими якими Java, C, C++, Python і т.д. Це дозволяє визначити власні функції,

які можна налаштувати. Мова структурованих запитів Postgre має багато можливостей, які ми можемо знайти і в інших БД.

Враховуючи, що PostgreSQL достатньо стара БД, ми можемо швидко знайти рішення проблем пов’язаних з даною технологією.

PortgreSQL потребує мінімальних зусиль завдяки своїй стабільності. Таким чином, для розробки програмного забезпечення на основі PostgreSQL, загальний час ознайомлення є досить низьким в порівнянні з іншими системами управління базами даних.

РОЗДІЛ 3.
СТВОРЕННЯ СИСТЕМИ АВТОМАТИЗАЦІЇ ДОКУМЕНТООБІГУ

3.1. Вимоги до структури та функціонування системи

Автоматизована система документообігу призначена для забезпечення збору, передачі, валідації та зберігання інформації, що буде одержана в процесі діяльності умовного підприємства.

Основною ціллю є створення автоматизованої системи, яка буде виконувати функції отримання інформації, її збереження та подальшого використання.

Система повинна виконувати такі завдання:

- Надавати безпечний трансфер даних на всіх рівнях з можливістю ідентифікації та верифікації довіреними особами або автоматично, якщо це передбачено специфікацією певного звіту;
- Надання користувачам можливості надання інформації через інтуїтивний інтерфейс користувача;
- Можливість формування аналітики подачі звітності в системі з використання стороннього програмного забезпечення;
- API сервіси для автоматичного надання інформації іншим корпоративним системам;
- Зменшення впливу людського фактору на обробку інформації за рахунок автоматизації процесів системи та алгоритмів перевірок коректності інформації.

Принципи побудови системи:

- Раціональне співвідношення між витратами і поставленими цілями;
- Модульна архітектура - система повинна будуватися за допомогою модулів, для забезпечення стабільності роботи кожного модулю та його автономності. Такий підхід забезпечує гнучкість функціоналу системи та зменшить вплив одних елементів системи на інші;

- Можливість надання користувачам різних рівнів доступу до інформації, її розміщення та редагування;
- Забезпечення постійного доступу до системи користувачам, які мають відповідні права;
- Автоматичне розміщення та актуалізація даних через інтерфейс користувача системи;
- Гарантія точності, достовірності та повноти даних в системі;
- Отримання системою достовірних даних про користувача за рахунок ідентифікації користувача за логіном та паролем;
- Забезпечення подальшої можливості додавання нових модулів без зміни існуючих за рахунок модульної архітектури.

На основі принципів побудови системи можемо сформувати загальні та функціональні вимоги до системи.

Загальні вимоги до системи такі:

- Вихідна інформація повинна бути представлена єдиною версією та відповідати вхідним даним, що отримані під час роботи системи;
- Вся інформація, що була внесена в систему повинна зберігатись в базі даних, якщо інформація змінюється, то попередню версію буде видалено з занесенням відповідного запису до журналу користувача;
- Всі модулі повинні бути побудовані за єдиною методологією із застосуванням спільних підходів щодо їх взаємодії та надійності;
- Розмежування інформації на основі прав доступу користувачів.
- Архітектура повинна забезпечувати захист інформації;
- Однотипна інформація повинна бути представлена у єдиному візуальному стилі;
- Модуль звітів повинен давати можливість створення нових звітів без редагування існуючих;
- Технічна розробка системи проводиться на основі технологій:
    - Python/Django REST API/ ReactJS/ PostgreSQL

Функціональні можливості системи повинні надавати змогу:

- Реєстрації всіх дій користувачів в системі;
- Створення нових звітів відповідальними особами з прописуванням логіки обрахунку та валідації даних системою;
- Налаштування сповіщень та контролю подання звітності у визначений час, якщо час подання звіту вийшов відповідно до регламенту, система повинна сповістити відповідальних за цей звіт людей по електронній пошті щодо необхідності подання інформації;
- Захист інформації в системі від знищення, пошкодження, несанкціонованого доступу та блокування доступу до неї за допомогою здійснення організаційних і технічних заходів, впровадження засобів програмного захисту інформації;
- Організація інформації в звітах з максимальною кількістю перевірок на коректність вхідних даних;
- Доступ до системи і розмежування прав доступу до інформації має бути надано тільки користувачам, у яких є відповідні права доступу налаштовані в системі.

Розробка системи буде відбуватися з використанням MVP патерну - це означає, що перша версія програмного продукту має містити мінімальний функціонал [31]. В межах даного проекту, відповідно до MVP патерну, мінімальним функціоналом є авторизація користувача та подання ним простого звіту.

Отже, на основі чітких формалізованих вимог, в наступному підрозділі ми будемо розробляти архітектуру бази даних та прототипи основних сторінок.

3.2. Основні модулі системи та їх прототипи

Програми, які мають реальне практичне значення і вирішують певні проблеми, мають чимало різних функцій. Якщо всі ці функції ми будемо записувати в один файл, то з часом в ньому буде важко орієнтуватися і підтримувати систему в робочому стані.

Технологія модульного програмування полягає у розбитті великої задачі на частини, з наступною реалізацією кожної підзадачі в окремому файлі, який називають модуль. На практиці, один програмний продукт може включати модулі, написані різними мовами програмування. На вибір мови впливає складність реалізації алгоритму [32].

За функціональними вимогами в нашій системі ми можемо виділити наступні модулі:

- Модуль Користувача (User). Він відповідає за збереження особистої інформації користувачів(ПІБ, email, телефон, посада, дата реєстрації та останнього візиту в системі тощо), приналежність до організації та департаменту. Через цей модуль відбувається авторизація користувача та надання доступу в систему;
- Модуль Звіту (Report). Дозволяє створювати і заповнювати звіти. В цьому модулі знаходиться функціонал створення шаблону звіту, типи даних, логіка окремих комірок та журнал операцій.

Таким чином, розподілення системи на вищевказані модулі дозволить спростити процес розробки і дає багато можливостей для масштабування без зміни логіки інших модулів.

Для того, щоб детальніше розглянути взаємодію цих модулів в системі, побудуємо діаграму потоків (Flow Diagram) мовою UML (Unified Modeling Language), що розроблена спеціально для розробки системного забезпечення.

Діаграма потоків (Flow Diagram) - схематичне представлення алгоритму розв'язування або аналізу задачі за допомогою геометричних елементів (блоків), які позначають операції, потік, дані тощо.

На діаграмі потоків (рис. 3.1) зображений базовий шлях користувача та адміністратора в системі. Як можемо бачити, користувач не може ввійти в систему, якщо адміністратор не додав його. Після успішної авторизації користувач може обрати макет та створити звіт.

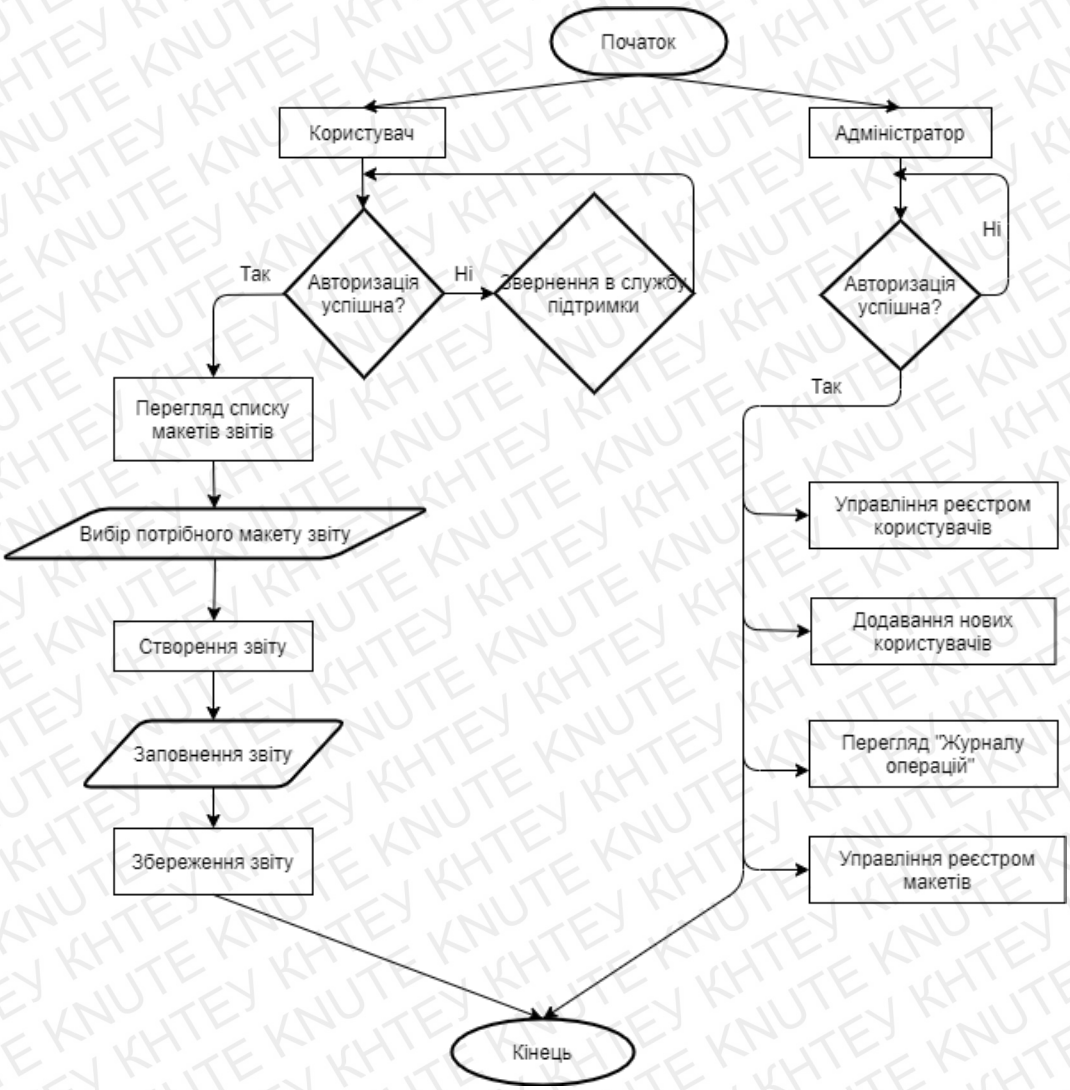

Рис. 3.1. Діаграма потоків створення звіту та адміністрування системи

На основі діаграми, ми створимо прототипи системи у програмі Justinmind Prototyper. Прототип у розробці веб-сервісів - це просте візуальне представлення інтерфейсу ресурсу або сервісу [33].

Створення прототипів є важливим, тому що:

- Робить ресурс зручнішим;
- Допомагає переконатися, що ви створюєте правильний функціонал;
- Допомагає всій команді краще зрозуміти майбутній продукт;
- Дозволяє перевіряти ідеї набагато швидше і з меншим ризиком.

Ми створимо попередні креслення, які показують структуру, розташування та ієрархію елементів в нашій системі. На цьому етапі, ми не будемо обирати шрифти, кольори та інші візуальні елементи.

Нам необхідно створити прототипи таких функціональних елементів:

- Сторінка авторизації;
- Відображення звітів користувача;
- Створення звіту з макетів;
- Сторінка заповнення звіту.

Ми не будемо створювати прототипи сторінок для адміністратора, оскільки Django має вже вбудовану панель адміністратора, яка дає можливість виконання всіх функцій перелічених в діаграмі потоків для адміністратора.

На сторінці авторизації вхід користувача відбувається за рахунок введення комбінації логін/пароль (рис. 3.2).

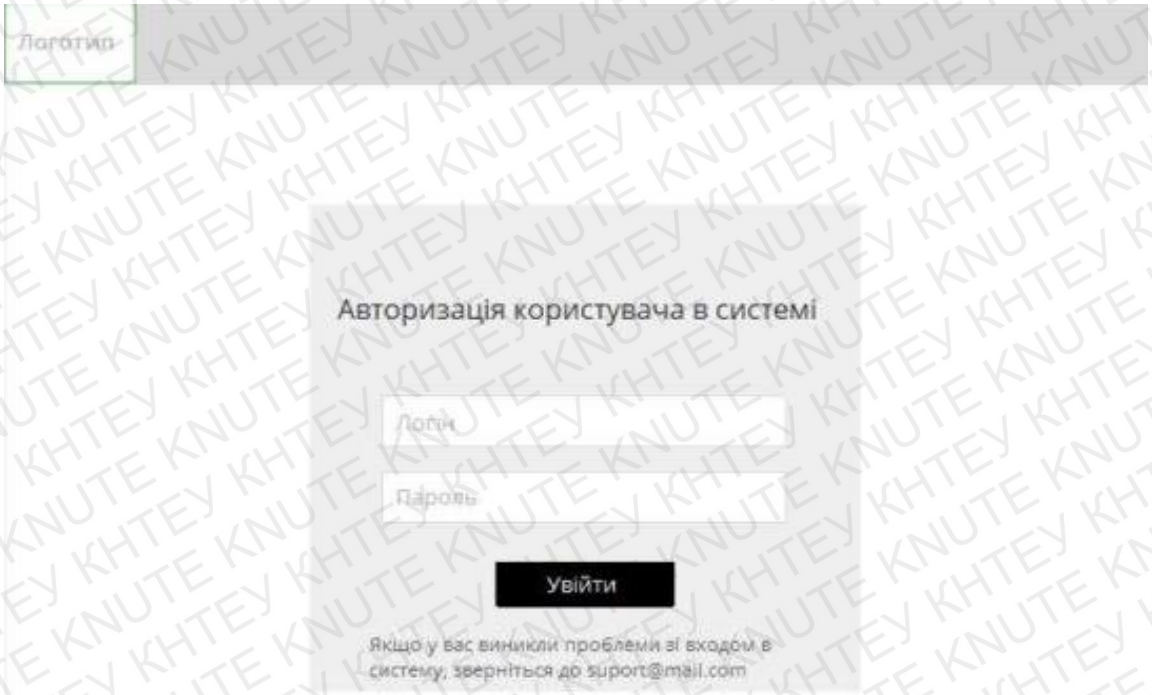

Рис 3.2. Прототип сторінки авторизації

Після натискання кнопки “Увійти” система здійснює перевірку на відповідність даних користувача та відображає сторінку з реєстром макету, якщо такий користувач є в системі.

На головній сторінці (рис.3.3) користувач бачить його звіти та звіти інших користувачів, з якими знаходиться в одній організації. В таблиці звітів вказано назву, дату створення, статус та останню дію над звітом.

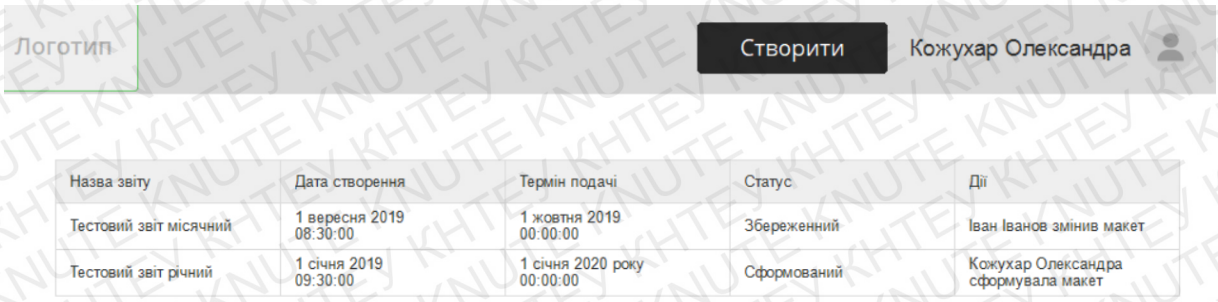

Рис 3.3. Прототип головної сторінки

При натисканні на кнопку “Створити”, відкривається модальне вікно з доступними макетами для створення (рис. 3.4). Користувач може обрати тільки один і після цього натиснути “Створити”.

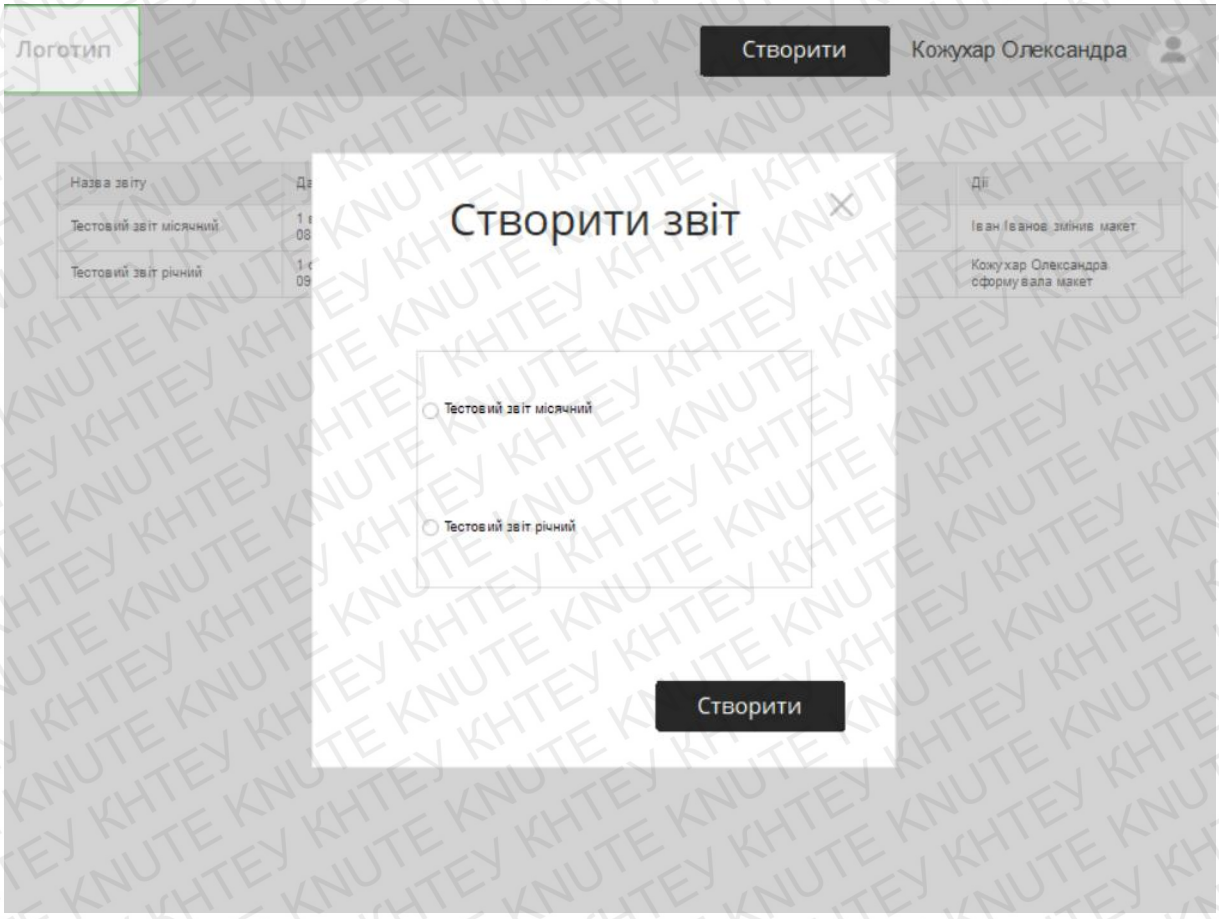

Рис. 3.4. Прототип модального вікна створення звіту

Після того, як користувач обрав макет, він переходить на сторінку створення звіту (рис 3.5). На цій сторінці ми бачимо назву організації від якої створюється звіт, назву звіту, таблицю звіту, історію та кнопку для збереження.

У блоці “Історія” ми маємо відповідний запис про створення звіту та додаткову інформацію про те, ким та коли був створений цей звіт.

Рис 3.5. Прототип сторінки створення звіту

Після заповнення звіту, користувач натискає на кнопку “Зберегти”. Система зберігає дані та додає відповідний запис про редагування в блок “Історія” (рис. 3.6).

Рис. 3.6. Прототип сторінки збереженого звіту

В даному підрозділі ми наглядно розглянули шлях користувача в системі та побудували прототипи відповідно до нього.

Саме на основі функціоналу, який було зображено на прототипах, ми будемо будувати серверну частину та користувацький інтерфейс MVP версії системи.

3.3. Розробка серверної частини системи

Розробку будь-якого програмного забезпечення слід починати з його архітектури. Під терміном “архітектура” слід розуміти:

- Об'єднання структурних елементів у логічні групи для отримання цілісної системи;
- Розбиття програмного забезпечення на модулі;
- Опис зв’язків між структурними елементами;
- Опис структур даних, що будуть використовуватися для взаємодії.

Нашу систему електронного документообігу ми будемо будувати з використанням сервісно-орієнтованої архітектури. Сервісно-орієнтована архітектура дозволяє серверній частині та користувацькому інтерфейсу працювати на різних платформах та інтегруватися з іншими сервісами та системами [34].

В нашому випадку, ми будемо з інтерфейсу користувача, який написаний на React звертатися до API наших сервісів на серверній частині Django REST для отримання, зберігання або зміни інформації в базі даних PostgreSQL.

В межах даного підрозділу ми опишемо сутності для сховища даних та розробимо його архітектуру.

У базі даних ми будемо зберігати всю інформацію про користувачів, організації, шаблони звітів, звіти, історію дій над звітом тощо.

Розглянемо основні SQL типи даних, які ми будемо використовувати під час розробки системи на основі джерел [35, 36]:

- INTEGER – середнє ціле число. Діапазон значень від -2147483648 до 2147483647;
- VARCHAR(розмір) – рядок різного розміру(може включати літери, числа та спеціальні символи). Параметр розміру визначає максимальну довжину стовпця в символах – може бути від 0 до 65535;
- DATE – дата в форматі РРРР-ММ-ДД. Підтримується діапазон від ‘1000-01-01’ до ‘9999-12-31’;
- TIMESTAMP – позначення часу. Значення зберігається як кількість секунд після епохи UNIX(‘1970-01-01 00:00:00’ UTC). Формат: РРРР-ММ-ДД год. Год .: мм: сс. Підтримується діапазон від ‘1970-01-01 00:00:01’ UTC до 2038-01-09 03:14:07 ‘UTC;

Під час побудови та розробки бази даних (рис. 3.7) ми будемо використовувати тип зв’язку між таблицями, який називається Зовнішній Ключ (Foreign Key) - це поле, пов’язане з полем первинного ключа іншої таблиці у співвідношенні між двома таблицями [37].

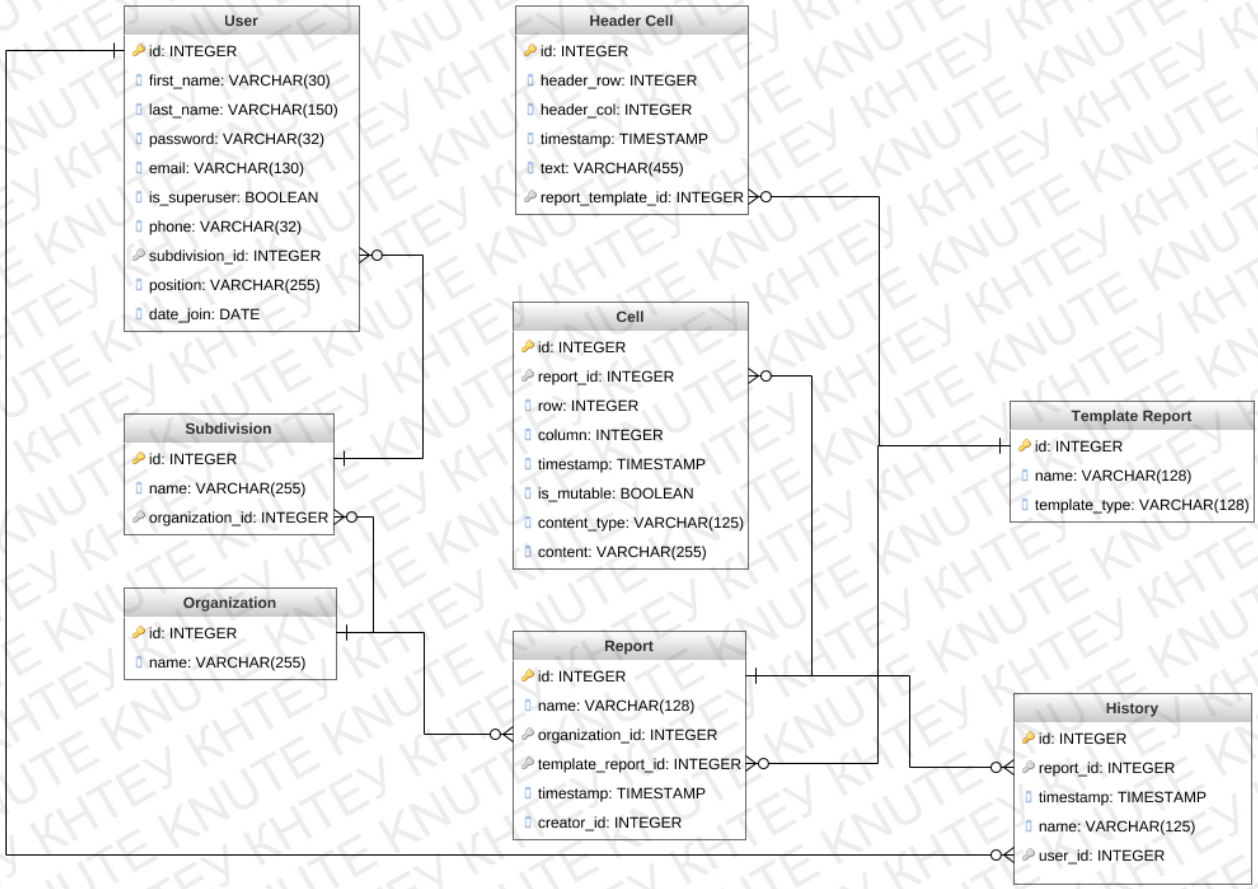

Рис. 3.7. Схема бази даних системи

Розробку системи розпочнемо з правильної файлової структури додатку. Кожен модуль повинен мати таку структуру (рис. 3.8):

- Admin.py використовується для відображення моделей в панелі адміністратора Django, а також для налаштувань;
- В models.py ми будемо зберігати об’єктне представлення відношень бази даних;
- Serializers.py дозволяє змінити структуру даних або стан об’єкту у формат, який зручно зберігати, передавати та використовувати потім;
- Views.py зберігають Python-функції, які приймають веб-запит і повертають веб-відповідь;
- Apps.py слугує для включення різних конфігурацій додатку;
- __init__.py необхідний для обробки Python директорій, які включають файл у вигляді пакетів.
- Директорія migrations формується автоматично для поширення змін, які вносяться в моделі, до схеми бази даних.

Рис. 3.8. Приклад файлової структури модулю

Для побудови API ми будемо використовувати такі HTTP методи:

- POST – створення нового запису в БД;
- GET – отримання інформації за запитом;
- PATCH – часткове оновлення або модифікація запису в БД;

Модуль під назвою “User” призначений для зберігання всієї інформації, про користувача, а також його організацію та департамент.

Для створення моделі User ми розширимо стандартну Django модель користувача AbstractUser в якій вже є стандартні методи валідації на унікальність логіну та надійність паролю.

В модуль User також входять моделі Organizations та Subdivisions, які відповідно пов’язані зовнішнім ключем з моделлю User.

В представленнях для User у нас мають бути такі точки входу, де id – це унікальний ідентифікатор користувача в БД:

- GET /api/v1/user/{id} – отримання всієї інформації про користувача (ім’я, прізвище, email, підрозділ, телефон, посада тощо);
- GET /api/v1/user/{id}/reports – отримання всіх звітів, які доступні користувачу відповідно до організації;
- GET /api/v1/user/{id}/templates – отримання всіх макетів звіті, які доступні користувачу відповідно до організації.

Для авторизації користувачів ми будемо використовувати стандартний пакет Django ModelBackend. Це основний сервер аутентифікації, який перевіряє базу даних користувачів Django і запитує вбудовані дозволи. В подальшому, для підвищення безпеки, буде додано аутентифікацію з видачою JWT-токену користувачу.

Модуль “Reports” - це модуль системи, в якому створюються макети звітів та самі звіти, записується історія роботи зі звітом.

До модулю Reports входять такі моделі:

- TemplateReport - це безпосередньо об’єкт макету звіту, який посилається на організацію, для якої передбачений макет звіту;
- HeaderCell - це комірки шапки таблиці звіту, який посилається на TemplateReport, шапка звіту завжди визначається типом макету;
- Report - це модель звіту, який має бути створена з макету звіту, посилання на який присутній в моделі. Також Report посилається на організацію, якій належить звіт. Класу Report також належить метод get_cells(self), який повертає всі комірки, які належать звіту та сортує їх по колонкам та рядкам;

- Cell - це модель звичайної комірки звіту, яка визначає позицію в межах всієї таблиці звіту та дані, які в ній зберігаються. Дана модель посилається на Report;
- History - це модель, для збереження дій над макетом, яка посилається на Report.

В представленнях для Report у нас мають бути такі точки входу, де id – це унікальний ідентифікатор звіту в системі:

- POST /api/v1/report – створення нового звіту, в параметри запиту передаємо ідентифікатор макету звіту.
- GET /api/v1/report/{id} – отримання інформації про звіт (комірки шапки звіту, комірки звіту, назва звіту, користувач, який створив звіт, його організація)
- GET /api/v1/report/{id}/history – отримання інформації про всі дії над звітом (користувач, який здійснив дію, дата події та дія);
- PATCH /api/v1/report/{id}/cell_update – оновлення даних в комірках звіту, в якості параметрів приймає комірки, які змінились.

Модуль Report спроектований з використанням патерну Будівельник. Патерн Будівельник пропонує винести конструювання об’єкта за межі його власного класу, доручивши цю справу окремим об’єктам, які називаються будівельниками [38].

В нашому випадку клас Будівельника виконує всі необхідні функції для відображення звіту користувачу, а саме:

- Знаходить організацію користувача та перевіряє чи має він доступ до звіту;
- Створює об’єкт Report;
- Знаходить всі комірки, які на лежать необхідному звіту;
- Повертає готовий звіт в форматі об’єкту Report.

В наступній версії планується додати функціонал регламенту подачі звіту, який передбачає те, що звіт буде самостійно створюватись по регламенту,

якого потребує звіт. Також необхідним є можливість відправлення звіту на погодження відповідно до ієрархії організацій, яка визначена макетом.

3.4. Розробка користувацького інтерфейсу

Як вже було сказано, в React основним структурним елементом є компоненти, які подібні до користувацьких, багаторазових HTML-елементів, для швидкого та ефективного створення інтерфейсів користувача. Також React регулює спосіб зберігання та обробки даних за рахунок станів та властивостей, які передаються від батьківського компоненту в дочірні. Таким чином створюється певне ієрархічне дерево компонентів.

Для створення компонентів ми будемо використовувати синтаксис JSX - це синтаксичне розширення для Javascript, яке дозволяє писати HTML структури в тому ж файлі, який містить Javascript код.

В React виділяють 2 види компонентів [39]:

- Функціональні компоненти – ці компоненти не мають власного стану і містять лише метод візуалізації, тому їх також називають компонентами без стану. Вони можуть отримувати дані з інших компонентів як реквізити(властивості);
- Класові компоненти – ці компоненти можуть зберігати та керувати своїм станом та мати окремий метод візуалізації для JSX. Їх ще називають становими компонентами, оскільки вони можуть мати стан.

Всі компоненти в системі ми розподілимо на 3 групи, які будуть відповідними директоріями:

- Components. Функціональні компоненти, найменші елементи системи, які часто будуть використовуватись для побудови класових компонентів;
- Containers. Класові компоненти, в які ми будемо імпортувати функціональні компоненти і пов’язувати їх спільною логікою;
- Layouts. Представлення, які являють собою сторінки веб-додатку, які складаються з контейнерів, в які передаються властивості та стани.

Саме з побудови функціональні компонентів ми почнемо розробку клієнтського інтерфейсу. До таких компонентів в нашій системі відносяться:

- Кнопка;
- Поле для введення тексту та числових даних;
- Спливаюче вікно;
- Модальне вікно;
- Логотип з назвою системи;
- Іконка користувача.

На прикладі компоненту кнопки розглянемо побудову функціональних компонентів в системі (рис 3.9). Для стилізації компонентів ми будемо використовувати React-бібліотеку “styled-components”, яка дозволяє динамічно змінювати стилі без використання класів, підтримує автоматичне додавання вендорних префіксів для браузерів тощо.

src > components > Buttons > MainButton.jsx > ...

```jsx
import React from 'react';
import PropTypes from 'prop-types';
import styled from 'styled-components';
// стилі для кнопки
export const MainButtonStyled = styled()`
  padding: 12px 24px;
  box-sizing: border-box;
  border-radius: 8px;
  font-size: 14px;
  font-weight: 500;
  color: #ffffff;
  background: #13a47f;
  :hover {
    background-color: #176652;
  }`;

const MainButton = props => {
// декларування змінних для властивостей
  const { name, onClick } = props;
// компонент кнопки
  return (
    <MainButtonStyled onClick={onClick} {...props}> {name} </MainButtonStyled>
  );
};

export default MainButton;
// типи даних для властивостей
MainButton.propTypes = {
  name: PropTypes.string,
  onClick: PropTypes.func
};
// значення по замовченню для властивостей
MainButton.defaultProps = {
  name: '',
  onClick: null
};

```

Рис. 3.9. Компонент кнопки

Для всіх компонентів ми будемо визначати тип даних для змінних та значення за замовчуванням для зменшення кількості помилок в системі. За аналогією компоненту кнопки будуємо і інші функціональні компоненти на клієнтській частині.

Оскільки, як вже було сказано, увесь інтерфейс є ієрархічним, тому усе, що ми бачимо на клієнтській частині - це один кореневий компонент “App”, який є обов’язковим. В цей компонент виносять спільні елементи всіх сторінок. В нашій системі таким компонентом є шапка сайту, до якої входять логотип, кнопка та іконка користувача із спливаючим меню. Також в кореневий компонент “App” ми винесемо компоненти представлення сторінок.

Усього наша система містить 3 компоненти представлення:

- Сторінка авторизації (LoginPage). Якщо користувач не авторизований, то це перша сторінка, на яку він потрапляє. Користувач повинен ввести логін та пароль, якщо спроби є невдалими, то він повинен звернутись в службу підтримки;
- Головна сторінка (MainPage). Сторінка, на якій користувач може переглянути вже створені макети, та створити новий з макетів, які йому доступні;
- Сторінка звіту (ReportPage). На цій сторінці користувач може створити, відредагувати або переглядати звіт та історію дій над ним.

Логіку відображення сторінок ми будемо реалізовувати за допомогою пакету “react-router-dom”, який відслідковує зміну URL адреси, та показує користувачу ту, сторінку, яка відповідає цій адресі.

На *сторінці авторизації* (рис. 3.10) містяться такі функціональні компоненти: логотип з назвою системи, кнопка, поля для вводу даних.

Класовим компонентом та контейнером є сама форма авторизація, яка містить в собі логіку, яка працює наступним чином: коли користувач, вводить інформацію, та натискає на “Увійти”, ми робимо запит API і перевіряємо чи зареєстрований цей користувач, в позитивному випадку сервер повертає повну

інформацію про користувача, доступні макети та звіти. Якщо такого користувача не існує система видає відповідну помилку (рис. 3.11).

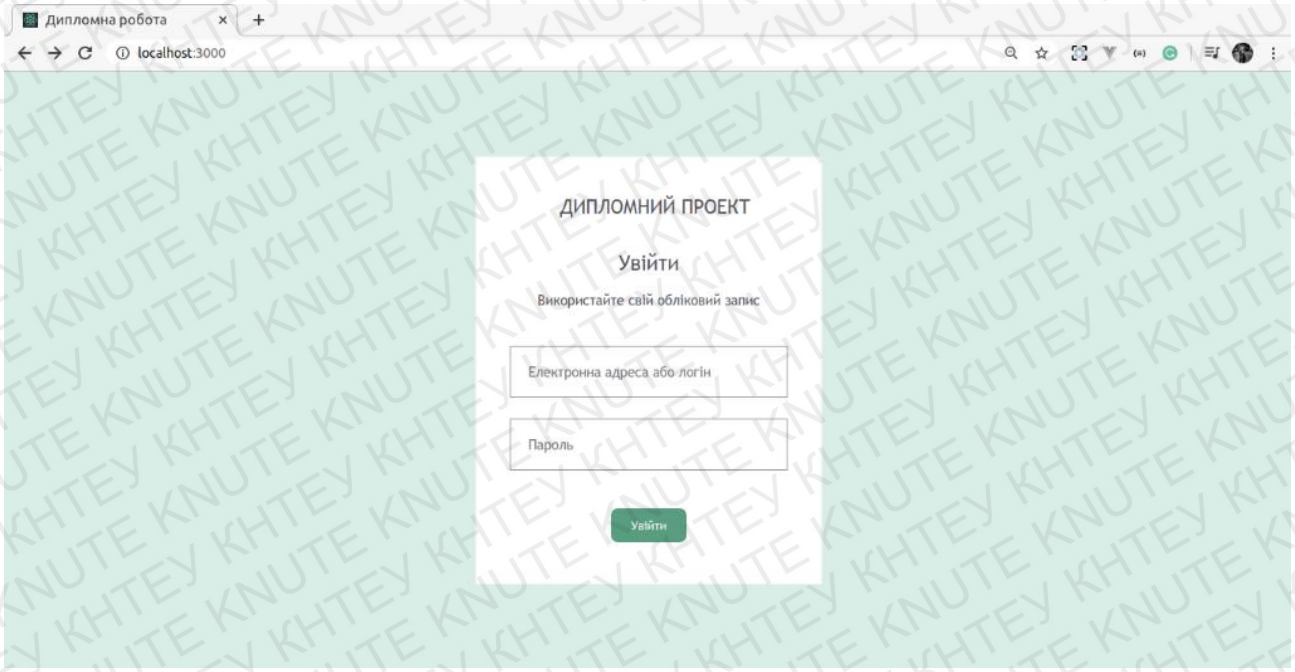

Рис. 3.10. Сторінка авторизації

Для того, щоб зменшити кількість запитів на сервер, в тих випадках, коли дані, наданні користувачем для авторизації, є очевидно некоректними, ми здійснюємо валідацію спочатку на клієнтській частині додатку на:

- Наповненість полів логіну та паролю;
- Кількість символів в паролі має бути більше трьох.

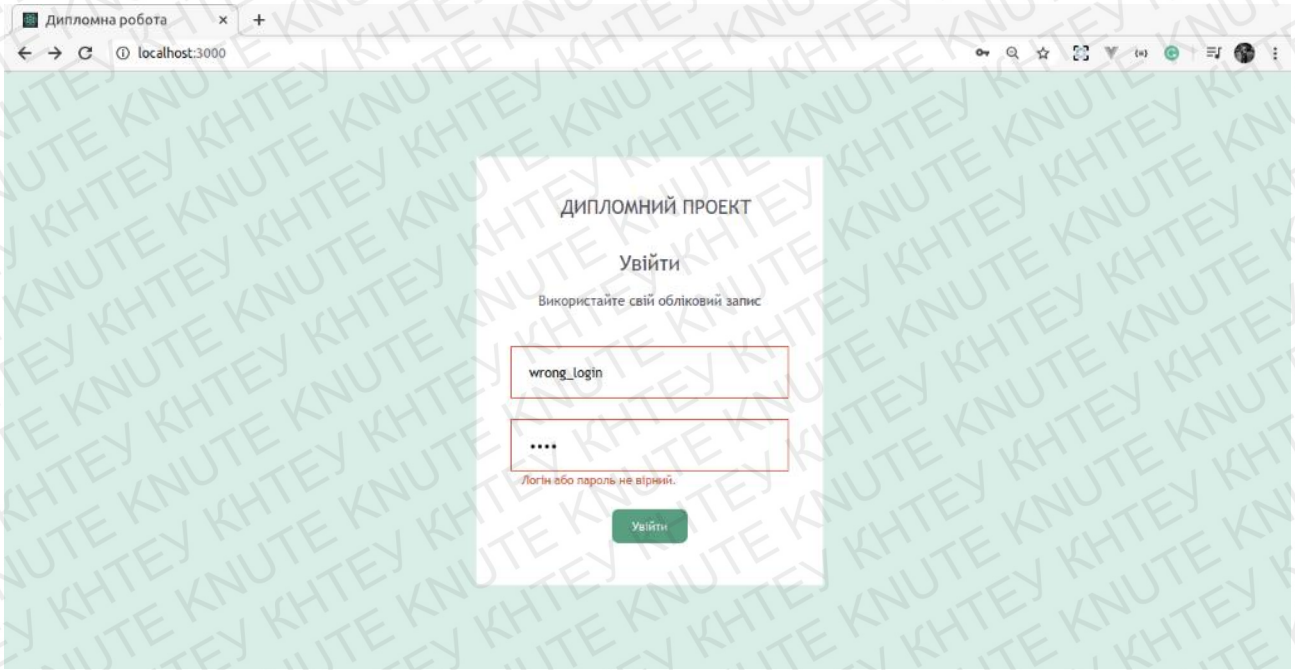

Рис. 3.11. Відображення помилки на сторінці авторизації

На *головній сторінці* (рис. 3.12) ми можемо бачити такі функціональні компоненти: кнопка, іконка користувача, модальне вікно, спливаюче меню.

До контейнерів на головній сторінці відносяться:

- Таблиця зі створеними макетами, яка відображає всі звіти, до яких користувач має доступ та інформацію про них. При кліку на рядок таблиці зі звітом, ми переходимо на сторінку вибраного звіту;
- При кліку на іконку користувача, у нас з'являється спливаюче вікно, в якому відображається ім'я та прізвище користувача і email. Кнопка "Вийти" завершує поточну сесію користувача і повертає його на сторінку авторизації;
- При натисканні на кнопку "Створити", з'являється модальне вікно (рис. 3.13), в якому користувач бачить макети звітів, які йому доступні. Користувач може обрати тільки один і натиснути "Створити" для переходу на сторінку звіту.

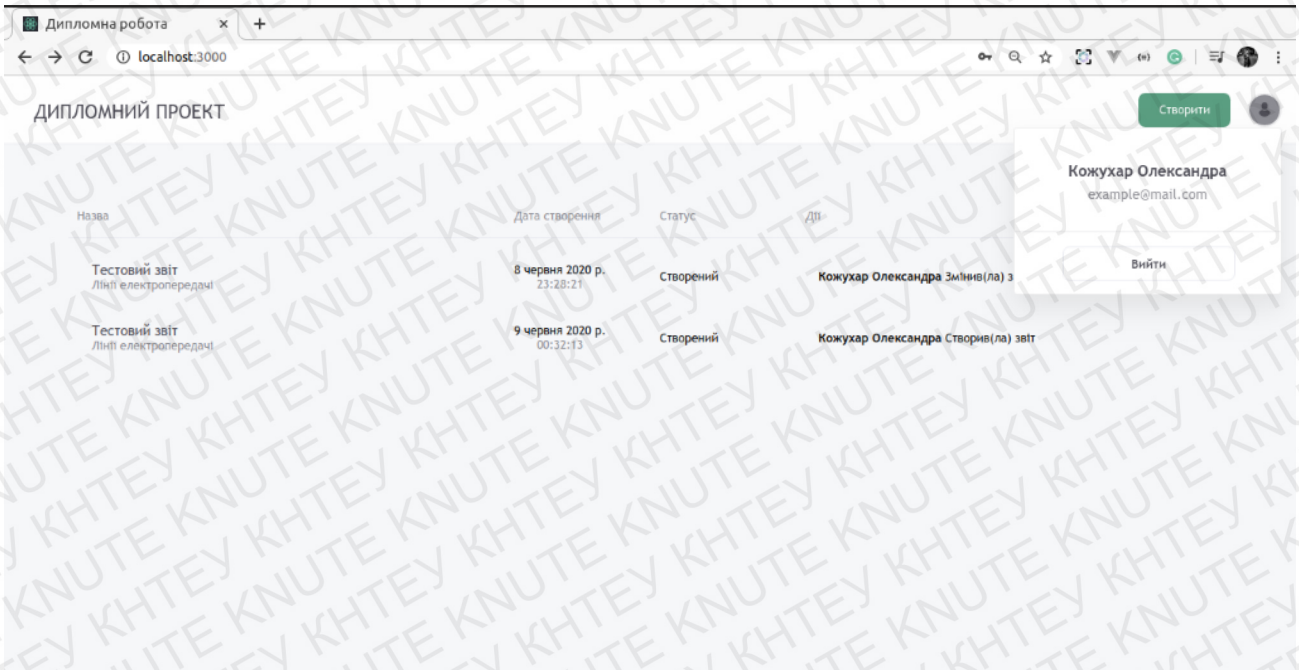

Рис. 3.12. Головна сторінка системи

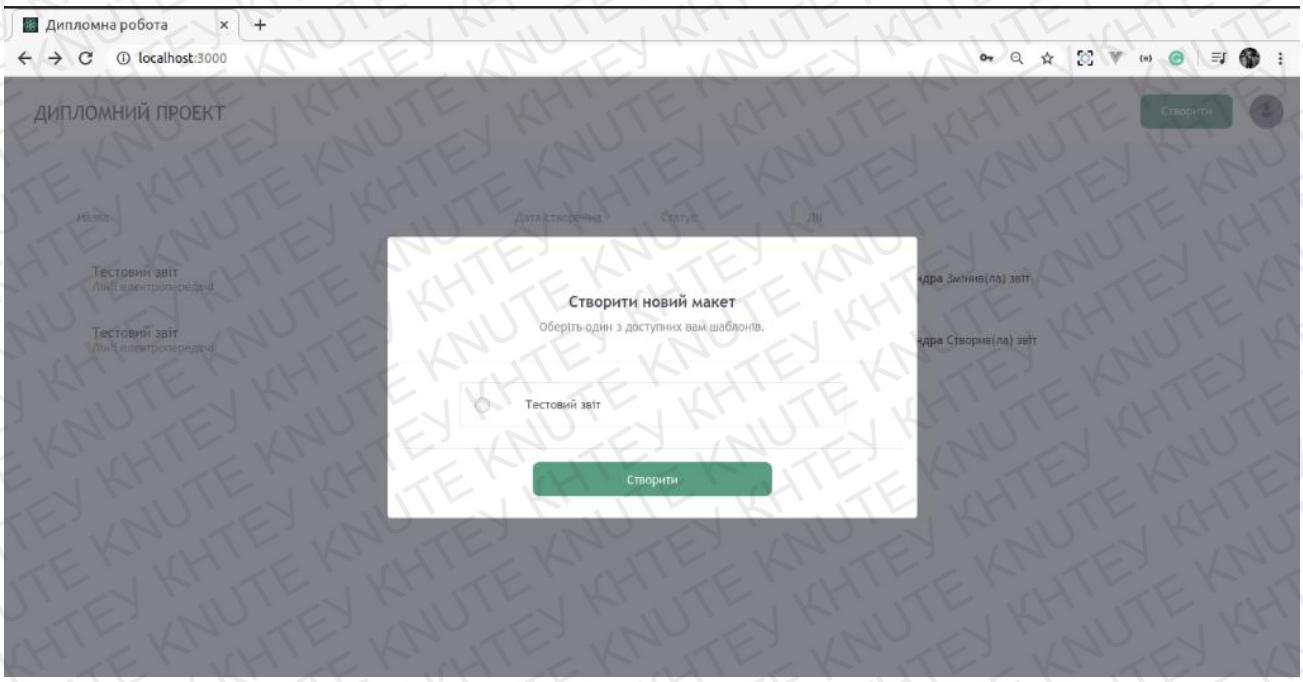

Рис. 3.13. Модальне вікно для створення звіту

В нашій системі передбачена одна сторінка для створення звіту і перегляду уже створених звітів.

На *сторінці звіту* (рис. 3.14) присутні такі функціональні компоненти: кнопки, іконки, поля для вводу даних.

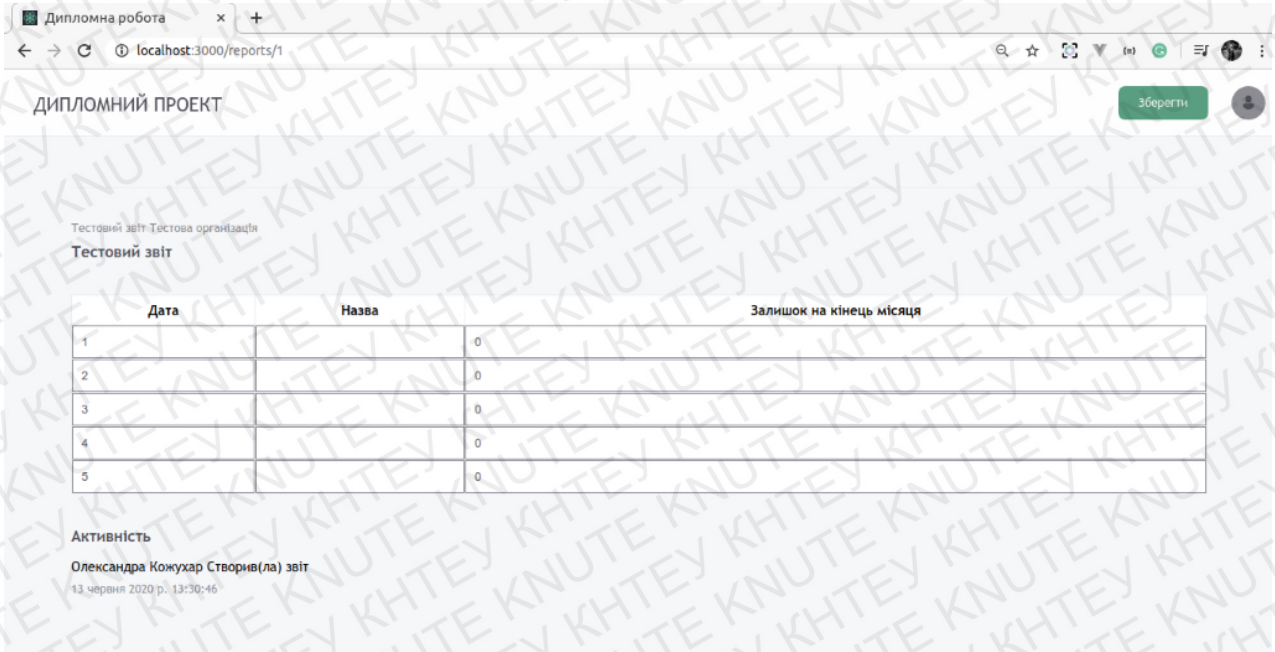

Рис. 3.14. Сторінка нового звіту

До класових компонентів ми відносимо:

- Історія звіту;

- Таблиця звіту. Один з найскладніших компонентів, який будується з окремих 2-х об’єктів:
    - комірок шапки звіту, серед яких знаходиться максимальне число колонок і рядків, відповідно до цього визначаються розміри таблиці. Потім по номеру рядку і номеру колонки сортуються всі комірки.
    - комірок тіла таблиці, які відбудовуються подібно до комірок в шапці. Потім ці 2 частини таблиці поєднуються в одну.

При натисканні на кнопки “Зберегти” на API відправляється id звіту та змінений об’єкт звіту.

Таким чином, ми спроектували архітектуру ієрархії в нашій системі електронного документообігу, такий підхід дозволяє суттєво пришвидшити процес розробки за рахунок повторного використання компонентів.

ВИСНОВКИ

У випускному кваліфікаційному проекті представлено результати теоретичних і прикладних досліджень, які мали на меті розробку сучасної та функціональної системи електронного документообігу. В межах даного проекту, було зібрано вимоги, спроектовано архітектуру та розроблено інтерфейс для збору звітності на умовному підприємстві. На основі проведених досліджень, отримуємо такі висновки:

1. Впровадження електронного документообігу на підприємстві є нелегким процесом, який потребує багато часу та ресурсів. Але все більше компаній в Україні обирає цей шлях, в першу чергу, завдяки нормативно-правовій базі, яка перебуває в постійному розвитку і поступовому впровадженні ЕДО в органах державного управління.
2. Проведений аналіз сучасних систем електронного документообігу в Україні показав, що цей ринок достатньо конкурентний і різноманітний. Завдяки цьому, на сьогоднішній день, підприємствам не потрібно витрачати купу коштів на розробку власної системи. Достатньо просто визначитись з вимогами та порівняти доступні варіанти.
3. Найважливішим етапом в розробці системи електронного документообігу є етап визначення вимог до майбутнього програмного забезпечення. В проекті ми дуже детально зупинились на цьому етапі, створивши діаграму користувача та прототипи майбутньої системи, що значно спростило подальшу розробку.
4. Для розробки системи, важливо обрати правильні технологій, які легко поєднуються між собою та доповнюють один одного. Сервісно-орієнтована архітектура, яка була обрана основою для розробки, ідеально працює з Django REST API та React. Найголовніше, що поєднання цих технологій дає нам можливість для подальшого масштабування системи, інтеграції з іншими корпоративними системами, аналізу даних, створення мобільного додатку системи, який буде працювати з Django REST API.

5. Створена система відповідає вимогам, які були поставлені до MVP. В подальшому для повноцінної роботи системи є необхідним розробка модулю регламенту, який самостійно створював би звіти з необхідною періодичністю, модулю оповіщень, який користувач міг би налаштувати самостійно і отримувати повідомлення з системи зручним для себе способом, також функціонал погодження для звітів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Про інформацію : Закон України від 2 жовтня 1992 р. No 2657-XII: [Електронний ресурс] – Режим доступу: http://www.rada.gov.ua
2. Копняк К. В. Автоматизація документообігу як складова підвищення ефективності діяльності підприємства / К. В. Копняк, Т. А. Костунець. // Економіка. Фінанси. Менеджмент: актуальні питання науки і практики. – 2017. – №11.
3. Про електронні документи та електронний документообіг : Закон України від 22 травня 2003 р. No 851-IV: [Електронний ресурс] – Режим доступу: http:// www.rada.gov.ua
4. Кадан А.М. Возможности системы электронного документооборота Alfresco для организации делопроизводства кафедры / А.М. Кадан, Р.В. Кизер // Технологии информатизации и управления : сб. науч. ст. – Минск : БГУ, 2011. – Вып. 2. – С. 388-392.
5. Прилипко Н.О. Вдосконалення системи електронного документообігу в органах державної влади / Н.О. Прилипко // Збірник наукових праць Донецького державного університету управління. Серія : Державне управління. – 2014. – Т. 15, Вип. 286. – С. 155-164.
6. M .E.Doc - своєчасна звітність та простий обмін документами. [Електронний ресурс]. - Режим доступу: https://medoc.ua.
7. СОТА: Звітність і документообіг. [Електронний ресурс]. - Режим доступу: https://sotabuh.com.ua.
8. FREDO. [Електронний ресурс]. - Режим доступу: https://fredo.com.ua.
9. Електронний Документообіг в Україні - Вчасно. [Електронний ресурс]. - Режим доступу: https://vchasno.ua.
10. FlyDoc. [Електронний ресурс]. - Режим доступу: https://flydoc.com.ua.
11. Закон України “Про електронний цифровий підпис”№ 852 від 22.05.2003 р. [Електронний ресурс] / Офіційний сайт Верховної ради України // Режим доступу: http://www.rada.gov.ua.

12. Про заходи щодо виконання у 2005 році Програми діяльності Кабінету Міністрів України "Назустріч людям": Постанова Кабінету Міністрів України від 6 травня 2005 р. N 324: [Електронний ресурс] – Режим доступу: http://www.rada.gov.ua.
13. Наказ Державної податкової адміністрації України “Про подання електронної податкової звітності” № 233 від 10.04.2008 р. (втратив чинність) [Електронний ресурс] / Офіційний сайт Верховної ради України // Режим доступу: http://www.rada.gov.ua.
14. Про затвердження Порядку засвідчення наявності електронного документа (електронних даних) на певний момент часу : Постанова Кабінету Міністрів України від 26 травня 2004 р. No 680: [Електронний ресурс] – Режим доступу: http://www.rada.gov.ua.
15. Про затвердження Порядку акредитації центру сертифікації ключів : Постанова Кабінету Міністрів України від 13 липня 2004 р. No 903: [Електронний ресурс] – Режим доступу: http://www.rada.gov.ua.
16. Про затвердження Положення про центральний засвідчувальний орган : Постанова Кабінету Міністрів України від 28 жовтня 2004 р. No 1451: [Електронний ресурс] – Режим доступу: http://www.rada.gov.ua.
17. Про затвердження Порядку застосування електронного цифрового підпису органами державної влади, органами місцевого самоврядування, підприємствами, установами та організаціями державної форми власності : Постанова Кабінету Міністрів України від 28 жовтня 2004 р. No 1452: [Електронний ресурс] – Режим доступу: http://www.rada.gov.ua.
18. Про затвердження Типового порядку здійснення електронного документообігу в органах виконавчої влади : Постанова Кабінету Міністрів України від 28 жовтня 2004 р. No 1453: [Електронний ресурс] – Режим доступу: http://www.rada.gov.ua.
19. Про затвердження порядку обов’язкової передачі документованої інформації : Постанова Кабінету Міністрів України від 28 жовтня 2004 р. No 1454: [Електронний ресурс] – Режим доступу: http://www.rada.gov.ua.

20. Іванов А. І. НОРМАТИВНО-ПРАВОВЕ РЕГУЛЮВАННЯ ВИКОРИСТАННЯ ЕЛЕКТРОННОГО ЦИФРОВОГО ПІДПИСУ / А. І. Іванов. // ВІСНИК ПЕНІТЕНЦІАРНОЇ АСОЦІАЦІЇ УКРАЇНИ. – 2018. – №1.
21. Django. [електронний ресурс]. - Режим доступу: https://www.djangoproject.com/
22. Hourieh A. Learning Website Development with Django / Ayman Hourieh. – Olton: Packt Publishing, 2008. – 248 P.
23. Holovaty A. The Definitive Guide to Django Web Development Done Right / A. Holovaty, J. Kaplan-Moss. – New York: Apress, 2008. – 447 P.
24. Використання RESTful [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: http://edu.asu.in.ua/mod/book/tool/print/index.php?id=117.
25. Django Rest Framework [Електронний ресурс] – Режим доступу до ресурсу: https://www.django-rest-framework.org/.
26. Angular. [Електронний ресурс] – Режим доступу до ресурсу: https://angular.io/features
27. Hebda A. React vs. Angular. Battle for the Front-End [Електронний ресурс] / Hebda Artur. – 2019. – Режим доступу до ресурсу: https://railsware.com/blog/react-vs-angular-battle-for-the-front-end/.
28. Reactjs. [Електронний ресурс] – Режим доступу до ресурсу: https://reactjs.org/.
29. MongoDB. [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/
30. Jeang-Kuo C. An Introduction of NoSQL Databases Based on Their Categories and Application Industries / C. Jeang-Kuo, L. Wei-Zhe. // Algorithms. – 2019. – №12.
31. Гладкикх Д. М. Паттерны: MVC, MVP и MVVM [Електронний ресурс] / Д. М. Гладкикх – 2010. – Режим доступу: https://www.outcoldman.com/ru/archive/2010/02/22/паттерны-mvc-mvp-имvvm/

32. Ришковець Ю. В. АЛГОРИТМІЗАЦІЯ ТА ПРОГРАМУВАННЯ / Ю. В. Ришковець, В. А. Висоцька. – Львів: Новий світ – 2000, 2017. – 460 с.
33. Дизайн система державних сайтів України [Електронний ресурс] – Режим доступу: https://design.gov.ua/ua/proektiroaie-i-prototipirovanie/testuvannya-prototipiv.
34. Hettiarachchi N. Common Software Architectural Patterns you need to know [Електронний ресурс] / Nethmi Hettiarachchi. – 2019. – Режим доступу: https://medium.com/@nethmihettiarachchi484/common-software-architectural-patterns-in-a-nutshell-7df312d3989c.
35. SQL Data Types for MySQL, SQL Server, and MS Access [Електронний ресурс] – Режим доступу: https://www.w3schools.com/sql/sql_datatypes.asp.
36. SQL Data Types [Електронний ресурс] – Режим доступу: https://www.journaldev.com/16774/sql-data-types.
37. What is a Foreign Key? [Електронний ресурс]. – 2016. – Режим доступу: https://database.guide/what-is-a-foreign-key/.
38. Будівельник [Електронний ресурс] – Режим доступу до ресурсу: https://refactoring.guru/uk/design-patterns/builder.
39. Sufiyan T. What is React? [Електронний ресурс] / Taha Sufiyan. – 2020. – Режим доступу до ресурсу: https://www.simplilearn.com/what-is-react-article.