Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
Кафедра Програмної інженерії

АТЕСТАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти – другий (магістерський)

Дослідження методів проєктування програмних систем для автоматизованої генерації цифрової звітності підприємства. UX/UI проєктування та розробка відкритого API

Виконав: студент 2 курсу, групи ІПЗм-18-4
Кривко Д.В.

спеціальності 121 – Інженерія програмного забезпечення
Освітньо-наукової програми
Інженерія програмного забезпечення

Керівник проф. каф. ПІ Білоус Н.В.

Допускається до захисту
Зав. кафедри, проф. _____ З.В. Дудар

2020 р

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет _Комп'ютерних наук_____
Кафедра _Програмної інженерії_____

Рівень вищої освіти - другий (магістерський)
Спеціальність _121-Інженерія програмного забезпечення_____
(код і повна назва)
Тип програми _____освітньо-наукова програма_____
Освітня програма _Інженерія програмного забезпечення_____

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові _Кривку Дмитру Володимировичу_____
(прізвище, ім'я, по батькові)

1.Тема роботи _ Дослідження методів проєктування програмних систем для автоматизованої генерації цифрової звітності підприємства. UX/UI проєктування та розробка відкритого API

затверджена наказом університету від "27" березня 2020 р № 473 Ст

2. Термін подання студентом роботи до екзаменаційної комісії
19 травня 2020 р.

3. Вихідні дані до роботи методи проєктування програмних систем, цифрова звітність, пояснювальна записка, система електронного документообігу, UI/UX.

4. Перелік питань, що потрібно опрацювати в роботі дослідити методи проєктування програмних систем для автоматизованої генерації цифрової звітності підприємства, проаналізувати підходи до проєктування UI/UX, розробити відкрите API для інтеграції системи електронного документообігу зі сторонніми сервісами.

5. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім’я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Спецчастина	проф. каф. ПІ Білоус Н.В.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка*
1	Аналіз предметної галузі	10 квітня 2020 р.	виконано
2	Огляд методів проєктування програмних систем	15 квітня 2020 р.	виконано
3	Моделювання архітектури UI та API	20 квітня 2020 р.	виконано
4	Підготовка пояснювальної записки	23 квітня 2020 р.	виконано
5	Спецчастина	01 травня 2020 р.	виконано
6	Підготовка презентації та доповіді	04 травня 2020 р.	виконано
7	Попередній захист	07 травня 2020 р.	виконано
8	Нормоконтроль, рецензування	09 травня 2020 р.	виконано
9	Занесення диплома в електронний архів	12 травня 2020 р.	виконано
10	Допуск до захисту у зав. кафедри	18 травня 2019р.	

Дата видачі завдання _____ _____2020 р.

Студент _____ Кириво Д.В.
(підпис)

Керівник роботи _____ проф. каф. ПІ Білоус Н.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ /ABSTRACT

Атестаційна робота магістра містить 78 с., 20 рис., 13 джер.

СИСТЕМА ДОКУМЕНТООБІГУ, ЦИФОРВА ЗВІТНІСТЬ, АНАЛІЗ СИСТЕМ, WEB-ПЛАТФОРМА, ГЕНЕРУВАННЯ ДОКУМЕНТІВ, КРОССПЛАТФОРМНІСТЬ, КОРИСТУВАЦЬКИЙ ІНТЕРФЕЙС, REST API.

Об’єкт дослідження – методи проєктування програмних систем для автоматизованої генерації цифрової звітності підприємства.

Мета роботи – аналіз методів проєктування програмних систем для автоматизованої генерації цифрової звітності підприємства малого та середнього бізнесу, дослідження методів проєктування користувацького інтерфейсу, проєктування та розробка UI системи електронного документообігу та програмна реалізація відкритого API для можливості інтеграції системи зі сторонніми сервісами та додатками.

Метод рішення – Web-платформа, JetBrains PhpStorm 2020, мова PHP та JavaScript, HTML5, CSS3.

В результаті роботи проведено дослідження та аналіз методів проєктування програмних систем автоматизації документообігу, порівняно аналоги на ринку та створено користувацький інтерфейс та інтерфейс взаємодії із зовнішніми додатками для системи автоматизації документообігу, що дозволяє легко організувати електронний документообіг на підприємстві.

Додаток може бути розміщений на хмарній платформі, що дозволяє істотно знизити витрати на комп'ютеризацію, підприємства.

DOCUMENTATION SYSTEM, DIGITAL REPORTING, SYSTEM ANALYSIS, WEB-PLATFORM, DOCUMENT GENERATION, CROSS-FORMALITY, USER INTERFACE, REST API.

Object of study - methods of designing software systems for automated generation of digital reporting of an enterprise.

The purpose of the work is to analyze the methods of designing software systems for automated digital reporting of small and medium enterprises, research methods of user interface design, design and development of UI electronic document management system and software implementation of open API to integrate the system with third-party services and applications.

Solution method - Web platform, JetBrains PhpStorm 2020, PHP and JavaScript language, HTML5, CSS3.

As a result, the research and analysis of methods for designing software automation systems, compared analogues in the market and created a user interface and interface for interaction with external applications for document automation, which allows you to easily organize electronic document management in the enterprise.

The application can be placed on a cloud platform, which can significantly reduce the cost of computerization, enterprise.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

AJAX – asynchronous JavaScript and XML;

API – application programming interface;

DOM – document object model;

ECM – enterprise content management;

JSON – JavaScript object notation;

LAMP – зв’язка Linux + Apache + MySQL + PHP;

UI – user interface;

UML – unified modeling language;

URL – uniform resource locator;

UX – user experience;

WAMP – зв’язка Windows + Apache + MySQL + PHP;

ЕДО – електронний документообіг;

ІКТ – інформаційно-комунікаційні технології;

ОРД – організаційно-розпорядчий документообіг;

СЕД – система електронного документообігу;

СУБД – система управління базами даних.

ВСТУП

Питання про необхідність автоматизації управління документообігом давно перейшло у практичну площину, і все більше українських підприємств впроваджують у себе системи електронного документообігу (СЕД), дозволяючи організаціям вже на власному досвіді оцінити переваги нової технології роботи з документами. Однак і для тих небагатьох, хто вважає автоматизацію документообігу пройденим етапом, можливо, незабаром буде потрібно переосмислити зроблений вибір і знову зануритися в проблему підвищення ефективності управління документообігом. Це обумовлюється, зокрема, зміною ринкової ситуації, зростанням організації, що створює кризи «перехідного віку» і призводить до необхідності реструктуризації, а також розвитком інформаційно-комунікаційних технологій (ІКТ), з одного боку, надають нові можливості для ведення бізнесу, з іншого - які змушують йти в ногу з часом, щоб не відстати від конкурентів.

Необхідність в автоматизації управління документообігом різні організації сьогодні бачать по-різному: одні - в підвищенні ефективності організаційно-розпорядчого документообігу (ОРД), інші - в підвищенні ефективності роботи функціональних фахівців, що створюють документи і використовують їх у повсякденній роботі, і лише деякі приділяють увагу обом аспектам. Такий поділ точок зору в питаннях документообігу визначається різною роллю і значимістю самих документів в діяльності організації, що залежить від розміру організації, стилю управління, галузі виробництва, загального рівня технологічної зрілості і багатьох інших факторів. Тому для одних документ може бути, наприклад, базовим інструментом управління, а для інших - засобом і продуктом виробництва.

Незалежно від того, чим визначається інтерес до документоорієнтованих інформаційних систем (ДІС), всі організації починають з вибору відповідної системи з незліченної безлічі, присутнього на українському ринку. Організації, яка вирішила серйозно підійти до цього питання, вибір доведеться почати з розстановки крапок над «і» в термінології виробників документоорієнтованих

інформаційних систем, давно стала об'єктом спекуляції внаслідок невизначеності і універсальності таких понять, як «документообіг», «діловодство», «архів» і ін. Що б не стояло за назвою тієї чи іншої ДІС, їх можна розділити на три основні групи: системи діловодства, системи документообігу та системи управління документами.

Доцільністю роботи є фактична необхідність подібних систем в малому та середньому бізнесі, а також майже повна відсутність систем з відкритим похідним кодом на ринку, що в деякій мірі обмежує організації в розширенні систем документообігу для своїх специфічних цілей.

Метою роботи є аналіз методів проєктування програмних систем для автоматизованої генерації цифрової звітності підприємства малого та середнього бізнесу, дослідження методів проєктування користувацького інтерфейсу, проєктування та розробка UI системи електронного документообігу та програмна реалізація відкритого API для можливості інтеграції системи зі сторонніми сервісами та додатками.

Об’єктом дослідження є методи для проєктування користувацького інтерфейсу та методи інтеграції систем документообігу із зовнішніми додатками.

Розроблена система дозволяє спростити роботу з електронною документацією організації завдяки простоті інтерфейсу. Також система має можливість інтеграції зі сторонніми сервісами за допомогою відкритого API, завдяки чому система може масштабуватись та модифікуватися під конкретні цілі підприємства.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Електронний обмін даними - це реальність, з якою сьогодні стикається практично кожен. Інформаційні системи, комп'ютерні мережі, електронна пошта - ось далеко не повний перелік тих засобів, за допомогою яких відбувається обмін даними в електронному вигляді.

В останнє десятиліття з'явилися і набули поширення нові інструментальні засоби ефективного забезпечення управлінських процесів. У тому числі мова йде про програмне забезпечення, призначене для обробки управлінських документів. Тут перш за все слід згадати програмне забезпечення класів "системи управління документами" і "системи управління бізнес-процесами"[1].

Такі системи є програмними комплексами, які застосовуються для вирішення ряду завдань, в тому числі і для побудови корпоративних систем електронного документообігу. В рамках автоматизації процесу обробки документа в організації з моменту його створення або отримання до моменту відправки кореспонденту або завершення виконання і списання в справу має бути забезпечено вирішення наступних функцій:

– реєстрація вхідних в організацію документів, що виходять з організації документів і внутрішніх документів;

– облік резолюцій, виданих за документами керівництвом організації, і постановка документів на контроль;

– централізований контроль обігу документів;

– списання документів у справу;

– ведення інформаційно-довідкової роботи;

– формування діловодних звітів по організації в цілому.

Використання системи електронного документообігу дозволяє організувати передачу даних про хід обігу документів в електронному вигляді, що якісно змінює організацію контролю роботи з документами. Картки централізовано

зареєстрованих документів з резолюціями керівництва розсилаються в електронному вигляді співробітникам відповідних підрозділів. Вони доповнюють їх резолюціями по виконанню документів, що видаються керівниками структурних підрозділів. У міру появи даних про хід обігу документів ці дані вносяться в систему.

Також значно видозмінюється процес узгодження проектів документів, в рамках якого співробітники, які беруть участь в процесі узгодження, отримують можливість обмінюватися електронними версіями узгоджуваних проектів. Така технологія дозволяє скоротити час, що витрачається на передачу проектів в паперовому вигляді.

Система електронного документообігу обов'язково включає поточний електронний архів, який вирішує проблеми оперативного доступу до інформації та наявності можливості одночасного використання документа декількома співробітниками. Така форма організації зберігання значно знижує ймовірність втрати інформації і підвищує оперативність роботи за рахунок скорочення часу пошуку потрібного документа. Зберігання текстів документів в електронному вигляді дозволяє реалізовувати повнотекстовий пошук, що відкриває принципово нові можливості при веденні інформаційно-довідкової роботи, наприклад, дозволяє робити тематичні добірки документів за їх змістом. Використання електронного архіву позбавляє від необхідності створювати фонд користування архівних документів, так як за запитом будь-якої миті може бути видана електронна копія документа.

З юридичної точки зору поняття електронного документообігу відрізняється від поняття електронного обміну даними. В основі першого лежить легітимність (процесуальна допустимість і доказова сила) електронних документів. Тому поряд з удосконаленням інформаційних технологій важливу роль в процесі створення інфраструктури електронного документообігу повинна зіграти його законодавча підтримка, суть якої полягає в наданні даними, створюваним і переданим електронним способом, юридичного статусу документа.

Основною функцією традиційного документа є засвідчення деякою інформацією. При складанні і використанні документа присутні два аспекти: по-перше, деяка інформація, а по-друге, - сам документ як матеріальна річ, яку можна пред'явити або передати. Наявність цієї матеріальної речі дозволяє підтвердити істинність інформації, що міститься в документі. Можливо, для підтвердження істинності необхідно проробити певну процедуру - експертизу по перевірці автентичності документа[2].

Саму інформацію, що міститься в документі, теж можна розділити на дві частини. Перша частина - безпосередньо зміст, друга - допоміжна інформація, яка дає можливість встановити його автентичність (справжність). До неї відносяться реквізити типу вихідного номера, підписів і печаток.

До складу інформації, як змістовної, так і про носії, можуть входити і дані про час, умови та місце складання документа.

Необхідно також відзначити, що в разі паперового документа оригінал існує в обмеженому, відомому заздалегідь кількості примірників. Наприклад, може бути зазначено, що договір укладений у трьох примірниках, які мають однакову силу. Будь-який додатковий примірник є копією, що в принципі може бути перевірено шляхом проведення відповідної експертизи.

У ряді випадків істотно наявність саме оригіналу документа. Наприклад, продаж акції, випущеної в документарній формі, зовсім не рівнозначна продажу копії її сертифіката, навіть завіреної нотаріально.

Таким чином, документ виконує наступні функції:

– фіксація деякої (змістовної) інформації;

– фіксація особи, яка підписала документ;

– фіксація умов складання документа;

– доказ в судовому розгляді;

– функція оригіналу, що забезпечується його унікальністю.

З точки зору традиційного документообігу можна виділити дві основні функції паперового документа: інформаційну і доказову (тобто можливість використовувати його в якості допустимого доказу). Головною причиною, за якою

саме паперові документи виконують ці функції, є те, що саме папір був протягом багатьох століть найбільш поширеним матеріальним носієм, використовуваним для передачі і зберігання інформації. В останні десятиліття ситуація різко змінилася, обсяги переданих в електронному вигляді даних стрімко ростуть. Як зазначалося раніше, системи безпаперового документообігу отримують все більш широке поширення в самих різних областях. У зв'язку з цим важливого значення набуває визначення правового статусу електронного документа - окреслення областей, де можливо і допустимо його застосування.

Документ - письмовий акт встановленої чи загальноприйнятої форми, складений певними та компетентними установами, підприємствами, організаціями, посадовими особами, а також громадянами для викладу відомостей про факти або посвідчення фактів, що мають юридичне значення, або для підтвердження прав і обов'язків[5].

Вимоги до документу, що випливають з наведеного визначення, можна розділити на три групи. Перша відображає інформаційну функцію документа: документом може бути не будь-яка інформація, зафіксована на паперовому носії, а тільки відомості певного характеру (ця вимога узгоджується з наведеними раніше визначенням ЕДО). Друга (вимоги до форми) група - це, по суті, вимоги, що забезпечують доказову функцію документа (реквізитами форми можуть служити наявність печатки та підпису певної особи, персональні дані про особу, видає документ, а також вимоги до паперового носія, наприклад, папір з захисними знаками і т. п.). Третя група (компетентність джерела документа) як би зв'язує перші дві, надає юридичну значимість документу. Документ, виданий некомпетентним органом, підписаний не уповноваженою на те особою або анонімний, не може служити підтвердженням викладених у ньому відомостей про факти, засвідчувати факти або підтверджувати права і обов'язки.

Очевидно, легко забезпечити для даних, записаних в комп'ютерному форматі, виконання умов першої та третьої груп. Деякі з вимог до форми документа (наприклад, дотримання певної послідовності викладу змісту і розташування тексту) теж можуть бути дотримані. Інші вимоги другої групи (наявність печатки

організації, власноручного підпису особи, спеціальний тип паперу) принципово неприйнятні для електронних документів внаслідок специфічної природи комп'ютерних носіїв інформації.

Саме фізичні характеристики електронних документів довгий час були об'єктом критики противників безпаперових систем документообігу. Зокрема, в якості одного з аргументів наводилося наступне твердження: те, що написано на папері, важко видалити і воно залишається навічно; дані же на комп'ютерних носіях можуть бути легко знищені, вони недовговічні. Але, по-перше, збереження паперових документів в значній мірі залежить від якості паперу і для їх тривалого зберігання необхідно застосування спеціальних заходів, а по-друге, сучасні носії комп'ютерних даних дозволяють зберігати інформацію тривалий час і при здійсненні відповідних заходів безпеки (в тому числі і періодичне копіювання) їх надійність не нижче, ніж у традиційних. Крім того, коли мова йде про ділову інформацію, зазвичай існують певні терміни зберігання такої інформації, що обчислюються роками або десятиліттями, а протягом зазначених строків можливе збереження інформації на магнітних носіях, компакт-дисках не кажучи уже про хмарні сховища які дозволяють зберігати інформацію майже вічно[4].

Ще один аргумент, що приводиться в користь паперового документа, полягає в тому, що він помітний (тобто будь-хто може фізично перевірити наявність документа), кожна грамотна людина може прочитати такий документ. Те, що електронний документ не може бути не посередньо сприйнятий людиною, не є непереборною складністю. Теоретично, звичайно, можна припустити, що створена система ЕДО, в якій один учасник вносить в документ Manchester, а інший отримує на екрані монітора або при роздруківці Liverpool. Однак дана проблема легко вирішується, якщо припустити, що існує узгоджена учасниками або певна нормативним актом уповноваженого органу процедура виготовлення за електронною оригіналу традиційної (паперової) копії документа. Більш докладно про це ми поговоримо, коли будемо розглядати процедури вирішення конфліктів, пов'язаних з використанням ЕДО. Паперовий документ майже неможливо змінити, в електронний же документ легко внести поправки, і дуже важко потім довести

факт їх внесення. Можна, звичайно, відзначити, що підробка традиційних документів має, напевно, не меншу історію, ніж історія писемності, але це не знімає проблеми ідентифікації електронних документів.

Вирішити прийнятним чином це завдання вдалося тільки в другій половині 1970-х років, коли американські математики У. Діффі і М. Е. Хеллмен запропонували використовувати цифровий підпис для підтвердження автентичності електронних повідомлень.Таким чином електронний документ можна визначити як набір даних, записаних в компьютерно-зчитуванному вигляді, для яких виконана така умова: існує визнана учасниками ЕДО або затверджена компетентним органом процедура, що дозволяє однозначно перетворити ці дані в документ традиційного режиму. Визнання зазначеної процедури має бути підтверджено учасниками системи ЕДО за допомогою традиційного (письмового) документа, або така процедура повинна бути санкціонована уповноваженим державним органом. Необхідність традиційного документа або акта уповноваженого органу для визнання процедури перетворення пояснюється тим, що в іншому випадку можливий порочне логічне коло, коли питання визнання чи невизнання юридичної сили електронного документа будуть вирішуватися на підставі іншого електронного документа, силу якого теж можна оскаржити.

1.2 Аналіз рішень для автоматизації документообігу

У виборі системи електронного документообігу сучасні компанії керуються загальною стратегією розвитку, цілями, наявністю конкурентного середовища, бажаною структурою і очікуваним економічним ефектом від впровадження такого рішення. До цілей впровадження СЕД можна віднести поліпшення контролю виконавчої дисципліни, скорочення числа втрачених документів; скорочення часу узгодження; зменшення кількості помилок в роботі з типовими документами.

Є ряд ключових вимог до функцій СЕД (ECM). Від відповідності системи цим вимогам залежить подальший успіх оптимізації документообігу компанії. Процеси узгодження документів і призначення завдань виконуються швидше, коли переведені з «паперового» в електронний вигляд, також скорочується час на обробку документів і доручень, і з'являється можливість відстежувати хід роботи з документом. При роботі з системою виконавці сповіщаються про нові документи автоматично, а терміни їх обробки знаходяться під контролем. Для швидкого доступу до документів, легкого пошуку і збереження документів організовується електронне сховище.

Важливо, щоб права доступу до захищених даних були розмежовані. Значно скорочує час роботи і автоматичне заповнення розділів типових документів за існуючими довідковими даними. Керівнику важливо мати зручні засоби контролю термінів виконання завдань і зведену звітність. Для підтримки інформативності в роботі компанії СЕД повинна легко інтегруватися з існуючою поштовою системою і з існуючими в компанії обліковими системами (кадровими, фінансовими, бухгалтерськими та системами управління виробничою діяльністю.

Також все більше організацій звертають увагу на можливість віддаленої роботи в системі. До важливих критеріїв оцінки системи відносяться можливість формування звітності по документам, виконавцям, статусам документів та ін .; швидке впровадження системи; вартість установки і підтримки системи; простота розвитку системи; можливість використання програмного забезпечення системи для вирішення додаткових завдань.

Додаткові вимоги:

– наявність потокового введення документів в систему, можливість роботи зі сканером;

– вбудований модуль управління договорами;

– планшетний версія;

– мобільні клієнти.

Проаналізуємо існуючі системи на українському ринку программного забезпечення.

FossDoc (рис.1.1) [6] — рішення на платформі FossLook, призначене для створення електронного архіву документів, організації корпоративного документообігу (workflow) і автоматизації бізнес-процесів на підприємствах, в установах і організаціях будь-якого роду діяльності. Програма дозволяє вирішити велику кількість завдань, реалізація яких покладена на відповідні модулі. Система може бути легко перелаштована з урахуванням специфіки роботи кожного конкретного підприємства.

Система автоматизує всі аспекти сучасного діловодства: створення реєстраційно-контрольних карток, РКК (карток документа); відправку доручень (аналог "бігунка" в звичайному документообігу); облік паперових оригіналів за допомогою спеціальних журналів; здійснення контролю над виконанням документів; підготовку резолюцій; роботу з електронно-цифровим підписом, генерацію звітів.

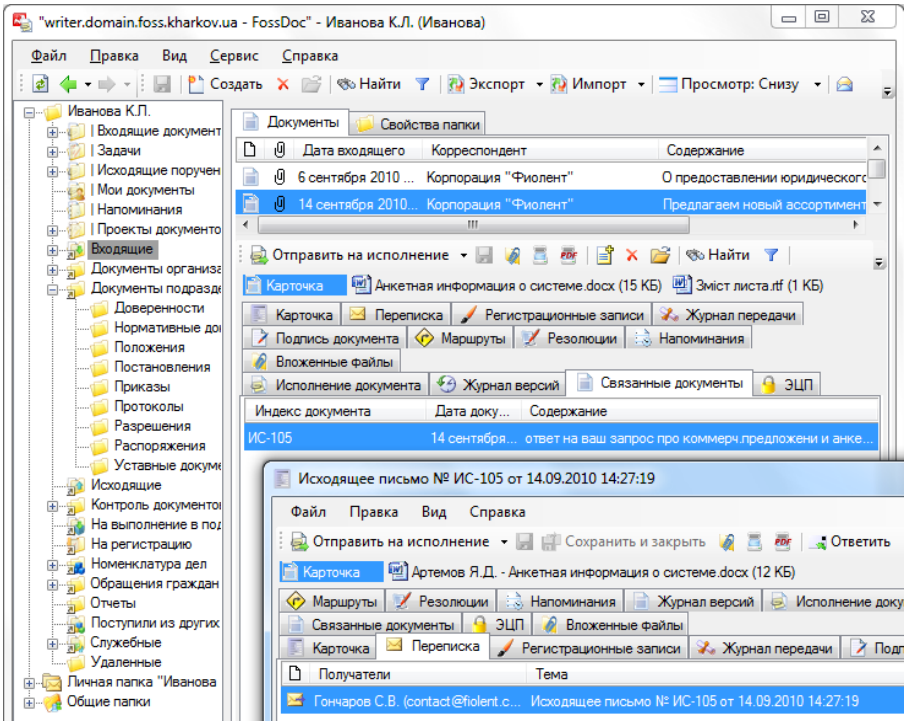

Рисунок 1.1 – Інтерфейс системи FossDoc.

Все різноманіття документів, з якими працюють користувачі, розділяється на окремі категорії - типи документів. В системі існують зумовлені типи документів (поширені у вітчизняному діловодстві): вхідні та вихідні листи, звернення громадян, службові записки, накази і т.д. Документи можуть посилатися на інші документи або бути дочірніми по відношенню до головних. Поля документів можуть заповнюватися значеннями з довідників.

elDoc (рис.1.2.) [7] - це високотехнологічне рішення рівня Enterprise, що розроблене на сучасних web 2.0 технологіях для автоматизації процесів обміну документами та організації електронного документообігу.

Вбудовані конструктори процесів та форм документів дозволяють створювати нові форми документів та маршрути процесів не залучаючи розробників. Конструктор процесів дозволяє встановлювати та змінювати логіку процесу як на рівні адміністратора система, так і рівні ініціатора документу, а форми документів легко кастомізуються під вимоги організації.

Система elDoc дозволяє оперативно, в режимі реального часу, відстежувати статус документу, здійснювати контроль виконання доручень та будувати звітність. Система elDoc підтримує функціонал автоматичних нагадувань про події в системі шляхом надсилання email-повідомлень. Навіть якщо Ви випустили з уваги щось важливе, система elDoc сповістить користувачів про необхідність виконати дії згідно встановлених резолюцій.

Архів та потужна система пошуку документів. Архівування документів за заданими параметрами згідно маршрутів проходження документів. Миттєвий пошук картки документу за сканованим штрих-кодом або ж контекстний пошук дозволить знайти оперативно потрібний документ або фрагмент запису серед сотень тисяч документів, зареєстрованих в системі elDoc.

elDoc надає взаємопов'язаний з електронним документообігом CRM функціонал: база контрагентів, клієнтів, їх контактні дані, облік дій в розрізі кожного контрагента, нагадування про необхідність виконати перелік дій по контрагенту.

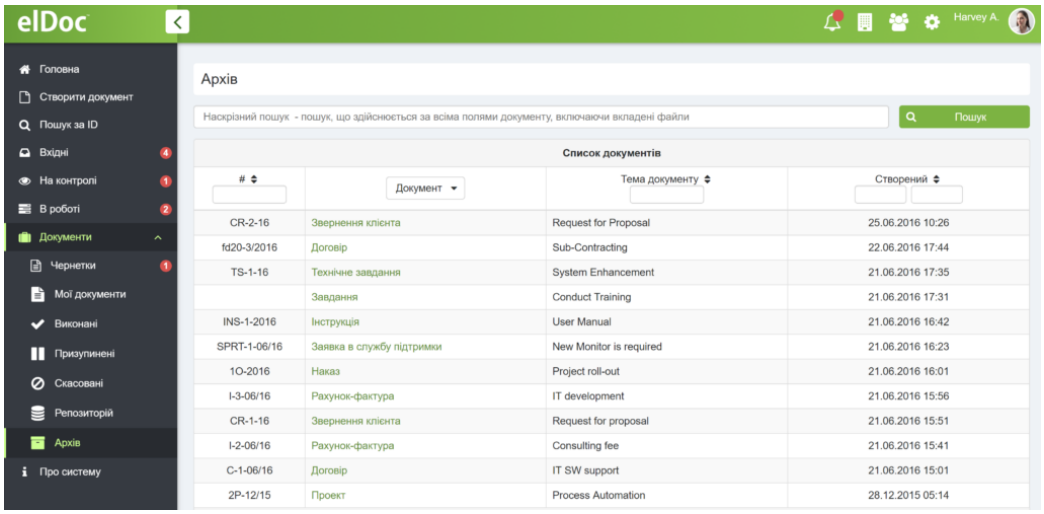

Рисунок 1.2 – Інтерфейс системи elDoc.

Optima WorkFlow (рис.1.3) [7] - є комплексною прикладною платформою для створення рішень в галузі управління документами, що інтегрує в себе інформаційні технології провідних зарубіжних виробників, в тому числі засоби криптографічного захисту інформації, засновані на міжнародних стандартах.

Закладений в основу розробки продукту інтеграційний підхід, заснований на застосуванні готових до використання і перевірених в ході тривалої експлуатації програмних компонентів, дозволяє в ході експлуатації «OPTIMA-WorkFlow» застосовувати можливості програмних рішень та інформаційних технологій, що здобули собі репутацію лідерів в своїй галузі.

Застосування в якості серверного компонента «OPTIMA-WorkFlow» сучасних систем управління базами даних (Microsoft SQL Server або Oracle Database) забезпечує всі переваги використання «клієнт-серверної» архітектури, надійну і стійку взаємодію з ресурсами обчислювальних мереж і операційних систем, здатність до масштабування рішення і до функціонування в розподілених корпоративних обчислювальних мережах.

Використання в якості призначеного для користувача інструментарію для створення і обробки документів, формування звітності і т.п. сучасних офісних додатків, в тому числі, програмних продуктів Microsoft Office, OpenOffice.org і Business Objects Crystal Reports, забезпечує простоту використання добре освоєних

коштів обробки документів з властивою їм функціональної потужністю і гнучкістю.

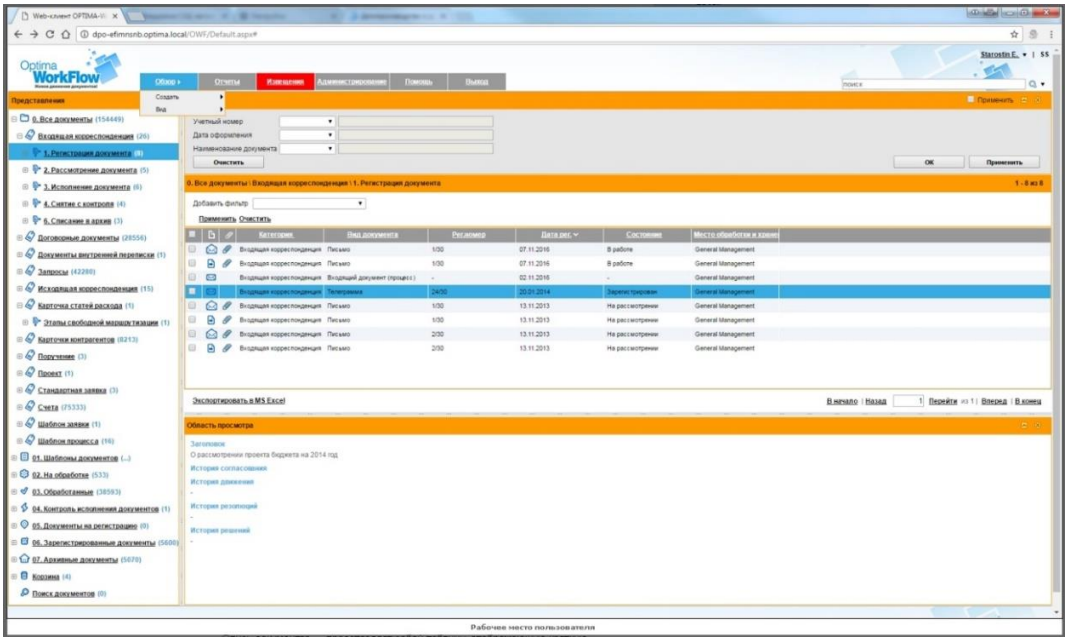

Рисунок 1.3 – Інтерфейс системи Optima WorkFlow.

Проаналізовані системи мають багато надлишкового функціоналу який зазвичай не використовується в малому та середньому бізнесі, через що дані системи мають високу вартість та здебільшого мають нагромаджений користувацький інтерфейс, що робить їх не простими для швидкого освоєння непідготовленими користувачами ПК. Також в зв’язку з високою вартістю таких систем, підприємства часто використовують неліцензійні копії данх систем, що є порушенням чинного законодавства[9]. Тому вирішено спроєктувати та розробити систему документообігу для малого та середнього бізнесу з можливістю віддаленого доступу до даних з будь-якої точки світу та яка матиме зручний та інтуїтивно зрозумілий користувацький інтерфейс.

Дана система буде актуальною на українському ринку програмного забезпечення адже систем які мають можливість доступу в режимі онлайн на українському ринку дуже мало та за часту такі системи мають занадто велику вартість та часто розробляються під замовлення підприємства.

1.3 Постановка задачі

З предметної галузі та ринку, можна чітко сформулювати постановку задачі та вимоги до програмного продукту, який буде спроектовано та розроблено.

Метою роботи є аналіз методів проєктування програмних систем для автоматизованої генерації цифрової звітності підприємства малого та середнього бізнесу, дослідження методів проєктування користувацького інтерфейсу, проєктування та розробка UI системи електронного документообігу та програмна реалізація відкритого API для можливості інтеграції системи зі сторонніми сервісами та додатками. Такий підхід спростить подальше масштабування проекту та можливість розробки специфічних систем документообігу на його основі.

Отже сформуємо на основі отриманих даних при аналізі предметної галузі постановку задачі. Для виконання всіх умов мети роботи потрібно:

– дослідити методи проєктування програмних систем для автоматизованої генерації цифрової звітності підприємства, щодо UI/UX;

– дослідити методи інтеграціїї програмних систем із зовнішнім програмним забезпеченням;

- спроєктувати та розробити зручний та інтуїтивно зрозумілий користувацький інтерфейс системи;
- розробити модуль інтеграції системи зі сторонніми сервісами за допомогою відкритого API.

Система буде спроектована та розроблена з використанням міжнародних стандартів проектування веб-додатків W3C (World Wide Web Consortium).

Для проєктування та реалізації системи документообігу будуть використані наступні технології:

– мова розмітки гіпертексту HTML;

– каскадна таблиця стилів CSS;

– клієнтський фремворк Bootstrap 4;

– мова прорамування PHP 7;

– мова програмування JavaScript;

– формат обміну даними JSON;

– веб-сервери Nginx та Apache;

– середовище розробки PHPStorm 2019.3.3.

Основною перевагою обраного для розробки стеку технологій є те, що для їх використання не потрібно обладнання високої потужності, та система може бути використана, як кроссплатформене рішення, тобто його можна використовувати на будь яких пристроях, мають доступ до локальної мережі або інтернету, що надає можливість використання даної системи як в режимі онлайн так і в режимі оффлайн, що робить розроблений продукт універсальним у викроистанні на невеликикх підприємствах[9]. Так як PHP є універсальною та найпопулярнішою мовою розробки кроссплатформених веб-орієнтованих додатків, то було вирішено використовувати її для розробки системи документообігу з можливістю її використання на різних платформах, таких як:

– персональні комп’ютери;

– ноутбуки;

– нетбуки;

– сервери;

– планшети;

– смартфони;

– телевізори.

Для проектування та створення користувацького інтерфейсу було обрано Twitter Bootstrap 4, як один із найпопулярніших та найзручніших фреймворків для створення UI, а також для зручної реалізації та проєктування системи було обрано середовище розробки PHPStorm так як основною мовою розробки серверної частини даної системи є мова PHP 7.2 та клієнтська мова JavaScript.

2 ДОСЛІДЖЕННЯ ТА АНАЛІЗ МЕТОДІВ РОЗРОБКИ КОРИСТУВАЦЬКОГО ІНТЕРФЕЙСУ ТА МЕТОДІВ ІНТЕГРАЦІЇ

2.1 Методи розробки користувацького інтерфейсу

З урахуванням широкого поширення Інтернету складність розробки веб-додатків зросла до рівня прикладних програм і найчастіше найбільш трудомістким процесом є проектування призначеного для користувача інтерфейсу (UI). На сьогодні кількість інтернет-користувачів по всьому світу складає близько 1 млрд. тому бізнес та інші предметні області вимагають продуктів з максимальною продуктивністю, функціоналом і зручністю. Зусилля, витрачені на розробку якісного UI, становлять близько 70% від усього обсягу робіт.

Якість будь-якого інтерфейсу в кінцевому підсумку визначається якістю взаємодії між людиною і системою. Незважаючи на безліч готових методологій і підходів, в області проектування інтерфейсів все ж є кілька актуальних проблем:

– забезпечення відповідності інтерфейсу предметної області. На початку будь-якого проекту аналітикам і проектувальникам доводиться збирати інформацію для створення продукту. Виконана робота підвищує якість продукту, але збір аналітики може бути недостатнім і займає певну кількість часу, що збільшує час і складність розробки. До того ж аналітики і розробники не завжди є експертами в певній предметній області, і дані, зібрані на початку проекту, можуть виявитися застарілими вже в кінці;

– відсутність впорядкованості або формалізації вимог до продукту. Специфікації вимог, які створюються для програмних продуктів, мають свої характеристики, які говорять про правильність сформульованих завдань кінцевих користувачів. В першу чергу вимоги повинні бути адекватними, тобто відповідати очікуванням користувачів. Для однакового сприйняття всіма особами, які беруть участь в розробці веб-додатки, вимоги повинні бути однозначними і ставитися тільки до заданої області знань. Не всі власники і користувачі спочатку знають, чого хочуть, іноді їм потрібно випробувати продукт і тільки потім вони розуміють,

що не так. Це завдання можна вирішити за допомогою створення «живого» прототипу у вигляді анімованого подання роботи вкладок, але знову ж таки, проблема трудомісткості створення таких презентацій залишається не вирішеним;

– гнучкість для користувача інтерфейсів. Наприклад, інтерфейс операційної системи користувач може підлаштувати під себе (додати потрібні програми на робочу область і т.д.), але самі додатки, на жаль, не часто мають дану властивість і за рахунок цього швидко втрачають свою актуальність.

У розробці прикладних програм свою перевагу довів компонентний підхід. Розподіл програми на компоненти забезпечує надійність, швидкодію та адаптивність продукту.

Якщо говорити про компонентні технології, варто визначити, що є компонент. Щодо архітектури ПЗ, компонент - це досить абстрактний елемент структури системи, певним чином виділений серед оточення, вирішуючий деякі підзадачі в рамках загальних завдань системи. В цьому визначенні робиться акцент на виділенні структурних елементів системи в цілому і декомпозицію вирішуваних нею завдань на більш дрібні підзадачі.

Компонент як виділена структурна одиниця має більш суворі вимоги до чіткості визначення інтерфейсу, ніж архітектурний компонент. Абсолютно всі його залежності від оточення повинні бути описані в рамках цього інтерфейсу. Один компонент може також мати кілька інтерфейсів, граючи кілька різних ролей в системі. Таким чином, будь-яка компонентна система забезпечує зниження складності цілому продукту шляхом надання природних бар'єрів, що приховують складність одних систем від інших.

Спроби застосування компонентного підходу до вебу почалися зі згадки про веб-компоненти на github в 2011 році. Спочатку поділ веб-додатків на компоненти вироблялося з боку сервера за рахунок розбиття програми на окремі сторінки, це наочно демонструвало структурованість програми, але ускладнювало організацію навігації.

Сучасні фреймворки (AngularJS, БЕМ і т.д.) в тій чи іншій мірі застосовують

компонентний підхід. Метою даних фреймворків є зменшення складності додатка за рахунок ізоляції пов'язаних груп коду для виконання загального функціоналу в межах контексту, але для деяких завдань опора на HTML, CSS і JavaScript вже не актуальна. Класичні технології в веб-програмуванні, в зокрема JavaScript потребують ретельної конфігурації, що вимагає більш глибокого знання мови і його можливостей, а застосування DOM (Document Object Model) не виключає високий ризик конфлікту між елементами інтерфейсу і певними для них стилями. Можна сказати, що існуючі рішення в області WebComponents дозволяють скоротити час роботи над кодом. Але в даному підході є свої мінуси: на практиці застосування БЕМ (Блок-Елемент-Модифікатор) не гарантує правильності складання проєкту за умови, якщо верстальник нечітко прописав класи, а якщо розмітка підвантажується через скрипт, то всі компоненти і їх зміни доведеться форматувати вручну.

У веб-інженерії компоненти - сукупність стандартів, які дозволяють робити «Віджети» з ізольованими стилями і скриптами у вигляді метатегів, така реалізація дозволяє накопичувати компонентну базу і використовувати її в подальшому.

На сьогоднішній день стандартів для створення веб-компонентів є призначені для користувача елементи (Custom Elements), тіньовий DOM (Shadow DOM), шаблони і HTML-обсяги імпорту (Templates, HTML-imports). Дані стандарти полегшують програмну частину проектування інтерфейсу, але ніяк не враховують якісні характеристики.

Початковий етап розробки будь-якого програмного продукту – це збір і аналіз вимог. Як відомо вимоги діляться на дві категорії: функціональні і якісні. Інтерфейс, відповідний цільовим завданням повинен максимально підходити під вимоги користувача. Так, очевидно, що проектування, налаштування і розробка інтерфейсу вимагає участі користувача.

Методи опису вимог до інформаційних систем досить активно розробляються, більш того, існує окремий напрямок, людино-орієнтованої розробки систем, яка містить в собі принципи, планування і подальший розвиток методів та інструментів для проєктування взаємодії.

Підхід до конфігурації інтерфейсу на основі опису вимог у вигляді

властивостей, якими повинен володіти кінцевий інтерфейс. Кожній властивості із загального набору відповідає ряд теоретичних і технологічних властивостей, які розкривають зміст вимоги і забезпечують його програмну реалізацію. На мою думку, такий підхід допоможе створювати інтерфейси різних типів систем користувачами самостійно і вирішить ряд проблем, серед яких забезпечення адекватності вимог до продукту, варіативності кінцевого рішення і конфігурації інтерфейсу відповідно до динаміки предметної галузі.

На шляху вирішення проблеми однорідності та повноти вимог деякі компанії для зручності їх аналізу створюють гайди, в яких вже виділені вимоги до основних складових інтерфейсу програмного продукту.

Також пропонується висловлювати функціональні вимоги у вигляді Abstract UI, який вдає із себе логічно згруповані завдання щодо предметної області. Використовуючи даний підхід компоненти можна класифікувати відповідно з областю застосування. Також функціональні вимоги можна описати у вигляді use case за допомогою універсальної мови моделювання UML. Для більшості завдань застосування UML не актуальне, тому що викличе ряд труднощів, пов'язаних перш за все з понятійним апаратом мови моделювання.

Також для побудови інтерфейсів використовується моделе-орієнтований підхід. Для аналізу вимог до інтерфейсу використовуються специфікації, які описують завдання користувачів, контент і структуру майбутньої програми. також

Недоліки даного підходу, такі як: відсутність гнучкості інтерфейсу (його все одно потрібно оновлювати вручну) і низька продуктивність методу, тому що багато інструментів розроблені тільки на експериментальному рівні.

Передбачається, що генератор може використовуватися не тільки професійними розробниками, але і звичайними користувачами.

2.1.1 Обробники шаблонів

Технології побудови призначених для користувача інтерфейсів на базі шаблонів, реалізованих на мовах розмітки, почали повсюдно застосовуватися з середини 1990-х. Основні переваги шаблонів - гнучкість і широта можливостей створення динамічних призначених для користувача веб-інтерфейсів, особливо з точки зору розробки структури і планування. Спочатку в таких інструментаріях використовувалися шаблони, в яких планування і структура UI задавалися за допомогою мови розмітки, а прив'язка до даних здійснювалася за допомогою невеликих блоків на мові високого рівня (Java, C #, PHP, Python і т. д.). Останні могли використовуватися в комбінації з розміткою; наприклад, шляхом впровадження тегів розмітки в цикл на Java могли створюватися ітеративні візуальні елементи на зразок таблиць і списків. Необхідність частої зміни синтаксису всередині веб-сторінки ускладнювала розробку і корекцію коду для програмістів, тому близько десяти років тому почався перехід з мов високого рівня на спеціалізовані бібліотеки тегів розмітки і мови виразів, створені для конкретних веб-технологій.

Теги розмітки стали використовувати для реалізації типових функцій веб-додатків, а вираження - для доступу до даних і доступу до більшості функцій, що зберігаються в серверних об'єктах. Типовий представник цієї групи - технологія JavaServer Pages (JSP), бібліотека тегів якої JSP Standard Tag Library підтримує такі завдання, як: маніпуляція з XML-документами, цикли, умови, опитування СУБД (прив'язка до даних) і інтернаціоналізація (форматування даних). Мова виразів JSP - EL, що служить засобом прив'язки до даних, пропонує зручну нотацію для роботи з об'єктами і властивостями додатку.

Існує цілий ряд схожих на JSP інструментаріїв веб-розробки: для планування і задання структури (в них використовуються шаблони), для прив'язки до даних за допомогою мови виразів, а поведінка UI задається за допомогою обробників подій, реалізованих засобами мови ECMAScript і інтерфейсу програмування Document

Object Model. Форматування даних виконується за допомогою спеціалізованих бібліотек тегів, для стилізації зовнішнього вигляду зазвичай застосовується CSS (Cascading Style Sheets). Популярні представники цієї категорії інструментів: ASP, PHP, Struts, WebWork, Struts2, Spring MVC, Spyce і Ruby on Rails.

2.1.2 Об’єктно-орієнтовані та подієві інструменти

Значна частка інструментаріїв для створення UI базується на об'єктно-орієнтованій моделі. Зазвичай ці інструментарії пропонують бібліотеку готових елементів UI, і їх головними перевагами є простота складання багаторазово використовуваних блоків з простих компонентів і інтуїтивно зрозумілий, гнучкий процес програмування поведінки та взаємодії, заснований на обробниках подій. У цих інструментаріях всі завдання розробки UI вирішуються з використанням спеціалізованих об'єктних API. До даної категорії відносяться середовища: Visual Basic, MFC, AWT, Swing, SWT, Delphi, Google Web Toolkit, Cocoa Touch UIKit, Vaadin і ін. Сюди ж можна віднести інструментарій Nokia Qt, що пропонує ряд оригінальних концепцій. У деяких інструментаріях вся складність взаємодії між елементами структури UI реалізується за допомогою обробників подій, а в Qt на додаток до них є «сигнали» і «слоти»: сигнал передається компонентом UI щоразу, коли відбувається певна подія. Слот - це метод, що викликається у відповідь на певний сигнал, який можна декларативно зв'язати з яким завгодно кількістю слотів, і навпаки, один слот може отримувати скільки завгодно сигналів. Елемент, що передає сигнал, «не знає», який слот його отримає. Таким чином, елементи призначеного для користувача інтерфейсу слабо пов'язані сполуками «сигнал-слот». Даний механізм сприяє використанню принципу інкапсуляції і надає можливість декларативно задавати поведінку UI.

2.1.3 Гібриди

Гібридні технології відносно нові в світі розробки UI загального призначення - поряд з шаблонами і мовами виразів в подібних інструментаріях застосовується об'єктний API. Типовий представник - JavaServer Faces: бібліотеки тегів служать для опису структури і планування, а також для форматування даних; мова виразів - для прив'язки елементів і подій до серверних об'єктів і коду додатків; об'єктний API - для відображення елементів, управління їх станом, обробки подій і контролю введення. Інші популярні інструментарії в цій категорії: ASP.NET MVC, Apache Wicket, Apache Tapestry, Apache Click і ZK Framework.

Середовище Adobe Flex концептуально близьке до технологій цієї категорії, так як в ній для структурування і планування використовуються шаблони, а програмування цілком виконується на мові ActionScript. Подібно Qt, середа Flex надає механізм для вирішення завдань, пов'язаних з програмуванням поведінки і прив'язкою до даних.

2.1.4 Декларативні інструментарії

Такі інструменти - новітній напрям в області засобів розробки UI. Для вказівки структури призначеного для користувача інтерфейсу в них використовуються мови на основі XML і JSON (JavaScript Object Notation), а для інших завдань розробки UI застосовується переважно декларативна нотація. На відміну від гібридних підходів, в основному розрахованих на веб-інтерфейси, декларативні застосовуються ще в розробці нативних додатків для мобільних і настільних платформ.

API користувацького інтерфейсу Android - подієво-залежний, об'єктно-орієнтована, але поряд з основним в ОС є допоміжний API, який базується на XML,

який дозволяє декларувати структуру і планування призначеного для користувача інтерфейсу, а також стилізувати його елементи і керувати їх властивостями. Декларативний опис інтерфейсу наочніше показує його структуру і допомагає в налагодженні; дозволяє без перекомпіляції міняти планування; допомагає адаптуватися до різних платформ, розмірами екрану і співвідношенням його сторін. При створенні більш динамічних призначених для користувача інтерфейсів вказувати і змінювати структуру елементів можна і програмно - за допомогою об'єктних API, але прив'язка до даних не підтримується. Існує, правда, Android-Binding - стороннє рішення з відкритим кодом, що дозволяє прив'язувати елементи призначеного для користувача інтерфейсу до моделей даних.

Створювати UI для програм Windows і функціонально багатих інтернет-додатків, заснованих, відповідно, на технологіях Windows Platform Foundation і Microsoft Silverlight, можна з використанням іншого XML-словника - eXtensible Application Markup Language (XAML). Він дозволяє задавати структуру, планування і стиль UI, а крім того, на відміну від мови розмітки Android, в ньому підтримується прив'язка до даних і можливість обробки подій.

В Nokia розробникам рекомендують Qt Quick – крос-платформну інструментарій для настільних, мобільних і вбудованих ОС, що підтримує QML (декларативний скриптова мова на основі синтаксису JSON). Опис призначеного для користувача інтерфейсу має ієрархічну структуру, а поведінка програмується на ECMAScript. Тут, як і в звичайному Qt, підтримується механізм «сигнал-слот». Qt Quick підтримує можливість прив'язки властивостей елементів UI до моделі даних, а також концепцію машини станів, що дозволяє графічно моделювати поведінку інтерфейсу.

Ще один приклад - Enyo, крос-платформний інструментарій для створення UI на ECMAScript, в якому структура інтерфейсу задається декларативно, а поведінка регулюється обробниками подій. Події обробляються трьома способами: на рівні окремих компонентів UI, шляхом передачі від нащадка до батька без прямої прив'язки, а також за рахунок широкомовної трансляції та підписки на такі повідомлення (теж без прямої прив'язки). Завдяки слабкому зв'язку елементів UI

розширюються можливості багаторазового використання і інкапсуляції великих фрагментів інтерфейсу. По суті, основна перевага Enyo - це модель інкапсуляції, завдяки якій UI можна компонувати з багаторазово використовуваних самодостатніх будівельних блоків з заданими інтерфейсами. Дана модель сприяє абстрагування і охоплює всі архітектурні рівні UI. Учасники проекту Enyo працюють над реалізацією підтримки прив'язки до даних.

Eclipse XML Window Toolkit - ще один інструментарій, орієнтований на декларативний опис UI. Початкове завдання його створення полягало в об'єднанні в Eclipse всіх інструментів розробки UI, включаючи SWT, JFace, Eclipse Forms та інших - всі їх елементи так чи інакше мають відповідності в XWT. Структура і планування UI в XWT задаються за допомогою мови на основі XML, а для прив'язки до даних (доступу до Java-об'єктів додатку) використовується мова виразів. Обробка подій програмується на Java, а для стилізації елементів інтерфейсу використовується CSS. Механізм виконання програм XWT реалізований у вигляді Java-аплету і елементу ActiveX, тобто може працювати практично в будь-якому браузері.

У цій категорії існує чимало схожих інструментів: в AmpleSDK, наприклад, в якості мови опису UI використовується XUL, функції ECMAScript застосовуються для програмування динамічної поведінки, CSS - для стилізації. У Dojo Toolkit інтерфейс задається декларативно і передбачений широкий вибір готових елементів, об'єктне сховище для доступу до даних і обробник подій на основі ECMAScript з механізмом публікації-підписки. Інструментарій підтримує інтернаціоналізацію, розвинений API для опитування даних, модуляризацію і множинне спадкування класів.

2.1.4 Інструментарії на основі моделей

Значна частина технологій розробки UI заснована на моделях і предметно-

орієнтованих мовах. В основному це моделі інтерфейсів, але можуть використовуватися і доменні моделі. В обох випадках модель потрібна для генерації користувальницького інтерфейсу заздалегідь або інтерпретується в період виконання. Цей клас технологій піднімає рівень абстракції, пропонує поліпшені систематичні методи проєктування і реалізації користувальницьких інтерфейсів, а також надає інфраструктуру автоматизації відповідних завдань. Однак, на думку деяких дослідників, модельно-орієнтовані технології не дають універсального способу інтеграції користувальницького інтерфейсу з додатком, а також поки немає згоди щодо того, який набір моделей оптимально підходить для опису UI. Не вирішена задача прив'язки даних, і не об'єднані моделі для вирішення інших завдань розробки UI.

Аналізуючи покоління модельно-орієнтованих підходів до розробки UI починаючи з 1990-х, можна прийти до висновку, що сьогодні є загальноприйняте уявлення про рівні абстракції і типи моделей, придатних для розробки сучасного інтерфейсу користувача, проте до цих пір немає єдиної думки (стандартів) щодо інформації (семантики), яку повинні містити різні моделі. Вважати базовими можна моделі задач, діалогів і презентації: презентаційна модель вирішує завдання структурування, планування та стилізації; модель задач відповідає за прив'язку до даних - для кожного завдання вказуються об'єкти UI і логіки, з якими доведеться працювати; діалогова модель охоплює поведінкові аспекти. Приклад моделі задач - Concurrent-TaskTrees (CTT), її можна використовувати спільно з мовою MARIA, який реалізує інші моделі UI. CTT в поєднанні з MARIA є повноцінним модельно-орієнтованим інструментарієм. Досить велике сімейство засобів моделювання UI покладається також на мову UML, моделі «сутність-зв'язок» або подібних. Профілі UML широко застосовуються в побудові користувальницьких інтерфейсів бізнес-додатків. Існують і інші інструментарії, які активно використовуються – наприклад, WebRatio, UMLi, Intellium Virtual Enterprise і SOLoist.

2.1.5 Узагальнені користувацькі інтерфейси

Невелика, але значуща підмножина технологій формування інтерфейсів користувача генерують UI, спираючись на моделі користувача, даних задач або інші види моделі додатків. Інтерфейс генерується виходячи з моделі цілком або в напівавтоматичному режимі. Моделі також можуть інтерпретуватися в період виконання без використання в якості основи моделі для створення інтерфейсу. У будь-якому випадку, завдяки високому рівню автоматизації побудови UI, технології даної категорії заощаджують час розробника і знижують кількість помилок, а інтерфейси, що генеруються мають однакову структуру. Однак узагальнені UI не відрізняються гнучкістю, мають обмежену функціональність і непередбачуваний процес генерації. Проте при наявності прямого зв'язку з доменною моделлю розробка додатків з узагальненими UI цілком реальна. У даній категорії близько десятка прикладів на чолі з широко застосовуваним архітектурним шаблоном Naked Objects. Автоматичну генерацію UI можна з успіхом застосовувати в окремих предметних галузях - наприклад, при дизайні діалогових вікон і призначених для користувача інтерфейсів для віддаленого управління системами. Подальший розвиток цього класу технологій дослідники бачать в удосконаленні методик моделювання і пошуку нових способів комбінування моделей з метою підвищення зручності згенерованих UI.

Сучасна розробка веб-додатків розвивається небувалими темпами, пропонуючи для зручності розробників і поліпшення якості відображення і роботи ресурсу найрізноманітніші технології, методи і способи.

Верстка сайту виступає одним з етапів розробки веб-сайту і є «пожвавленням» дизайн-макету сайту, тобто написання програмного коду сайту відповідно до заздалегідь підготовленому шаблону, продуманій структурі та функціоналу, що дозволяє сторінкам відкриватися і працювати в браузерах на різних пристроях .

Front-end розробка дозволяє дивитися ще глибше, забезпечуючи точне налаштування і оптимізацію всіх процесів, роботу всіх елементів інтерфейсу.

Bootstrap - це повноцінний CSS-фреймворк з розширеними можливостями, що дозволяє швидко створювати сайти на основі дизайн-макетів. Використання даного фреймворка настільки популярно, що забезпечує технології безперервний розвиток, вдосконалення, розширення можливостей.

Bootstrap поєднує в собі html, css і javascript, а тому з його допомогою можна створити будь-яку сітку сайту і елементи інтерфейсу.

Перевагами використання даної технології в сайтобудуванні є:

– висока швидкість створення макетів завдяки величезній бібліотеці готових рішень;

– забезпечення адаптивності сайту і кросс-браузерності;

– простота освоєння і застосування.

Також ефективно і зручно працювати з будь-яким з елементів DOM, подіями, використовувати технологію Ajax, створювати всілякі складні візуальні ефекти і завжди мати під рукою величезну кількість JS-плагінів для створення користувацьких інтерфейсів дозволяє JavaScript-бібліотека jQuery. За допомогою даного фреймворка веб-розробникам вдається надати сайту динамічність.

Тому було прийнято рішення при розробці дизайну системи документообігу використовувати саме ці технгології.

2.2. Методи інтеграції

Проаналізуємо фактори, що впливають на інтеграцію:

– прискорення процесів. Розвиток організації вимагає все частіше і частіше міняти структури даних, бізнес-процеси, не кажучи вже про дизайн та інтерфейс користувача, який просто постійно знаходиться в зміні. Ось, як раз в таких

динамічних областях, де "мінливість" є самою суттю і природою системи, завдання інтеграції посилюється і перетворюється в серйозну проблему;

– розподіленість. Організації стають все більшими, а можуть бути вирішені завдання все більш комплексними, з'являється логічна, організаційна та географічна розпорошеність;

– гетерогенність. У великому проєкті, майже ніколи немає можливості дотримуватися платформ та інструментів від одного виробника, тому доводиться враховувати і підтримувати особливості декількох платформ;

– спадковість. Неможливість повністю відмовитися від legacy систем, морально застарілих технологій, старого апаратного забезпечення, яке, до речі, іноді дає цілком хороші показники по надійності і продуктивності але вже аж ніяк не сприяє інтеграції;

– хаотичність. Не завжди є можливість повністю формалізувати, уточнювати і структурувати дані, і частина моделі залишається "слабо-пов'язаною", яка не піддається або слабо піддається машинній обробці, аналізу, індексації, обрахунку.

– обумовленість. На жаль, інформаційні системи обмежені не тільки технічними рамками, а й звичками людей (яких складно переучувати), особливостями законодавства (яке просто не готове до появи таких систем), безліччю інших чинників, не залежних від розробників;

– інтерактивність. Споживач інформації постійно підвищує свої очікування про швидкість реакції системи, швидкодії і оперативності доставки інформації. Більшість процесів прагнуть до виконання в реальному часі;

– мобільність. Користувач систем став пересуватися швидше, а взаємодія з ним ведеться через канали зв'язку загального користування в транспорті, вдома і на вулиці, в громадських місцях і повсюдно;

– безпека. Поки дані зберігалися на носії всередині приміщення, що охороняється, то особливо ніхто не турбувався про шифрування, але тепер мережеві пакети літають в повітрі і це не можна залишати без уваги.

– високонавантаженість. На складність інтеграції впливають: кількість користувачів в системі, інтенсивність потоку обробки даних, обсяги даних і ресурсомісткість обчислень;

– безперервність циклу роботи. Інтеграція і апгрейд систем майже завжди повинні проводитися без зупинки їх функціонування, плавно, поступово і непомітно для організації та її клієнтів;

– міжсистемна інтеграція. Завдання стикування не обмежені рамками організації, все частіше потрібно інтегруватися з партнерами, клієнтами, постачальниками, підрядниками та навіть державними структурами.

– Тепер проаналізуємо завдання з іншого ракурсу, виділимо параметри, що відповідають за складність інтеграції і розглянемо варіанти мінімізації негативного впливу цих параметрів:

– концептуальна різниця - ґрунтується та тому, що розробники різних систем спочатку прийняли різні рішення, припущення і допущення, які концептуально не стикуються між собою. Вирішується введенням ще одного шару абстракції, який концептуально суперечить обом підходам. При цьому, є два варіанти реалізації: коли система-посередник стає централізованою, а дві і більше інтегровані системи перетворюються в підсистеми та варіант коли ми використовуємо архітектуру брокера (посередника, який не є центром), при цьому системи залишаються незалежними, а брокер забезпечує прошарок між ними;

– технологічна різниця - коли ми маємо несумісні формати обміну даними, протоколи взаємодії та інтерфейси. Вирішується написанням конвертерів, прошарків, брокерів та інших примочок, не цілком гарних, але достатньо надійних.

Розглянемо методи вирішення проблеми інтеграції програмних систем:

2.2.1 Стандартизація

Потрібно і важливо використовувати якомога більше міжнародних, державних і галузевих стандартів, а якщо якихось не вистачає, а вони явно просяться, то потрібно вводити корпоративні стандарти, а часто має сенс і просувати їх у відповідних організаціях для якнайшвидшого поширення і популяризації.

2.2.2 Інтеграція на рівні брокерів

Переваги: універсальність - практично завжди можна створити додатковий програмний модуль, який будуть звертатися в обидві системи, ще й різними способами (наприклад, в одну через базу даних, а в іншу через RPC).

Недоліки: складність, трудомісткість, а отже висока вартість розробки, впровадження та володіння.

2.2.3 Інтеграція на рівні даних

Коли кілька додатків можуть звертатися в одну базу даних або в кілька баз даних, пов'язаних реплікаціями. Переваги: низька вартість інтеграції, а при використанні однієї СУБД це стає дуже привабливим рішенням. Недоліки: якщо база даних не екранована збереженими процедурами і не має необхідних обмежень цілісності (у вигляді вказівки каскадних операцій і тригерів), то різні додатки можуть приводити дані в суперечливий стан. Якщо ж база екранована і цілісність забезпечується, то і в цьому випадку, в паралельно працюючих з однією БД

додатках, будуть дубльовані частини коду, що виконують однакові або схожі операції. Крім того, при змінах структури бази ми будемо окремо переписувати код всіх додатків, з нею працюють.

2.2.4 Інтеграція на рівні сервісів

Це гарна інтеграція, заснована на фіксації інтерфейсів і форматів даних з двох сторін і дозволяє налагодити швидке відпрацювання міжкорпоративної бізнес-логіки. Недоліки: присутня фіксація, а якщо структура або процеси змінюються, то утворюються проблеми і вузько спеціалізовані, виняткові рішення.

2.2.5 Інтеграція на рівні користувача

Це крайній випадок, або не автоматизована інтеграція, коли користувачі переміщують дані між системами через копі-паст, файли, пошту та інші неподобства. Такі методи ми використовувати не будемо, але вони, на жаль, часто застосовуються в той період, доки програмні системи не готові, а розвиток компанії не дозволяє чекати.

3 ОБҐРУНТУВАННЯ ВИБОРУ ПРОГРАМНИХ ЗАСОБІВ ТА ОПИС ПРИЙНЯТИХ ПРОЄКТНИХ РІШЕНЬ

3.1 Опис архітектури та вибір програмних засобів

3.1.1 Системні та функціональні вимоги

Перед тим як почати моделювання та проєктування системи, необхідно визначити обмеження предметної галузі проєкту, що розробляється. Додаток який буде спроєктований та розроблений є веб-орієнтованою системою. Це найперше обмеження даної системи, тому що її неможливо використовувати без веб-сервера. Також сама система буде виконуватися в браузері користувача програми. Це також є обов'язковими обмеженнями, адже для того щоб система могла працювати на користувацькому пристрої, потрібно задовольняти такі умови:

а) Рекомендовані системні вимоги для сервера:

1) операційна система Windows Server 2016 і вище, Debian 8 і вище, CentOS 7 і вище, Red Hat Enterprise Linux 7 і вище, Ubuntu 14.04 і вище;

2) веб-сервер Apache 2.4 і вище або Nginx 1.17 і вище;

3) PHP 7.2 і вище;

4) MySQL 5.8 і вище;

5) процесор Intel Core i3 або новіше;

6) RAM 4 GB і більше;

б) Мінімальні системні вимоги для клієнтської машини:

1) операційна система Windows або Linux;

2) процесор Intel Pentium Gold або новіше (для десктопних пристроїв);

3) RAM 2 GB і більше;

4) дозвіл на використання JavaScript в браузері;

5) браузер Chrome 64.0 і вище, Safari 11.0 і вище, Opera 51.0 і вище, Mozilla Firefox 58.0 і вище, Microsoft Edge 41 і вище;

в) Рекомендовані системні вимоги для клієнтської машини:

1) операційна система Windows або Linux;

2) процесор Intel Core i3 або новіше (для десктопних пристроїв);

3) RAM 4 GB і більше;

4) дозвіл на використання JavaScript в браузері;

5) браузер Chrome 64.0 і вище, Safari 11.0 і вище, Opera 51.0 і вище, Mozilla Firefox 58.0 і вище, Microsoft Edge 41 і вище;

г) Крім програмних і системних обмежень є обмеження зовнішніх факторів, пов’язаних з нею:

1) швидкість інтернет-з’єднання 512 Кбіт/с і вище.

В якості сховища даних системи обрано СУБД MySQL, так як дана система управління базами даних, найбільш підходить для збереження даних при використанні мови програмування PHP. Систему було вирішено створити у вигляді веб-додатку (сайту), який має дві ролі користувачів: користувач та адміністратор. Після вводу свого логіну і паролю користувачам та адміністратору надається доступ до функціоналу системи відповідно до його дозволів та прав в системі.

Для того щоб розуміти, що в системі електронного документообігу повинно бути реалізовано складемо функціональні вимоги до системи.

Основна функціональність даної системи полягає у наступному:

– генерація електронних договорів;

– генерація електронних актів;

– генерація електронних рахунків;

– генерація електронних накладних;

– генерація електронних специфікацій.

При створенні додатку повинно бути взято до уваги:

– зручність використання додатку – інтерфейс має бути зручним та інтуїтивно зрозумілим, не вимагати додаткової підготовки користувачів;

– простота доступу до необхідної інформації;

– зручність генерації документації в системі;

Підводячи підсумок, можна зібрати до купи усі обмеження предметної області програмного продукту. Користувач повинен мати пристрій на базі

операційної системи Windows Server 2016 і вище або Linux, з процесором Intel Core i3 або новіше, та має дозволити використовувати JavaScript в браузері. Також програмне забезпечення потрібно використовувати на WAMP або LAMP сервері щоб продукт працював як треба. У подальшому, із розвитком інформаційних технологій, деякі із цих пунктів можна буде виключити із списку обмежень системи[11].

3.1.2 Прототипування системи

Сформувавши постановку задачі та визначивши системні та функціональні вимоги до програмного продукту перейдемо до його схематичного прототипування (Mockup).

Спочатку створимо графічний прототип сторінки автентифікації користувача в системі (рис.4.1).

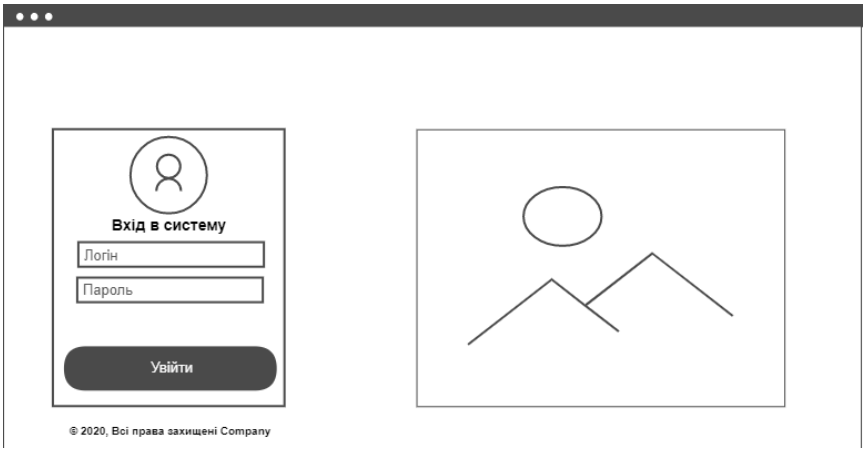

Рисунок 3.1 – Графічний прототип системи. Сторінка автентифікації

На рисунку прототипу ми можемо спостерігати що сторінка автентифікації матиме форму входу до системи за допомогою логіну та паролю, а також на сторінці буде розміщено логотип системи.

Після побудови прототипу сторінки автентифікації створимо прототип основного графічного інтерфейсу нашої системи документообігу. Прототип має відображати основну концепцію системи та місцезнаходження елементів управління системою (рис 4.2).

Рисунок 3.2 – Графічний прототип системи. Основний інтерфейс

На рисунку прототипу основного інтерфейсу системи ми можемо спостерігати місцезнаходження всіх елементів інтерфейсу в системі. Система буде спроєктована в вигляді двохколонкового веб-додатку. В лівій колонці буде знаходитися логотип системи чи компанії, інформація про користувача що пройшов процес автентифікації в системі та меню керування системою. В правій колонці знаходиться основний контент додатку (змінний контент), в залежності від вибраного пункту меню чи сторінки, а також в правому кутку знаходиться меню взаємодії з додатком та кнопка виходу з системи.

3.1.3 UML проєктування

Сьогодні процес створення складних програмних додатків неможливо уявити без поділу на етапи життєвого циклу. Під життєвим циклом програми будемо розуміти сукупність етапів:

– аналіз предметної газузі і створення ТЗ;

– проектування структури програми;

– кодування (набір програмного коду згідно з проектною документацією);

– тестування та налагодження;

– впровадження програми;

– супровід програми;

– утилізація.

Зупинимося детально на процесі проектування. В ході проектування архітектором або досвідченим програмістом створюється проектна документація, що включає текстові описи, діаграми, моделі майбутньої програми. У цій нелегкій справі нам допоможе мова UML.

UML – є графічною мовою для візуалізації, опису параметрів, конструювання та документування різних систем (програм зокрема). Діаграми створюються за допомогою спеціальних CASE засобів. На основі технології UML будується єдина інформаційна модель. Наведені вище CASE засоби здатні генерувати код на різних об'єктно-орієнтованих мовах, а так само мають дуже корисною функцією реверсивного інжинірингу. (Реверсивний інжиніринг дозволяє створити графічну модель з наявного програмного коду і коментарів до нього.)

Розглянемо типи діаграм для візуалізації моделі (це must have, хоча типів набагато більше):

– діаграма варіантів використання (use case diagram);

– діаграма класів (class diagram);

– діаграма станів (statechart diagram);

– діаграма послідовності (sequence diagram);

– діаграма компонентів (component diagram);

– діаграма розгортання (deployment diagram).

Проектована система представляється у вигляді безлічі сутностей або акторів, що взаємодіють з системою за допомогою, так званих прецедентів. При цьому актором (actor) або дійовою особою називається будь-яка сутність, що взаємодіє з системою ззовні. Іншими словами, кожен варіант використання визначає деякий набір дій, який чинять системою при діалозі з актором. При цьому нічого не говориться про те, яким чином буде реалізовано взаємодію акторів з системою.

Діаграма класів служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. Діаграма класів може відбивати, зокрема, різні взаємозв'язки між окремими сутностями предметної області, такими як об'єкти і підсистеми, а також описує їх внутрішню структуру (поля, методи ...) і типи відносин (спадкування, реалізація інтерфейсів ...). На даній діаграмі не вказується інформація про тимчасові аспектах функціонування системи. З цієї точки зору діаграма класів є подальшим розвитком концептуальної моделі проектованої системи. На цьому етапі принципово знання ООП підходу і патернів проектування.

Побудуємо для нашої системи діаграму прецедентів використання (use case), де ми можемо спостерігати, як користувачі з різними ролями в системі можуть її використовувати та обслуговувати.

На рисунку 3.3 зображено всі варіанти використання системи двома акторами.

Як ми можемо спостерігати на діаграмі актор «Користувач» має такі прецеденти:

– управління документами;

– управління клієнтами;

– управління постачальниками;

– верифікація документів;

– управління товарами.

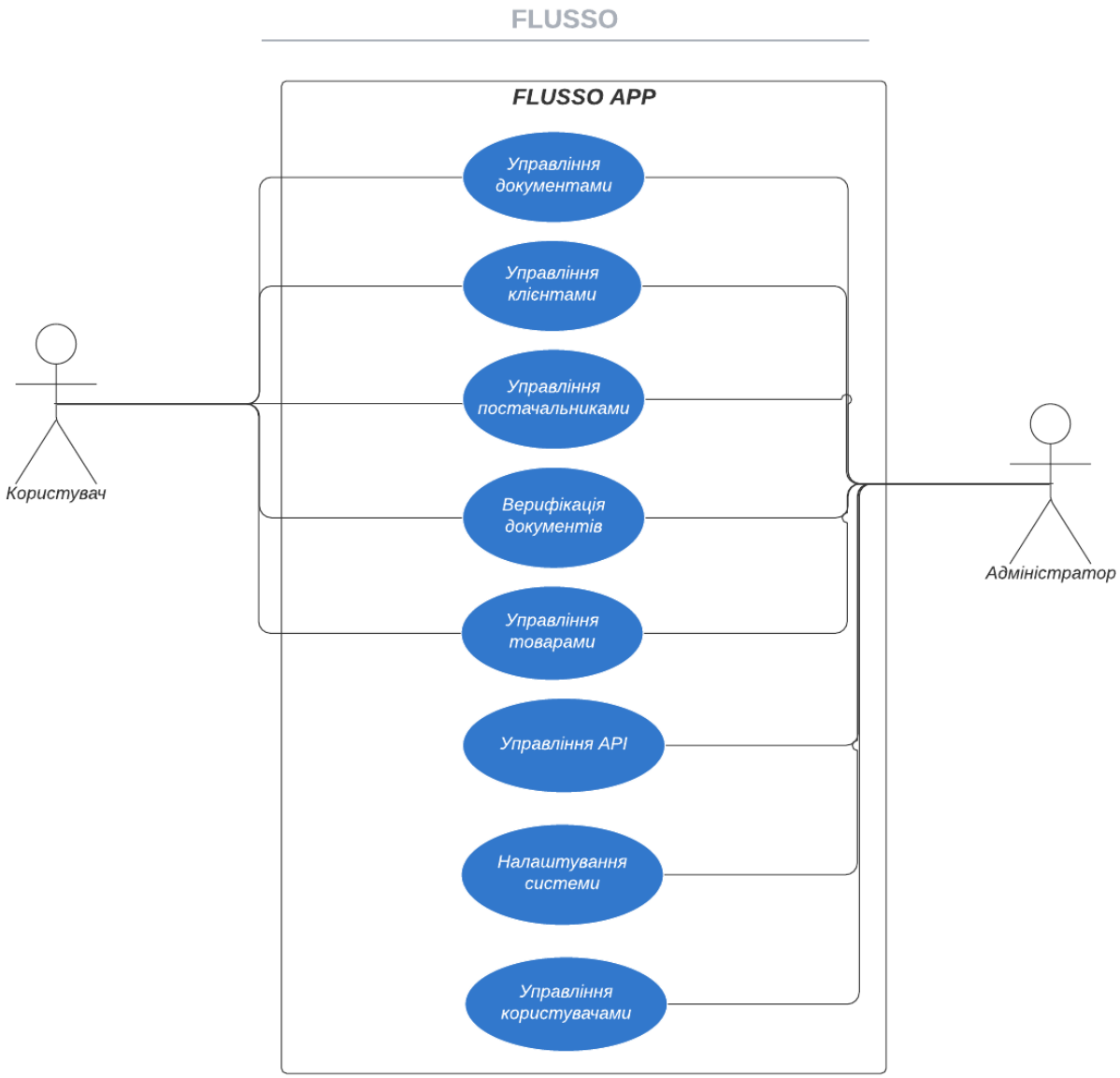

Рисунок 3.3 – UML діаграма прецедентів використання

В той час як актор «Адміністратор», має всі привілеї користувача та ще декілька спеціальних притаманних йому, таких як:

– управління API;

– налаштування системи;

– управління користувачами.

Таким чином ми визначили всі обмеження для користувачів системи за допомогою спроектованої діаграми варіантів використання.

3.1.4 Розробка UI та візуалізація

Для розробки користувацького інтерфейсу було обрано фреймворк Twitter Bootstrap 4, як один з найкращих на найпоширеніших клієнтських фреймоврків для створення якісного UI. Bootstrap - сучасний помічник, розробників інтерфейсів, дизайнерів і вебмайстрів, доступний для використання за відкритою ліцензією. Цей фреймворк дуже динамічний і регулярно оновлюваний, тому не всі його функції можуть коректно підтримуватися старими браузерами.

Bootstrap зараз на піку популярності і можна впевнено сказати, що на його основі можна зробити будь-який веб-інтерфейс.

Фрейморк має такі компоненти проєктрування:

– шрифти;

– кнопки;

– форми;

– мітки;

– навігація;

– сітка;

– JavaScript-розширення.

Зараз найбільш популярною версією фреймворка Bootstrap є четверта.

Основними перевагами Bootstarp є:

– економія часу та витрат - ви економите зусилля бо використовуєте вже готові класи і дизайн;

– адаптивність (mobile first), висока швидкість і оптимізація, стандартизація інтерфейсів - динамічні макети bootstrap якісно відображаються на самих різних пристроях без необхідності внесення змін до розмітку;

– дизайн - єдині шаблони і стильове оформлення елементів макета і всіх сторінок на сайті в цілому. І при цьому Bootstrap кросбраузерний і добре відображається у всіх браузерах Safari, Firefox, IE, EDGE і тих, що на основі Chromium (Опера, Гугл Хром). Регулярне оновлення і доповнення фреймворка найсучаснішими можливостями HTML і CSS;

– простота і відкритість - відкритий вихідний код дозволяє самому брати участь в розробці, модифікувати під свої потреби або просто користуватися хорошим безкоштовним рішенням.

При цьому код HTML, JavaScript і CSS в Bootstrap продуманий і розглянутий під мікроскопом сотнями розробників з усього світу - все для того, щоб пересічні вебмастери та верстальники могли легко і просто налаштувати сітку сайту або вбудувати необхідні елементи в інтерфейс.

Також, в Bootstrap використовує динамічну мову стилів LESS, яка розширює можливості CSS: розробники можуть керувати кольоровою гаммою, створювати вкладені колонки і змінні.

Створення окремої мобільної версії для сайту звичайно є виходом з ситуації адаптації для кишенькових ґаджетів, але тоді нам буде потрібно витратити в два рази більше часу на розробку, проєктування і підтримку системи, а це не завжди економічно вигідно.

Концепція чуйного адаптивного веб-дизайну, втіленого в фреймворці Bootstrap, вирішує саме цю проблему: сайт однаково «відгукується» і відображає інформацію найбільш повним чином незалежно від типу екрану і розміру пристрою. Зміст і колірна гамма не змінюються, змінюється лише форма і спосіб згрупування інформаційних та навігаційних блоків сайту найбільш зручним для користувача способом.

На основі створеного графічного прототипу системи створюємо користувацький інтерфейс системи. На рисунку 3.4 зображено скріншот готового макету сторінки автентифікації користувача.

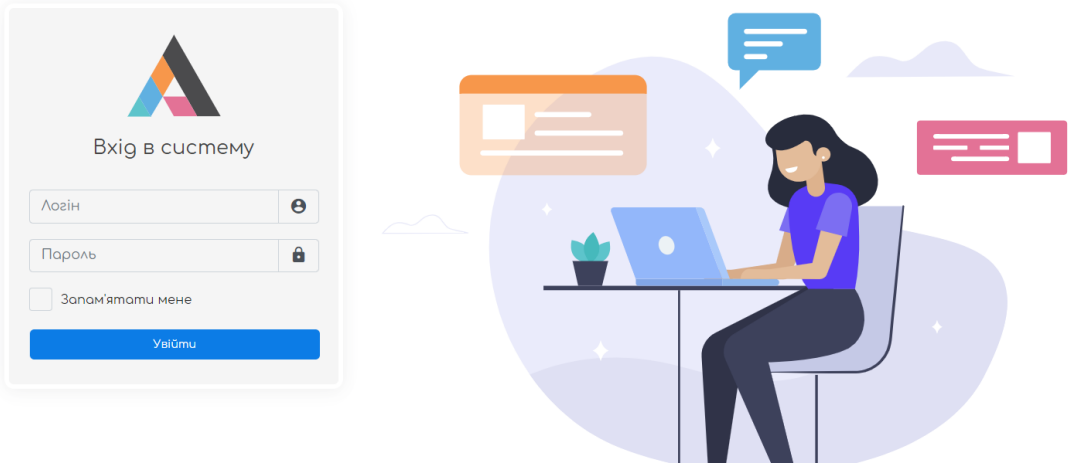

Рисунок 3.4 – Користувацький інтерфейс системи. Сторінка автентифікації

Після створення сторінки автентифікації користувача переходимо до створення основного інтерфейсу системи документообігу (рис.3.5).

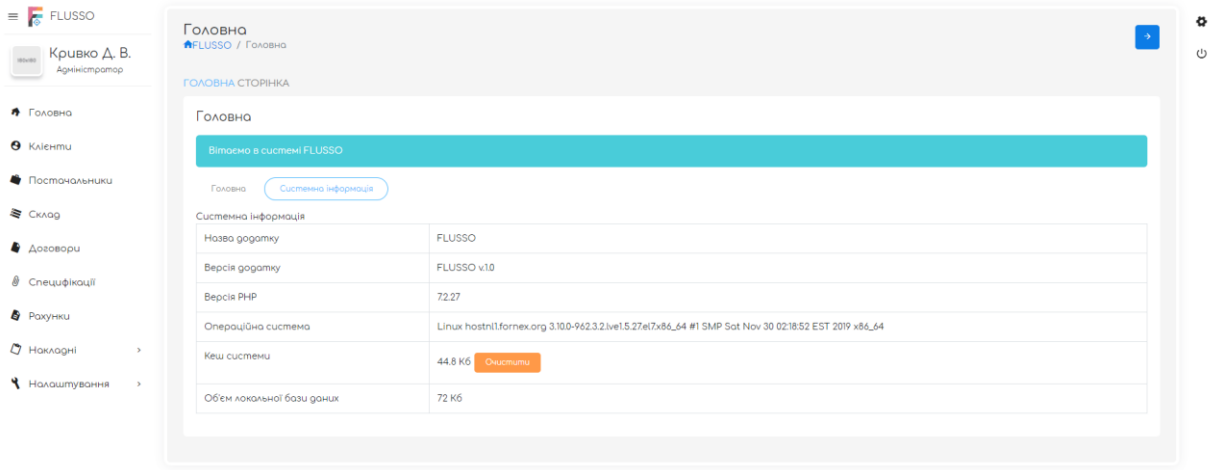

Рисунок 3.5 – Користувацький інтерфейс системи. Основний інтерфейс системи (Головна сторінка)

На рисунку 3.5 зображено концепт UI системи, який має логотип, інформацію про поточного авторизованого користувача, основне меню взаємодії з інформацією, блок відображення контенту та меню взаємодії з аккаунтом користувача. Основне меню має такі пункти та підпункти:

д) головна;

е) клієнти;

ж) постачальники;

з) склад;

и) договори;

к) специфікації;

л) рахунки;

м) накладні;

1) видаткові наладні;

2) прибуткові накладні;

н) налаштування;

1) управління підприємствами;

2) управління категоріями товарів;

3) управління видами ПДВ;

4) управління одиницями виміру;

5) користувачі;

6) ключі приватного API;

7) налаштування системи.

В додатковому меню, що використовується для взаємодії користувача зі своїм аккаунтом знаходяться кнопки дій, що дозволяють перейти налаштувань системи та розміщена кнопка завершення роботи із системою.

Після проєктування та розробки UI головної сторінки та основної концепції розташування меню переходимо до розробки інтерфейсу наступних сторінок взаємодії з системою.

На рисунку 3.6 зображено сторінку взаємодії з інформацією про склад товарів на підприємстві.

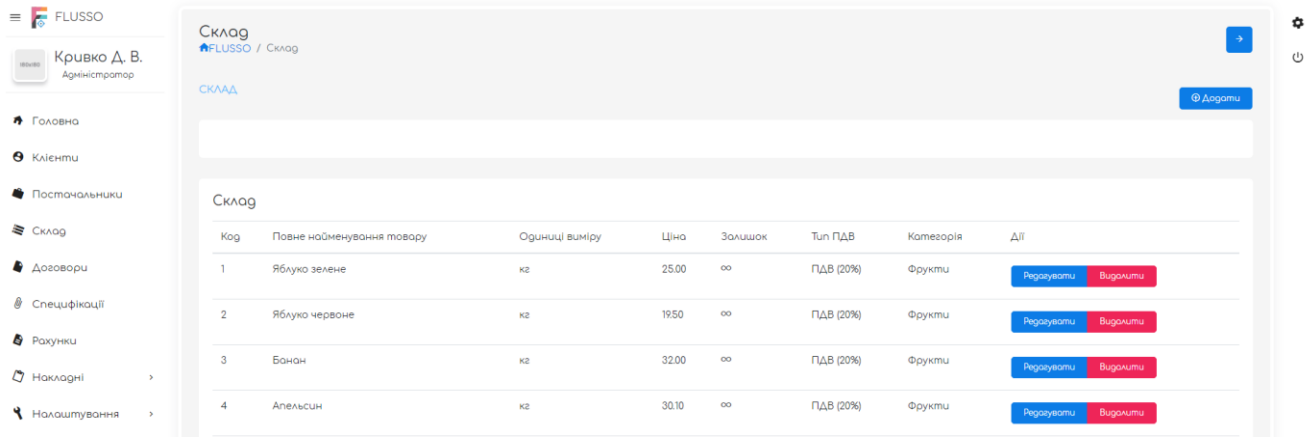

Рисунок 3.6 – Користувацький інтерфейс системи. Склад

В цьому розділі користувач може додати, видалити та відредагувати будь-який товар в системі. І таким чином наповнити базу даних системи товарами для подальшої взаємодії з ними при формуванні цифрової документації.

Після того як було спроєктовано інтерфейс сторінки «Склад» сформуємо пакет сторінок для таких даних як «Клієнти», «Постачальники», «Компанії».

Спочатку сформуємо таблицю зі списком всіх клієнтів. Як ми можемо спостерігати на рисунку 3.7 сторінка складається з таблиці з переліком усіх внесених до бази даних клієнтів, де користувач може переглянути про кожного інформацію, створити, або видалити контрагента.

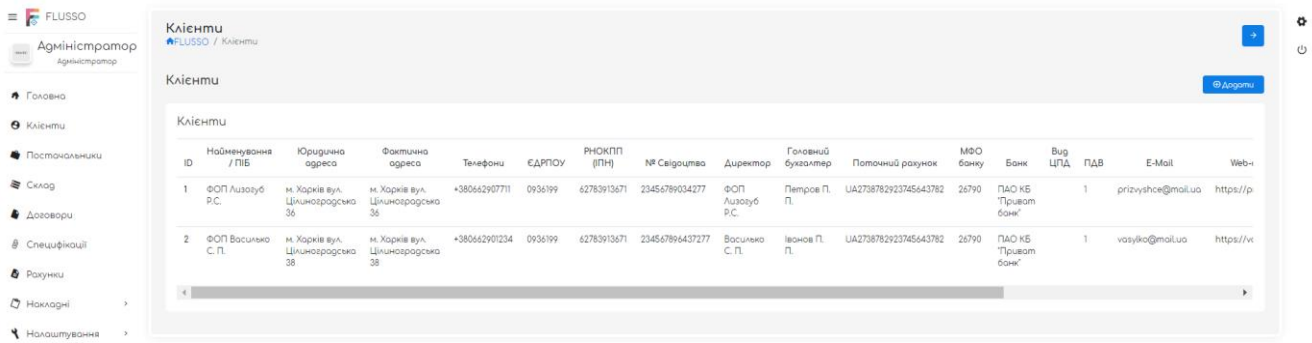

Рисунок 3.7 – Користувацький інтерфейс. Клієнти

Якщо користувач бажає переглянути повну інформацію про клієнта то натиснувши на кнопку переглянути у відповідного клієнта, він потрапить на сторінку з розширеною інформацією (рис.3.8).

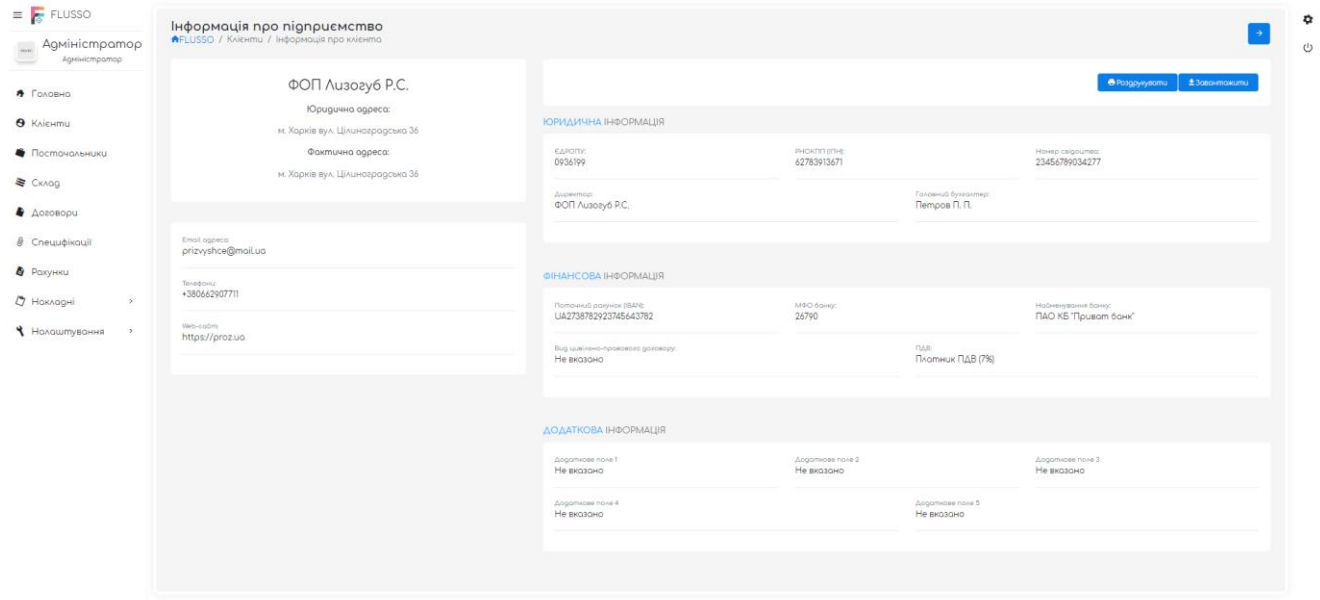

Рисунок 3.8 – Користувацький інтерфейс. Сторінка інформації про клієнта

Сторінки «Постачальники» та «Компанії» виконані в аналогічному стилі, тому по дизайну інтерфейсу не відрізняються.

Для перегляду інформації про сформовані документи, наприклад рахунки-фактури користувач може перейти за відповідним посиланням (рис.3.9).

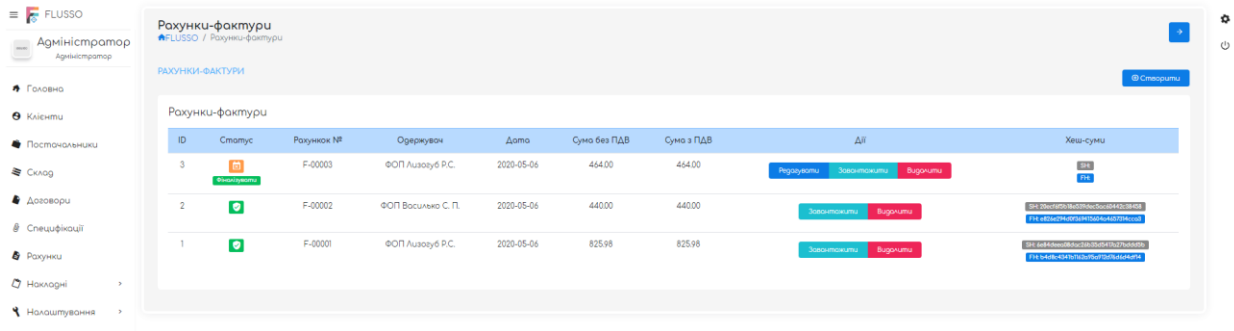

Рисунок 3.9 – Користувацький інтерфейс. Рахунки-фактури

При створенні нового рахунку користувач потрапляє на сторінку генерування документу (рис.3.10), де вводить потрібні йому дані після чого документ буде згенеровано в автоматичному режимі та занесено до бази даних. В залежності від того фіналізований рахунок чи ні буде згенеровано тимчасовий або постійний файл відповідно.

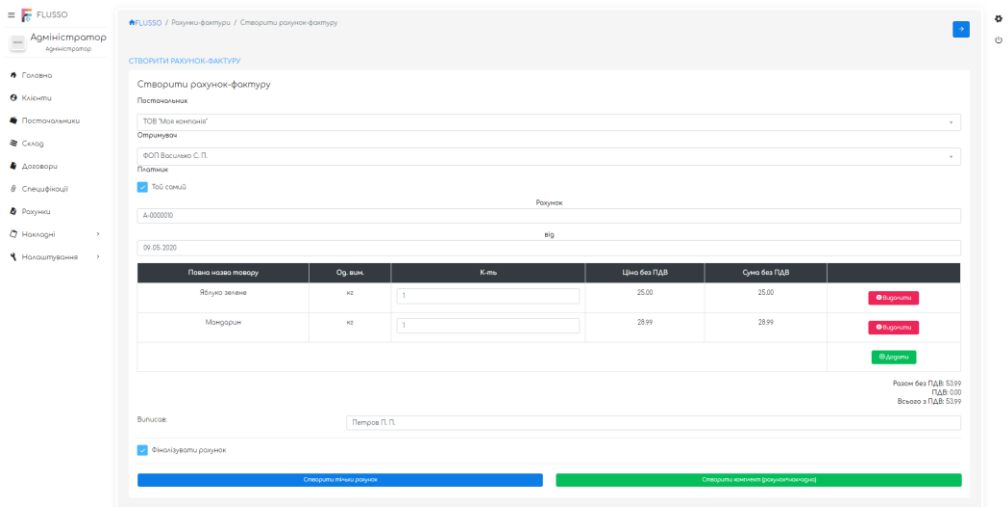

Рисунок 3.10 – Користувацький інтерфейс. Створення рахунку-фактури

Після того як документ було сформовано його можна перевірити на валідність, для цього було спроєктовано інтерфейс системи перевірки документів (рис. 3.11).

Рисунок 3.11 – Користувацький інтерфейс. Перевірка документу на валідність

Після перевірки документу користувач отримає зручну таблицю з інформацією про документ та кнопку на завантаження дублікату в разі необхідності.

Таким чином було спроєктовано концепт системи документообігу, що дозволяє автоматизувати електронний документообіг на підприємстві та спростити роботу з внутрішніми та зовнішніми документами і розвантажити паперовий документообіг в організації де дана система буде впроваджена.

4 РОЗРОБКА ВІДКРИТОГО API СИСТЕМИ

4.1 Інтеграція зі сторонніми сервісами

Для інтеграції системи зі сторонніми сервісами було розроблено програмний інтерфейс REST API який працює на запитах до системи по методу GET. На рисунку 4.1 зображена файлова структура відкритого API системи FLUSSO.

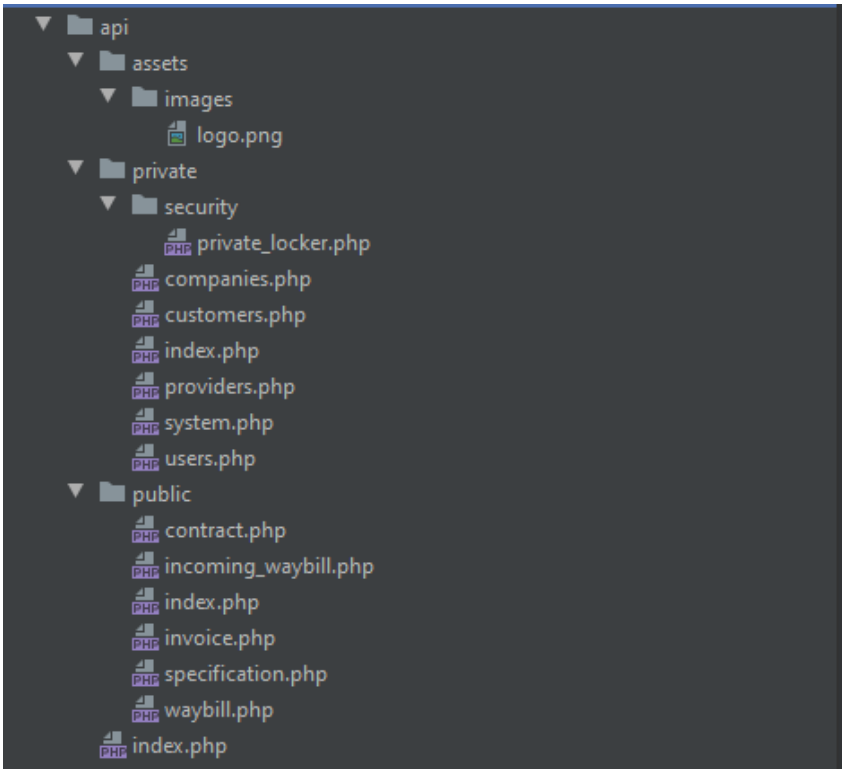

Рисунок 4.1 – Файлова структура API

Файлова структура API включає в себе такі елементи:

а) private – модуль приватного API;
   1) security;
      – private_locker – файл захисту від несанкціонованого доступу до приватного API;
   2) companies – набір методів для роботи з компаніями системи;
   3) customers – набір методів для роботи з клієнтами;
   4) providers – набір методів для роботи з постачальниками;

5) system – набір методів для роботи із системними функціями;
6) users – набір методів для роботи з користувачами системи;

б) public – модуль публічного API;
1) contract – набір методів для роботи з договорами;
2) incoming_waybill – набір методів для роботи з прибутковими накладними;
3) invoice – набір методів для роботи з рахунками-фактурами;
4) specification – набір методів для роботи зі специфікаціями до договорів;
5) waybill – набір методів для роботи з видатковими накладними;

Метод обробки запиту до API виглядає наступним чином:

```php
if (isset($_GET['allComplex'])) {
    $obj_invoice = new Invoices();
    $invoice = $obj_invoice->getComplexAll();
    $json = "[";
    for ($i = 0; $i < count($invoice); $i++) {
        $dataTable = $obj_invoice->getDataTableById($invoice[$i]['id']);
        $json .= "{";
        $json .= "\"invoice\": {";
        $json .= "\"provider\":\"" . $invoice[$i]['company_id'] . "\",";
        $json .= "\"customer\":\"" . $invoice[$i]['receiver_customer_id'] . "\",";
        $json .= "\"payer\":\"" . $invoice[$i]['payer_customer_id'] . "\",";
        $json .= "\"same\":\"" . $invoice[$i]['same'] . "\",";
        $json .= "\"invoice_number\":\"" . $invoice[$i]['invoce_number'] . "\",";
        $json .= "\"date\":\"" . $invoice[$i]['date'] . "\",";
        $json .= "\"dataTable\":[";
        for ($j = 0, $count = count($dataTable); $j < $count; $j++) {
            $json .= "{";
            $json .= "\"name\":\"" . $dataTable[$j]['product_name'] . "\",";
            $json .= "\"units\":\"" . $dataTable[$j]['units'] . "\",";
            $json .= "\"quantity\":\"" . $dataTable[$j]['quantity'] . "\",";
            $json .= "\"price\":\"" . $dataTable[$j]['price_without_vat'] . "\",";
            $json .= "\"sum\":\"" . $dataTable[$j]['sum'] . "\"";
            if ($j == $count - 1) {
                $json .= "}";
            } else {
                $json .= "},";
            }
        }
        $json .= "],";
        $json .= "\"total_without_vat\":\"" . $invoice[$i]['total_without_vat'] . "\",";
        $json .= "\"total_vat\":\"" . $invoice[$i]['vat'] . "\",";
        $json .= "\"total_with_vat\":\"" . $invoice[$i]['total'] . "\",";
        $json .= "\"written_by\":\"" . $invoice[$i]['written_by'] . "\",";
        $json .= "\"finalize\":\"" . $invoice[$i]['finalize'] . "\",";
        $json .= "\"security_hash\":\"" . $invoice[$i]['security_hash'] . "\",";
        $json .= "\"file_hash\":\"" . $invoice[$i]['file_hash'] . "\",";
        $json .= "\"finalized_path\":\"" . $invoice[$i]['finalized_path'] . "\"}";
        if ($i == count($invoice) - 1) {
            $json .= "}";
        } else {
            $json .= "},";
        }
    }
    $json .= "]";
    echo $json;
}
```

Рисунок 4.2 – Фрагмент коду генерації JSON відповіді

При зверненні до API за допомогою GET-запиту користувач отримає відповідь в форматі JSON (рис.4.3) після чого можна провести інтеграцію інформації із системи FLUSSO до стороннього сервісу, або також можна використовувати відповідь API для розширення функціоналу системи.

GET запит до API виконується за захищеним протоколом HTTPS та виглядає наступним чином: https://flusso.app/api/public/invoice.php?allComplex


```
{
  "invoice": {
    "provider": "1",
    "customer": "",
    "payer": "1",
    "same": "1",
    "invoice_number": "F-00002",
    "date": "2020-05-06",
    "dataTable": [
      {
        "name": "Авокадо",
        "units": "шт.",
        "quantity": "2.00",
        "price": "40.00",
        "sum": "80.00"
      },
      {
        "name": "Помідор жовтий",
        "units": "кг",
        "quantity": "30.00",
        "price": "12.00",
        "sum": "360.00"
      }
    ],
    "total_without_vat": "440.00",
    "total_vat": "0.00",
    "total_with_vat": "440.00",
    "written_by": "Кривко Д. В.",
    "finalize": "1",
    "security_hash": "20ecf6f5b18e539dec5ac60442c38458",
    "file_hash": "e826e294d0f369415604a4657314cca3",
    "finalized_path": "/app/resources/finalized/invoices/Рахунок F-00002 FINALIZED.docx"
  }
}
```


Рисунок 4.3 – Відповідь API в форматі JSON

Якщо користувач звернувся до неіснуючого методу API або не вибере метод чи функцію він отримає відповідне повідомлення (рис.4.4).

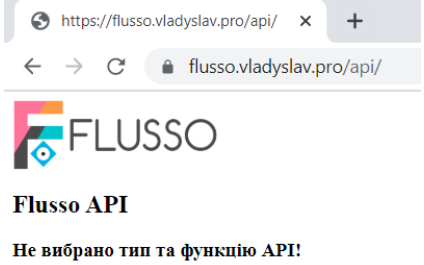

Рисунок 4.4 – Сповіщення про не вибрану функцію API

При зверненні до приватного API, якщо у користувача відсутній ключ доступу він отримає повідомлення про те що він не може скористатися приватним API (рис. 4.5).

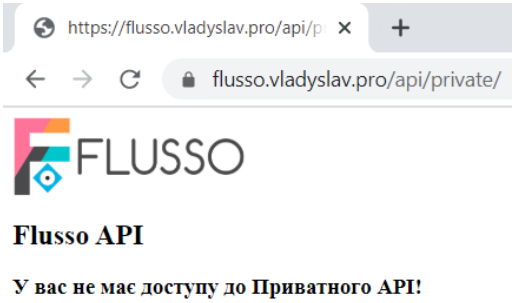

Рисунок 4.5 – Сповіщення про відсутність доступу до приватного API

Для отримання доступу до приватного API адміністратор повинен згенерувати ключ в системі перейшовши за відповідним посиланням (рис.4.6).

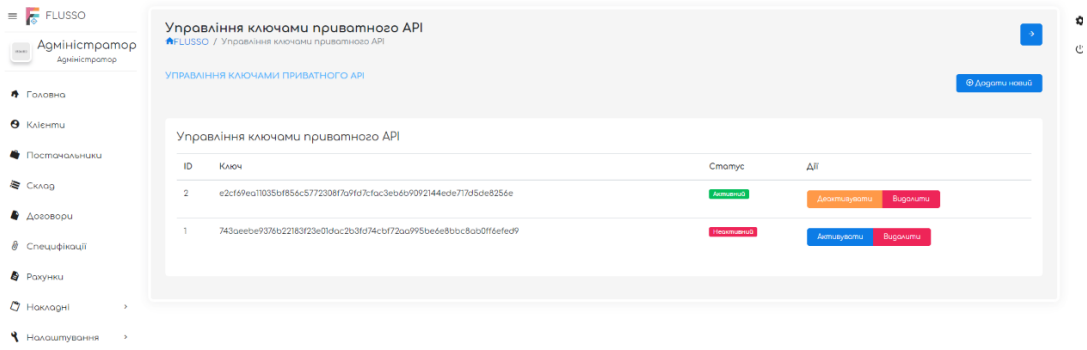

Рисунок 4.6 – Сторінка генерації ключів для приватного API

Після генерації ключа доступу користувач зможе використовувати приватне API за призначенням використавши ключ доступу в запиті:

https://flusso.app/api/private/providers.php?private_key=e2cf69ea11035bf856c5772308f7a9fd7cfac3eb6b9092144ede717d5de8256e&all

ВИСНОВКИ

В результаті атестаційної роботи магістра було досліджено методи проєктування систем електронного документообігу, що стосуються користувацького інтерфейсу, досліджено методи інтеграції програмних систем зі сторонніми додатками, проведено аналіз існуючих систем, спроєктовано та розроблено зручний та інтуїтивно зрозумілий користувацький інтерфейс для онлайн-системи документообігу підприємства, розроблено відкрите API для можливості інтеграції системи зі сторонніми сервісами та спрощення розширення системи при подальших розробках на її основі.

Для проєктуваннння системи використано фреймворк Bootstrap 4 та технологію дизайну Aero UI, що дозволило зробити адаптивним інтерфейс користувача для великої кількості платформ використання.

Отже, після завершення проєктування та розробки системи можна зазначити притаманні їй переваги:

– кросплатформність інтерфейсу, можливість використання системи на ноутбуках, планшетах, смартфонах, персональних комп’ютерах, телевізорах тощо;

– зручність та інтуїтивно зрозумілий інтерфейс, який дозволяє непідготованому користувачу шивдко здобути навички роботи з системою документообігу;

– можливість використання системи для різних задач підприємства;

– можливість використання системи в режимі віддаленого доступу;

– можливість використання системи як псевдопрогресивного веб-додатку або локальної системи;

– можливість інтеграції системи документообігу зі сторонніми сервісами та додатками за допомогою відкритого API.

До негативних моментів проєкту можна віднести такі особливості:

– незручність використання системи на дисплеях з роздільною здатністю менше ніж 640x480;

– при використанні системи в режимі онлайн потрібна наявність інтернету та швидкість каналу не менше ніж 512 КБіт/с, тобто при використанні системи з мобільним каналом зв'язку потрібна наявність 3G, або 4G покриття.

Наявність таких негативних чинників не є проблемою у повноцінній роботі з системою та подальшій розробці на її основі, так як сучасні пристрої мають роздільну здатність 1280x720 і вище та сучасні мобільні та стаціонарні канали зв’язку дозволяють використовувати систему в повному обсязі без особливих перешкод.

Унікальність розробленої системи є в тому, що вона на відміну від комерційних систем електронного документообігу, які є в основному пропрієтарними, має зручний користувацький інтерфейс та має можливість інтеграції зі сторонніми ресурсами за допомогою відкритого API та має відкритий похідний код, що дозволяє модифікувати систему під специфічні задачі різних сфер бізнесу.

В подальшому в розроблене програмне забезпечення можливе впровадження системи більш широкої інтеграції, наприклад інтеграція з державними ресурсами для автоматизації подачі звітності до податкової та фіскальної служби Україні.

Отже, в ході атестаційної роботи магістра було виконано всі вимоги відповідно до постановки задачі, функціоналу програмного продукту, а також спроєктовано та розроблено програмний продукт таким чином, що він має можливість досить легкого розширення для специфічних задач.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Автоматизированные информационные технологии в экономике: підручник. Москва: ИНФРА-М, 1998. – 258 с.

2. Банк В. Р., Зверев В. С. Информационные системы в экономике, Москва: ЭКОНОМИСТЪ, 2006. – 477с.

3. Белоус Н. В., Стужко С.М., Слипченко Н.И., Борисенко В.П. Интегрированная информационная система учебного научно-производственного центра аутсорсинга // 15-я международная научно-практическая конференция "Современные информационные и электронные технологии" ("СИЭТ-2014") 26-30 мая 2014, Украина, Одесса / Сборник научных трудов конференции, с. 96-97.

4. Кузнецова Т.В. Делопроизводство (Документационное обеспечение управления). – Москва: «Интел-Синтез», 2000. – 818 с.

5. Остелло М. Влияние новой информационной технологии на управленческий процесс // Проблемы теории и практики управления. – 2002. – №6. - с.49.

6. Система електронного документообігу FossDoc. Офіційний сайт / Сайт сервісу FossDoc. URL: https://fossdoc.com/ (дата звернення: 05.03.2020).

7. elDoc – система електронного документообігу / Сайт сервісу elDoc. URL: https://dms-solutions.co/uk/products/eldoc-document-management-system/ (дата звернення: 08.03.2020).

8. Система электронного документооборота (СЭД) – Optima-WorkFlow / Сайт сервісу Optima-WorkFlow. URL : http://optima-workflow.ru/ (дата звернення: 09.03.2020).

9. Печникова Т.В., Печникова А.В. Практика работы с документами в организации: Навч. посібник. – Москва: ЭМОС, 1999. – 208с.

10. Маккоу А. Веб-приложения на JavaScript: навч. посіб. – Санкт-Петербург: Питер, 2012. – 288 с.

11. Веллинг Л., Томсон Л. Разработка веб-приложений на PHP и MySQL., 4-е изд.: підручник – Москва: Вильямс, 2013. – 848 с.

12. Емельянова Н.З., Патртыка Т.Л., Попов И.И. Проектирование информационных систем: навч. посібник. – Москва: ФОРУМ, 2009. – 432 с.

13. Степанова Е.Е., Хмелевская Н.В. Информационное обеспечение управленческой деятельности.: навч. посібник – Москва: ФОРУМ, 2010 – 192 с.