МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«ЗАСТОСУВАННЯ PYTHON ДЛЯ РЕАЛІЗАЦІЇ БІЗНЕС-АНАЛІЗУ В КОРПОРАТИВНІЙ ІНФОРМАЦІЙНІЙ СИСТЕМІ»**

Виконав: студент 2 курсу, групи 8.1211-з
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми інженерія програмного забезпечення
(назва освітньої програми)
С.О. Овчаренко
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, доцент, к.ф.-м.н. Горбенко В.І.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри фундаментальної та прикладної математики, доцент, к.ф.-м.н. Панасенко Є.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення

(шифр і назва)

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.

(підпис)

« ____ » _____ 2022 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Овчаренко Сергію Олександровичу

(прізвище, ім'я та по-батькові)

1. Тема роботи (проєкту) Застосування Python для реалізації бізнес-аналізу в корпоративній інформаційній системі

керівник роботи (проєкту) Горбенко Віталій Іванович, к.ф.-м.н., доцент

(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 10 » травня 2022 року № 514-с

2. Строк подання студентом роботи

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Постановка задачі.
2. Основні теоретичні відомості.
3. Переваги користування інформаційною системою.

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

Презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	11.05.2022	
2.	Збір вихідних даних.	28.05.2022	
3.	Обробка методичних та теоретичних	06.09.2022	
	джерел.		
4.	Розробка першого та другого розділу.	17.10.2022	
5.	Розробка третього розділу.	21.11.2022	
6.	Оформлення та нормоконтроль	02.12.2022	
	кваліфікаційної роботи.		
7.	Підготовка доповіді та презентації.	09.12.2022	
8.	Захист кваліфікаційної роботи.	16.12.2022	

Студент _____ С.О. Овчаренко
(підпис) (ініціали та прізвище)

Керівник роботи _____ В.І. Горбенко
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Застосування Python для реалізації бізнес-аналізу в корпоративній інформаційній системі»: 60 с., 22 рис., 27 джерел, 1 додаток.

КОРПОТАТИВНА ІНФОРМАЦІЙНА СИСТЕМА (КІС), МОДУЛЬ, ERP, HTML, JS, PYTHON.

Об’єкт дослідження – застосування Python для реалізації бізнес-аналізу в корпоративній інформаційній системі.

Мета роботи – розробка програмного модуля, який дозволяє автоматизувати процес управління ресурсами підприємства.

Методи дослідження – методи програмної автоматизації систем планування ресурсів підприємств.

Результати – розроблений програмний модуль, що дозволяє спеціалістам швидко та легко отримати дані та проаналізувати їх.

SUMMARY

Master’s qualifying paper «Application of the Python for the Implementation of Business Analysis in Corporate Information System»: 60 pages, 22 figures, 27 references, 1 supplement.

CORPORATE INFORMATION SYSTEM (CIS), MODULE, ERP, HTML, JS, PYTHON.

Object of the study – is the application of Python for the implementation of business analysis in the corporate information system.

Aim of the study: the purpose of the work is to develop a software module that allows automating the process of managing enterprise resources.

Method of research – methods of software automation of enterprise resource planning systems.

Results is a developed software module that allows specialists to quickly and easily obtain data and analyze it.

In the third section, the stress-strain state of a folding cylindrical shell, which is a model of the walls of a cylindrical reservoir, is investigated.

ЗМІСТ

ВСТУП

Повсякденна робота бізнес-аналітика є мистецтвом, але складається також із шаблонних, повторюваних маніпуляцій збору, огляду та візуалізації даних, які можна автоматизувати за допомогою мови програмування Python.

Python – це скриптова мова програмування. На ньому зручно писати скрипти або послідовності інструкцій для реалізації однакових, схожих один на одного операцій. Мова має у своєму складі кошти для роботи з різними базами даних та інструментарій для інтеграції зі спеціалізованими програмами, наприклад, офісними, для наукових розрахунків, адміністрування IT-систем або бізнес-аналізу.

Мова програмування Python дозволяє швидко створювати свої унікальні інструменти для роботи з бізнес-даними та не обмежуватися логікою та функціоналом систем Business Intelligence. Такий багатофункціональний інструмент буде дуже придатний для роботи бізнес-аналітиків. Мета написання цієї кваліфікаційної роботи – дослідження призначення, функцій, структури та еталонів корпоративних інформаційних систем, розробка та застосування сучасного інструментарію для моделювання бізнес-процесів та продуктового менеджменту підприємства, розробка інформаційної системи мінімізації ризиків управління ресурсами підприємства.

1 КОРПОРАТИВНА ІНФОРМАЦІЙНА СИСТЕМА

Сучасні підприємства є важкими динамічними системами. Вони прогресують у часі і включають величезну кількість елементів, що реалізують різні виробничі та управлінські функції. Такі економічні об’єкти мають багаторівневу схему, а також великі зовнішні та внутрішні інформаційні зв’язки. В даний час починають розуміти всю необхідність комплексного підходу до автоматизації інформаційних процесів на підприємствах і організацій. На своїй навичці багато розробників зрозуміли, що результативність автоматизації в першу чергу залежить від того, наскільки широко вона охоплює комплекси розрахунків, що проводяться в управлінні. Відтак останнім часом стала настільки знаменитою ідея побудови корпоративних інформаційних систем (КІС) щодо не лише величезних, регіонально-розподілених інформаційних систем, а й усіляких підприємств, самостійно від їхнього масштабу та форми власності. Організація, маючи сьогодні одну мережу з локальним сервером і десятком комп’ютерів, завтра може розширитися і являти собою саморегулюючу систему, здатну еластично і швидко перебудовувати тези свого функціонування, маючи в своєму активі інтеграцію великої кількості програмних продуктів [6, 10].

Корпоративні ІС призначені для автоматизації всіх функцій управління фірмою чи корпорацією, що має регіональну роз’єднаність між підрозділами, філіями, відділеннями, офісами. Корпоративна ІС – це інформаційна система, що підтримує оперативний та управлінський контроль на підприємстві та представляє інформацію для оперативного прийняття управлінських рішень. Корпоративна ІС (КІС) охоплює всі бізнес-функції та всі управлінські процеси організації. В умовах величезних підприємств та корпорацій вона може бути більш ефективною, тому що забезпечує взаємодія масових і якісно організованих процесів швидкодіючими засобами нових інформаційних і телекомунікаційних спец спецтехнологій високого науково-технічного рівня.

Основними особливостями корпоративних ІС є:

– комплексність охоплення функцій управління;

– підвищена впорядкованість ділових процесів;

– масовість операцій;

– результативність застосування комп'ютерно-телекомунікаційного обладнання та програмного забезпечення;

– можливість локальної установки та впровадження окремих частин системи;

– адаптивність функціональної та інструментальної структури системи до особливостей керованого об'єкта;

– можливість поліпшення системи після її впровадження.

Інформаційна система управління для індустріального підприємства повинна об'єднувати в собі три рівні управління процесами, що протікають на підприємстві:

– управління бізнес-процесами;

– управління проектно-конструкторськими розробками;

– управління технологічною дією виробництва.

Організація, щоб домогтися оперативності та еластичності, необхідної для виживання за нових умов конкурентної боротьби, реорганізується розподілом на дрібні тактично самостійні бізнес-одиниці. Паралельно відбувається делегування управлінських повноважень згори донизу. Ураховуючи, що інформаційні технології відіграватимуть дедалі важливішу роль у комерційному успіху, більшість бізнес-одиниць, очолюваних висококваліфікованими керівниками, обов'язково творять множину стратегій інформаційних технологій на базі програмних і технологічних продуктів кількох постачальників. Підтримка продуктів різних постачальників слугуватиме стандартом, критично важливим для об'єднання компанії в єдине ціле.

Система має працювати як взаємопов'язаний комплекс певних елементів структури.

ІС – це система, яка з підтримкою певних інструментів збирає дані, передає та обробляє їх, та надає цю оброблену інформацію працівникам будь-якого рівня з метою реалізації функцій управління та підтримки циклу на підприємстві. Схему можна подивитися на рисунку 1.1 – це сукупність окремих елементів системи, які працюючи спільно називаються підсистемами.

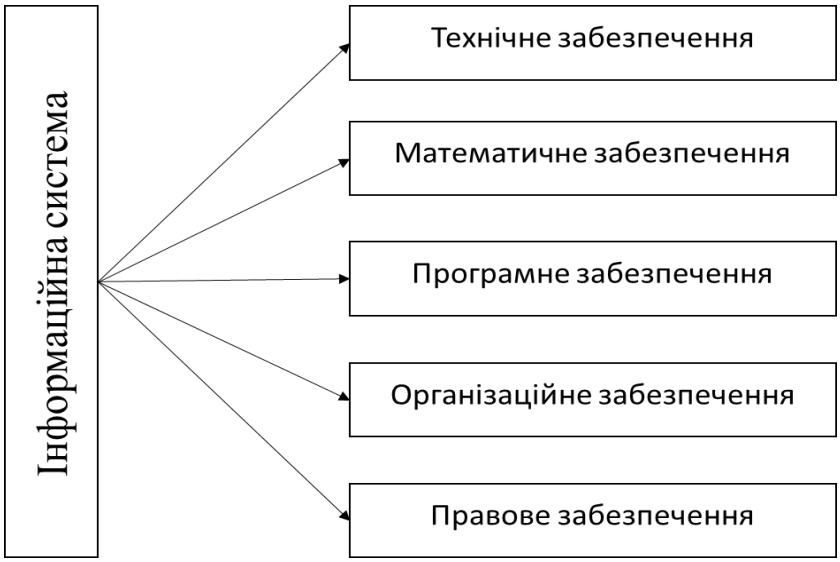


Рисунок 1.1 – Структура інформаційної системи підприємства

1.1 Задачі бізнес-аналізу в корпоративній інформаційній системі

Цілісність інформаційної системи управління підприємством у тому, що дані, отримані чи запроваджені будь-якому рівні системи, повинні бути доступні всім її компонентам (метод одноразового введення).

Сучасна автоматизована система управління бізнес-процесами індустріального підприємства повинна забезпечувати:

– максимально допустимий комплекс функцій для управління всіма бізнес-процесами підприємства: управління маркетингом та продажами, управління постачанням, управління фінансами, життєвий цикл виробу від конструкторських розробок до масового виробництва та сервісного обслуговування;

– тактику виробництва, орієнтовану на покупця, вільно від цього, розробляє підприємство продукцію на замовлення, відправляє їх у склад, веде одиничне, дрібносерійне чи багатосерійне виробництво;

– управління виробничим процесом та постійне контролювання його параметрів на відхилення від можливих значень, починаючи зі стадії планування замовлення на реалізацію до відвантаження готової продукції покупцю;

– реалізацію методології управління витратами та центрами витрат, яка потребує планування собівартості виробів, заяви планових нормативів та контроль відхилень фактичних витрат від їх нормативів для своєчасного вжиття заходів, контроль витрат повинен проводитись за місцями їх появи та дозволяти управлінському персоналу вести огляд;

– система має надати цілісність даних фінансового та управлінського обліку, а саме за допомогою виробничого плану та нормативної собівартості система розраховує кошторис витрат на виробництво;

– дані, введені в систему повинні бути доступні відразу після реєстрації господарської операції всім, хто відчуває в них потребу: від обліку в цеху до керуючого підприємством.

Це дозволить здійснювати контроль над виробництвом лише на рівні виробничих кошторисів за умов функціонування підприємства [8].

Структура єдиної інформаційної системи управління корпорацією повинна бути наступною (див. рис. 1.2).

Світовий досвід використання ІТ показує, що структура такої виняткової інформаційної системи управління підприємством має бути такою [7, 10].

Першим і основним елементом інформаційної системи управління підприємством є система управління бізнес процесами підприємства – система класу ERP (Enterprise Resources Planning – Цінування джерел підприємства). Основним призначенням ERP систем є автоматизація процесів планування, обліку та управління за основними напрямами діяльності підприємства і, отже,

ERP-системи – системи планування ресурсів підприємства), в загальних рисах можна розглядати як інтегровану спільність наступних основних підсистем:

- управління фінансами;
- управління фізичними потоками;
- керування виробництвом;
- управління планами;
- управління сервісним обслуговуванням;
- управління якістю;
- управління персоналом.

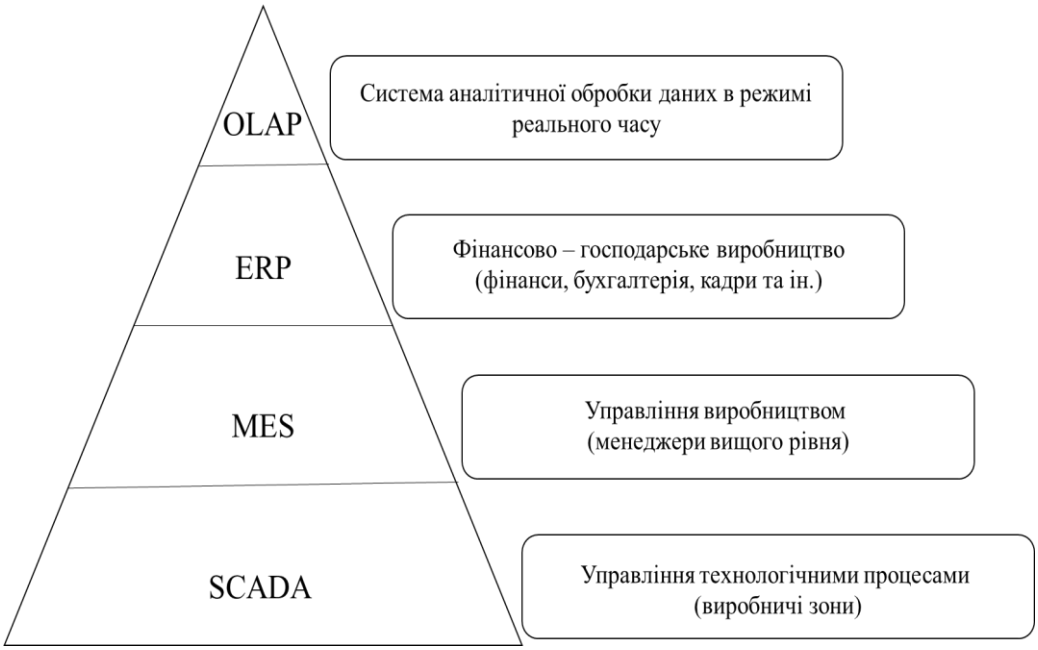


Рисунок 1.2 – Структура єдиної інформаційної системи управління корпорацією

Корпоративна інформаційна система (КІС) – це інтегрована система управління підприємствам чи корпорацією, що зумовлює застосування всіх даних підприємства та його поглиблений огляд. Також вона працює з документообігом та електронним діловодством. Тобто, вона веде повний цикл знаходження товару на підприємстві, щодо його надходження, знаходження, пересування та передачі. Вона реалізує та автоматизує ідеї та способи за

допомогою інструментів та технічних засобів автоматизації. До неї також можна включити: технічне забезпечення підприємства, математичне, програмне, інформаційне, організаційне та правове забезпечення як зображено на доданому рисунку (див. рис. 1.1).

Головним завданням такої корпоративної системи є правильне управління всіма джерелами організації, що допомагає підвищити прибуток для власника організації та максимально задовольнити фінансову сторону, тобто зменшити кількість зайвих рухів працівників та збільшити кількість продажів на підприємстві [6, 10].

Систему управління можна лише тоді назвати корпоративною інформаційною системою, коли вона включає наступні вимоги:

– довговічність та охорона даних;

– реалізовано віддалений доступ до системи та додатків;

– є можливість подальшого супроводу системи, наявність відповідних інструментів;

– інструментальні засоби для адаптації системи на іншу частину організації, наприклад при перенесенні даних на нову філію;

– інтеграція та консолідація нової інформації в систему;

– можливість обміну даними з іншими програмними продуктами, системами, програмами тощо, тобто можливість написання модулів;

– можливість огляду стану системи, аналіз процесу експлуатації.

Корпоративна система має також багато переваг застосування:

– власник підприємства чи керуючий будь-яким підрозділом, може швидко отримати дані про стан підприємства, відстежити переміщення товару, завдяки додатку, він може працювати з власного телефону;

– перегляд нинішніх процесів у підприємстві;

– висока результативність управління підприємством завдяки постійному контролю та безперервному оновленню даних у системі;

– зменшення часу збору замовлення та скорочення робочої операції.

Є багато видів інформаційних систем управління.

1.2 Архітектура та технології КІС

Розглядаючи архітектуру великих організацій, прийнято використовувати поняття «корпоративна архітектура».

Її можна представити у вигляді сукупності кількох типів архітектури:

- бізнес архітектура (Business architecture);
- IT-архітектура (Information Technology architecture);
- архітектура даних (Data architecture);
- програмна архітектура (Software architecture);
- технічна архітектура (Hardware architecture).

Модель корпоративної архітектури запропоновано на рисунку 1.3.

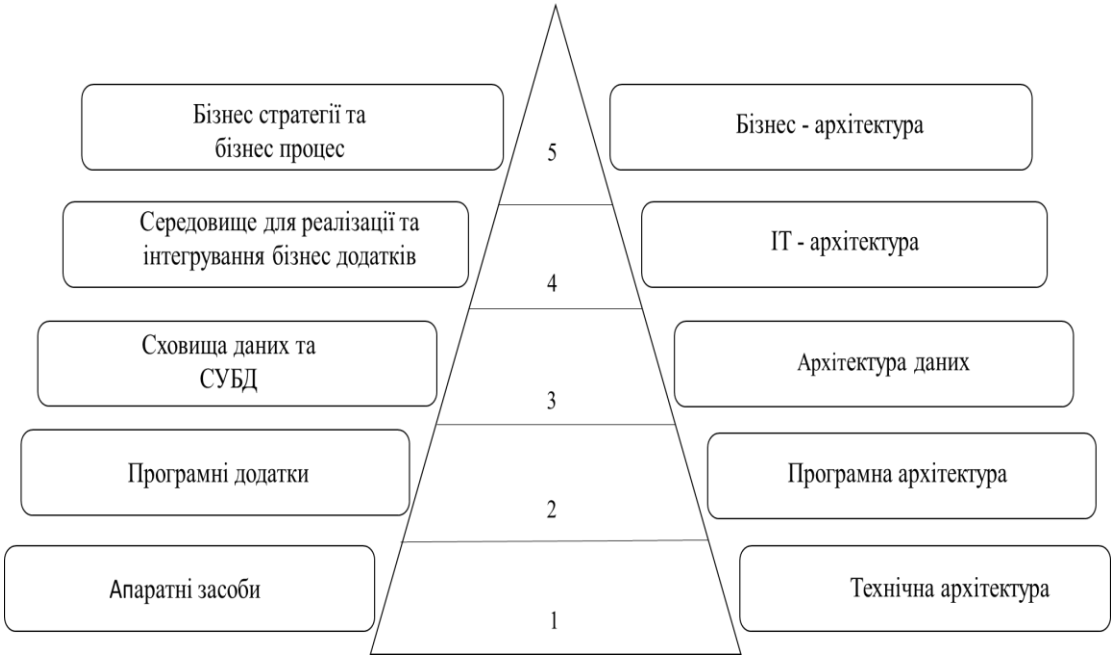


Рисунок 1.3 – Модель корпоративної інформаційної системи

Технічна архітектура є першим рівнем архітектури інформаційної системи. Вона визначає всі апаратні засоби, що використовуються при виконанні заявленого комплекту функцій, а також включає засоби забезпечення мережевої взаємодії та безпеки. У технічній архітектурі вказуються периферійні пристрої, мережеві комутатори та маршрутизатори,

жорсткі диски, оперативна пам’ять, процесори, з’єднувальні кабелі, джерела безперебійного живлення тощо.

Програмна архітектура є спільність комп’ютерних програм, призначених на вирішення певних завдань. Даний тип архітектури призначений для опису додатків, що входять до складу інформаційної системи. На цьому рівні описують програмні інтерфейси, складові та поведінку.

Архітектура даних поєднує у собі, як фізичні сховища даних, так і засоби управління даними. Крім того, до неї входять логічні сховища даних, а при орієнтованості розглянутої організації на роботу з вміннями може бути виділений відокремлений рівень – архітектура знань (Knowledge architecture). На цьому рівні описуються логічні та фізичні моделі даних, визначаються правила цілісності, складаються обмеження даних [8].

Слід особливо виділити рівень IT-архітектури, оскільки він є сполучним. На ньому формується базовий комплект сервісів, що застосовуються на рівні програмної архітектури та на рівні архітектури даних. Якщо будь-яка специфіка функціонування цих 2-х ярусів була передбачена, то сильно зростає ймовірність збоїв у роботі, отже, втрат для бізнесу. У деяких випадках неможливо розділити IT-архітектуру та архітектуру окремої програми. Це цілком можливо за високого ступеня інтеграції додатків. Прикладом IT-архітектури може бути SharePoint від організації Microsoft. Цей товар надає послуги для спільної роботи та зберігання інформації, що є дуже значним аспектом функціонування будь-якої організації. Його базові системні модулі відносяться до IT-архітектури, а користувацькі – до програмного. Головною функцією IT-архітектури є забезпечення функціонування головних бізнес-додатків для досягнення зазначених бізнес-цілей. Якщо деяка функція необхідна відразу у кількох додатках, її раціонально перенести рівень IT-архітектури, цим підвищивши інтеграцію системи та знизити складність архітектури додатків.

Останнім у ієрархії є рівень бізнес-архітектури чи архітектури бізнес процесів. На цьому рівні визначаються стратегії ведення бізнесу, методи управління, тези організації та основні процеси, що становлять для бізнесу велику важливість.

Процес проектування інформаційної системи тісним образом пов’язаний з її архітектурним викладом, що відображено в деяких визначеннях термінах «архітектура».

Можна виділити п’ять різних підходів до проектування:

– календарній підхід;
– підхід, за основу якого взято процес керування вимогами;
– підхід, встановлений на процесі розробки документації;
– підхід, заснований на системі керування якістю;
– архітектурний підхід.

Архітектурний підхід до проектування інформаційних систем можна вважати найбільш зрілим. Його ключовим аспектом є реалізація фрейм ворку, тобто каркаса, адаптація якого під потреби певної системи буде легко здійсненна. Відповідно до цього, завдання проектування розбивається на дві частини: розробки неодноразово використовуваного каркаса та реалізація системи на його основі. Слід зазначити, що ці під завдання можуть вирішуватися різними групами експертів. При застосуванні каркасів з’являється можливість досить швидко змінювати функціональність системи за рахунок інтерактивності процесу проектування. Архітектурний підхід покликаний усунути недоліки, що у процесі проектування, заснованому на управлінні вимогами.

Користувачі нових інформаційних систем практично завжди взаємодіють із нею з допомогою спеціальних програмних модулів, від показників якості яких залежить рівень якості всієї інформаційної системи.

Існує безліч стандартів для виробництва позитивної та правильної архітектури, а також для розробки та інтеграції програмних систем. Використання цих стандартів істотно збільшить шанси на успішну розробку

системи та її подальше безвідмовне функціонування, хоча раціональність їх застосування має визначатися на момент початку робіт, оскільки складність системи при їхній інтеграції може істотно зрости.

Якістю програмного забезпечення вважатимуться спільність його якостей, що характеризують можливість задовольняти певні чи умовні запити всіх зацікавлених осіб.

Можна виділити три аспекти якості:

– внутрішня якість (властивості програмного забезпечення);

– зовнішня якість (поведінкові властивості програмного забезпечення);

– контекстна якість (відчуття користувачів у різних контекстах застосування).

Керуючись цими аспектами, стандарт ISO 9126 виділяє шість властивостей якості програмного забезпечення:

– функціональність;

– довговічність;

– ефективність;

– зручність використання;

– зручність супроводу;

– переносимості.

Функціональність – це здатність ПО вирішувати завдання в певних умовах і ділиться на наступні підхарактеристик:

– функціональна придатність (suitability);

– здатність вирішувати потрібний комплект завдань;

– точність (accuracy) – здатність отримувати необхідні результати;

– здатність взаємодії (interoperability) – здатність взаємодії з необхідним комплектом інших систем;

– захищеність (security) – здатність запобігати неавторизованому доступу до даних та програм;

– відповідність еталонам та правилам (compliance) – відповідність програмного забезпечення різним регламентуючим нормам.

Довговічність (reability) характеризується здатністю програмного забезпечення тримати функціональність у заданих рамках за певних умов та ділиться на такі підхарактеристики:

– зрілість (maturity) – величина, обернена до частоти відмов програмного забезпечення;

– стійкість до відмов (fault tolerance);

– здатність тримати певний рівень працездатності при різних відмовах та порушеннях правил взаємодії з оточуючими;

– здатність до відновлення (recoverability) – здатність відновлювати необхідний рівень працездатності після відмови;

– відповідність безпеки (reability compliance).

Продуктивність (eficience) визначається здатністю програмного забезпечення за певних умов гарантувати необхідну працездатність відповідно до виділених для цього джерел. Можна також визначити, як ставлення одержуваних підсумків до витрачених джерел. Ця оцінка поділяється на наступні характеристики:

– відповідність ефективності (eficience compliance);

– зручність застосування (usability) характеризується привабливістю для користувачів, зручністю в навчанні та застосуванні програмного забезпечення. Має наступні підхарактеристики:

– зрозумілість (understandability) – величина зворотна зусиллям, витраченим користувачами усвідомлення застосовності програмного забезпечення на вирішення необхідних задач;

– зручність роботи (operability) – величина зворотна зусиллям, витраченим користувачами, на вирішення необхідних завдань із підтримкою програмного забезпечення;

– зручність навчання (learnability) – величина зворотна зусиллям, витраченим користувачами, на процес навчання роботі з програмним забезпеченням;

– привабливість (attractiveness) – здатність програмного забезпечення бути симпатичною для користувачів;

– відповідність стандартам комфорту застосування (usability compliance);

– зручність супроводу (maintainability) характеризується зручністю супроводу програмного забезпечення. Ця оцінка також включає низку підхарактеристик:

– аналізованість (analyzability) характеризується зручністю виконання огляду помилок, недоліків, недоліків, необхідності внесення змін та їх допустимих наслідків;

– зручність внесення змін (changeability) – величина зворотна трудовитрат на виконання необхідних змін;

– стабільність (stability) – величина зворотна ризику появи непередбачуваних наслідків при внесенні необхідних змін;

– зручність перевірки (testability) – величина зворотна необхідним трудовитратам на тестування й інші види перевірок досягнення передбачених результатів при внесенні змін;

– відповідність стандартам зручності супроводу (maintainability compliance).

Переносність (portability) характеризується здатністю програмного забезпечення зберігати працездатність при зміні організаційних, апаратних та програмних аспектів оточення. Для цієї властивості видаються наступні підсвистав:

– адаптованість (adaptability) – здатність програмного забезпечення без скоєння непередбачуваних дій пристосовуватися до змін оточення;

– зручність установки (installabity) – здатність програмного забезпечення встановлюватись у попередньо визначене оточення;

– здатність до співіснування (coexistence) – здатність програмного забезпечення працювати у загальному оточенні з іншими програмами, розділяючи із нею джерела;

– зручність заміни (replaceability) – можливість використання програмного забезпечення замість того, що вже використовується для вирішення тих же завдань, у тому ж оточенні;

– відповідність стандартам переносимості (portability compliance).

Всі ці характеристики описують внутрішню та зовнішню якість програмного забезпечення. Для викладення контекстної якості існує інший, зменшений комплект характеристик:

– ефективність (effectiveness) – здатність програмного забезпечення вирішувати завдання користувача із заданою точністю і в заданому контексті;

– продуктивність (productivity) – здатність програмного забезпечення отримувати потрібні результати при застосуванні попередньо визначеної кількості джерел;

– безпека (safety) – здатність програмного забезпечення підтримувати потрібний низький рівень ризику заподіяння збитків людям, бізнесу та навколишньому середовищу;

– задоволеність користувачів (satisfaction) – здатність програмного забезпечення при використанні в певному контексті приносити задоволення користувачам.

Керуючись показниками, що розглядаються, можна істотно збільшити якість програмних модулів, а, отже, і всієї інформаційної системи в цілому. Розглядаючи метод декомпозиції, прийнято проектувати інформаційні системи з розподілом функціонального призначення їх компонентів, тобто створювати багаторівневе уявлення [7, 8].

Можна виділити три основні функціональні групи, призначені для розв’язання різноманітних завдань:

– взаємодія з користувачами;

– бізнес-логіка;

– управління джерелами.

Реалізація такого функціоналу відбувається з допомогою виробництва відповідної програмної системи.

Така система має багаторівневе представлення компонентів (див. рис. 1.4).

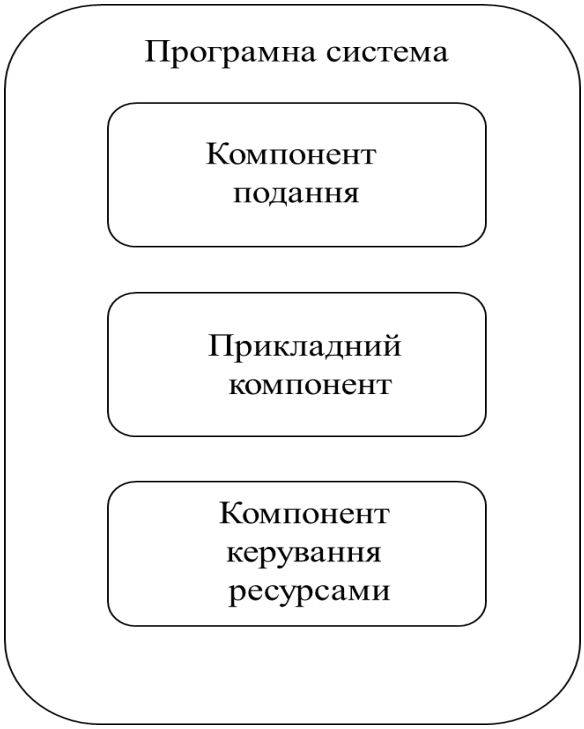


Рисунок 1.4 – Компоненти програмної системи

Компонент подання служить задля забезпечення взаємо функціонування користувачів із програмою, тобто обробляє натискання клавіш, руху різних контролерів, здійснює результат інформації – надає інтерфейс.

Прикладний компонент є комплект правил і алгоритмів реалізації функцій системи, реакцій на впливу користувачів чи внутрішніх обставин, обробки даних.

Компонент управління джерелами відповідає за зберігання, модифікацію, вибірку та видалення даних, пов’язаних з розв’язуваною прикладною задачею.

Одним із найважливіших етапів проектування архітектури інформаційної системи є поділ цих функціональних компонентів з обраної платформної архітектури.

1.3 Постановка задачі, застосування Python для реалізації бізнес-аналізу в корпоративній інформаційній системі

Поки не існує бездоганного програмного рішення для всіх бізнес-процесів, але технології ERP все краще об’єднують їх, удосконалюючи спільну роботу, допомагаючи приймати зважені рішення за допомогою даних і підвищуючи продуктивність бізнесу. ERP включає безліч функцій організації. Наприклад, фінанси – сучасні ERP-системи пропонують огляд фінансових показників в режимі реального часу з будь-якого пристрою. Це також допоможе відмовитися від введення даних вручну: автоматизувати щоденні завдання та відслідковувати відповідність бізнесу нормативним вимогам.

Фінансова компонента має вузький зв’язок із багатьма іншими компонентами системи, такими як:

– репозитарій контрактів;

– сторінка проектного коду;

– модуль внесення робочого часу та витрат на відрядження.

Умовно процес виставлення рахунків можна розділити на такі кроки:

– внесення підписаного договору проведення певних робіт у системі «репозитарій контрактів», оформлення картки документа;

– розробка сторінки проектного коду у компоненті «плани».

Проектні коди бувають TM (time and material), що ґрунтується на внесенні часу працівників у систему та Fx – координований певний бюджет на розробку, які у свою чергу формуються щомісяця (Date Driven) або при наданні письмової згоди замовника (Based on Acceptance).

Нижче зображено блок-схему функціонування компоненти виставлення рахунків у системі ERP.Netcracker.com, що є основним орієнтиром розробки майбутнього об’єкта кваліфікаційної роботи – модуля фінансової звітності.

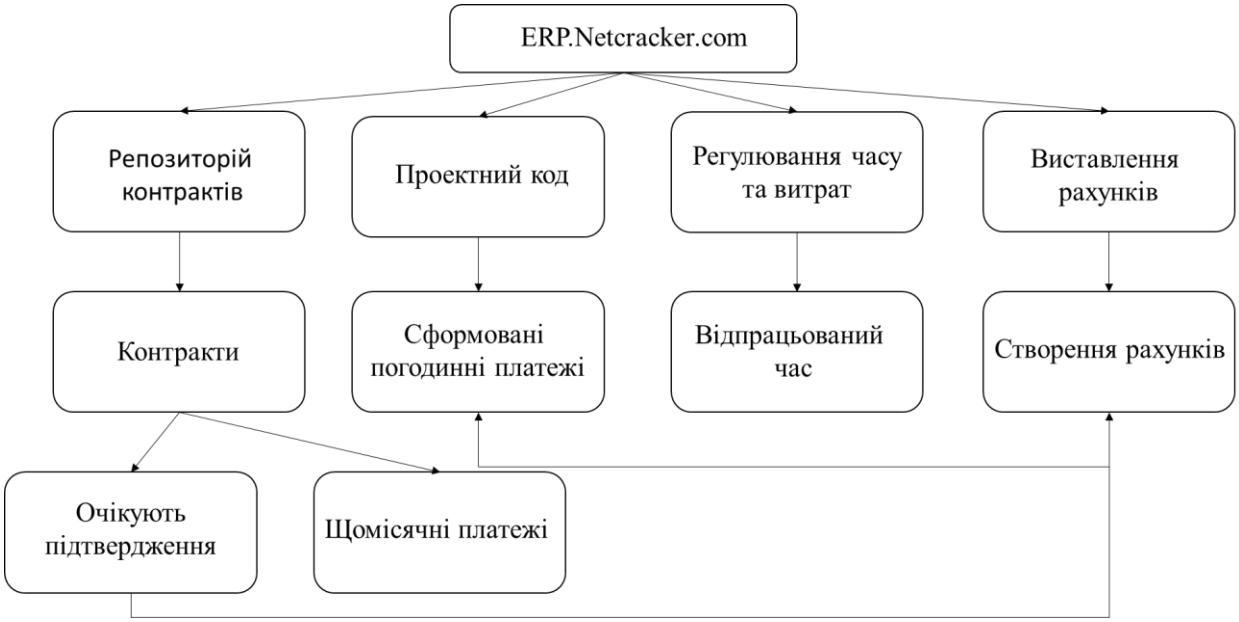


Рисунок 1.5 – Блок-схема процесу виставлення рахунку у системі

Після утворення рахунку важливо мати можливість відстежувати ранг їх оплати. Цей звіт можна отримати через інший модуль Звітності в системі ERP.Netcracer.com.

Метою роботи є розробка програмного модуля для автоматизації створення звітності з виставлених рахунків у відкритій ERP-системі, що ґрунтується на Odoo [21]. Даний модуль легко можна налаштувати і використовувати, зручність інтерфейсу з ймовірністю мануальна додавати необхідні зміни ERP-системи зі зразків або налаштувань з ймовірністю редагування на будь-яких етапах установки або видалення пакетів програмного забезпечення. Застосування цього модуля може скоротити час та підвищити якість роботи за рахунок мінімізації ймовірності помилок фінансових експертів.

Розроблений модуль матиме наступні функції:

– угруповання даних по виставлених рахунках у непостійний xls файл з підтримкою моделі TransientModel [20] та бібліотек Python IO, що включає дві вкладки: перша – всі рахунки замовникам, друга – сумарні рахунки по кожному замовнику окремо;

– можливість вибирати період для звітності (за умовчанням встановленого початку року до цієї дати);

– вибирати статус рахунків («все», «сплачені», «не сплачені»);

– підтримує мультивалютність для одного замовника (додає нові рядки для інших валют та підсумовує дані щодо різних валют окремо).

Розроблений програмний модуль розширює можливості існуючого модуля «Рахунки» та має інтуїтивний зручний і зрозумілий користувачеві інтерфейс.

Для реалізації проектного плану необхідно розв’язати такі завдання:

– визначити функціонал програмного модуля;

– вибрати технічні засоби реалізації;

– розробити код модуля;

– сформувати зразки налаштувань проектного плану;

– організувати тестування розробленого модуля.

2 РЕАЛІЗАЦІЯ БІЗНЕС-АНАЛІЗУ

2.1 Модель корпоративної інформаційної системи реалізації програмного модуля

Для реалізації програмного модуля були вибрані наступні засоби: JavaScript, HTML, CSS (візуальна частина), інтерпретована об’єктно-орієнтована мова програмування – Python (функціональна частина), json файл, середовище розробки PyCharm.

Архітектура модуля складається з трьох рівнів:

– замовна/візуальна частина (HTML, CSS, JavaScript);

– функціонально-логічна частина модуля (Python);

– вхідні дані json файл.

Нижче представлені засоби реалізації цих рівнів:

– презентаційний рівень (клієнт) – HTML, CSS, JS – MVC (View, Controller);

– прикладний рівень (серверна частина) – Python – MVC (Model) – один з найчастіше використовуваних у світі веб-розробок для виробництва особливо масштабованих і розширюваних програм [9].

(Model) – логіка даних, з якою працює користувач. Клієнтський об’єкт отримує інформацію про замовника з бази даних, маніпулює нею та оновлює ці дані у базу даних і навіть використовує їх візуалізації даних на інтерфейсі.

Перегляд (View) – застосовується для логіки інтерфейсу. Подання замовника буде включати всі компоненти інтерфейсу користувача, такі як текстові поля, меню, що випадають, і т.п., з якими взаємодіє останній користувач.

Контролер (Controller) виступає у ролі інтерфейсу між моделлю та виглядом, обробляє всю бізнес-логіку та запити. Клієнтський контролер оброблятиме всі взаємодії та входи за поданням форми замовника та

оновлюватиме базу даних з підтримкою модуля замовника, і той же контролер застосовуватиметься для перегляду даних замовника.

Рівень даних (база даних) – json файл. Серверні модулі розробляються на Python. ORM (Object, Relational, Mapping) взаємодії з базою даних є центральною частиною Odoo [16].

Модуль даних описується і маніпулюється за допомогою класів і об’єктів python. Завданням ORM є максимально прозоре подолання обриву для розробника між Python та базою даних PostgreSQL, що забезпечує сталість, потрібну для наших об’єктів.

JavaScript (JS) – динамічна, об’єктно-орієнтована прототипна мова програмування. Реалізація ECMAScript. Найчастіше застосовується для сценаріїв веб-сторінок, що дозволяє за замовника (пристрою фінального користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними із сервером, змінювати схему і зовнішній вигляд веб-сторінки [20]. JavaScript систематизують як прототипну (підмножину об’єктно-орієнтованого), скриптову мову програмування з динамічною типізацією. Крім прототипної JavaScript також частково підтримує інші парадигми програмування (імперативну і функціональну) і деякі необхідні архітектурні властивості, зокрема: динамічна і слабка типізація, механічне управління пам’яттю, прототипне наслідування, функції як об’єкти першого класу.

Мова JavaScript застосовується для:

– написання сценаріїв веб-сторінок для надання їм інтерактивності;
– реалізація одно сторінкових веб-додатків (React, AngularJS, Vue.js);
– програмування за сервера (Node.js(Express.js));
– нерухомих програм (Electron, NW.js);
– сценаріїв у додатку (наприклад, у програмах Adobe Creative Suite чи Apache JMeter);
– всередині PDF-файлів тощо.

Мова розмітки HyperText (HTML) – це звичайна мова розмітки, яка використовує загальні скорочення, звані «теги», щоб вказати веб-браузеру, як автори хотіли б, щоб були висловлені розділи веб-сторінки.

Каскадні таблиці жанрів – CSS – дають вам творчий контроль над макетом та дизайном ваших веб-сторінок. Застосовуючи CSS, ви можете увібрати свій текст веб-сайту з симпатичними заголовками, точками та рамками.

Ви також можете впорядковувати зображення з точністю, створювати стовпці та банери та виділіть свої посилання за допомогою динамічних результатів перекидання. Можна навіть зробити елементи зникаючими чи що виходять із поля зору, переміщують об’єкти сторінкою чи повільно змінюють кольору кнопки, коли відвідувач наводить курсор миші з неї [12].

За зберігання даних, необхідні роботи програмного модуля обрано стандарт XML через простоти редагування і інтеграції з середовищем розробки. XML файли із загальною конфігурацією та окремими налаштуваннями певних програм зберігаються на мережному диску, до якого є доступ як із хостових систем, так і віртуальних машин на підприємстві.

2.2 Програмна платформа та технології реалізації задач модуля звітності

Програмний модуль – функціонально закінчений уривок програми, оформлений у вигляді окремого файлу з кодом чи його іменованої частини, призначений до застосування у інших програмах. Модулі дозволяють розбивати важкі завдання на менші відповідно до принципу модульності. Зазвичай проектується таким чином, щоб надати програмістам зручну для багаторазового використання функціональність (інтерфейс) [9].

Для моделювання бізнес-процесів застосовується кілька різних методів, основою яких є як структурний, і об’єктно-орієнтований підходи до

моделювання. Одним із особливо поширених способів моделювання бізнес-процесів є спосіб функціонального моделювання SADT (IDEF0).

SADT-метод (Structured Analysis and Design Technique) вважається типовим способом процесного підходу до управління. Головний метод даного підходу полягає в класифікації діяльності організації відповідно до її бізнес-процесу, що є сукупністю правил і маніпуляцій, призначених для побудови функціональної моделі об’єкта будь-якої предметної області.

Функціональна модель SADT відбиває функціональну схему об’єкта, тобто функціонування, що він робить і зв’язок між цими діями [4]. IDEF0 – модель дозволяє отримати точну специфікацію всіх операцій та процесів, які є у бізнес-процесі, а також характер зв’язку поміж ними [4].

Особливо важлива функція розташована у верхньому лівому кутку. А з’єднуються функції між собою за допомогою стрілок та викладів функціональних блоків. У цьому будь-який вид стрілки чи активності має значення. Ця модель дозволяє описати всі основні види процесів, як адміністративних, так і організаційних.

Стрілки можуть бути:

– вхідні – вступні, які ставлять певне завдання;

– початкові – виводять підсумок діяльності;

– керівні (зверху донизу) – механізми керування (положення, інструкції тощо);

– механізми (знизу вгору) – застосовується у тому, щоб виконати необхідну роботу.

У результаті застосування IDEF0-моделювання зроблено діаграму плану кваліфікаційної роботи з подальшою декомпозицією (див. рис. 2.1).

Рисунок 2.1 – Діаграма кваліфікаційної роботи

Завантажити(Upload) файл ERP репорту. Вибираємо та завантажуємо файл (рис. 2.2).

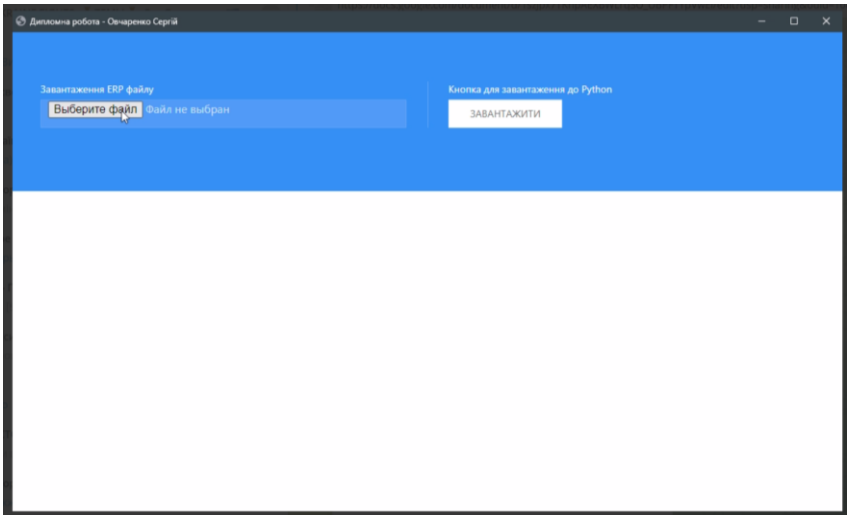


Рисунок 2.2 – Екран завантаження файлу

Після того як користувач завантажив файл, з’явилась функція перегляду файлу та перезавантаження нового файлу генерованого ERP системою (див. рис. 2.3).

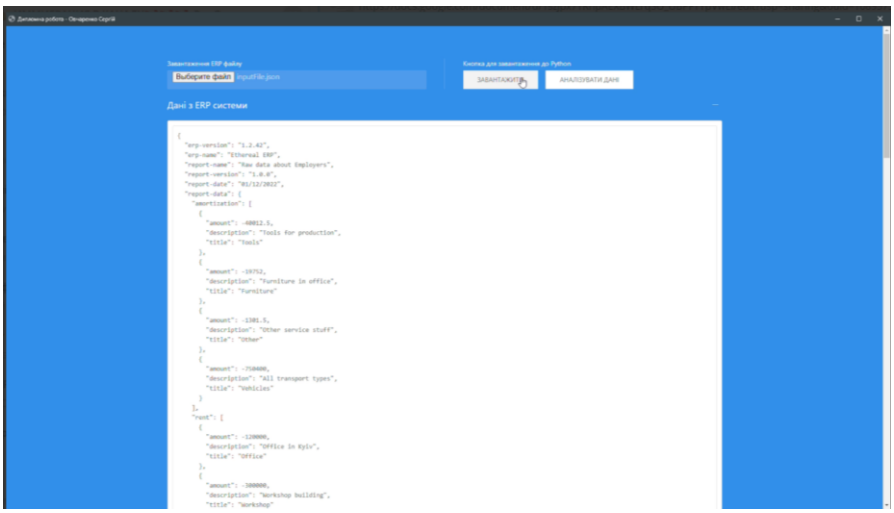


Рисунок 2.3 – Завантажені дані на сервер

На наступному етапі користувачу відкривається доступ до звітних файлів даних, згенерованих ERP системою та можливість аналізувати їх (див. рис. 2.4).

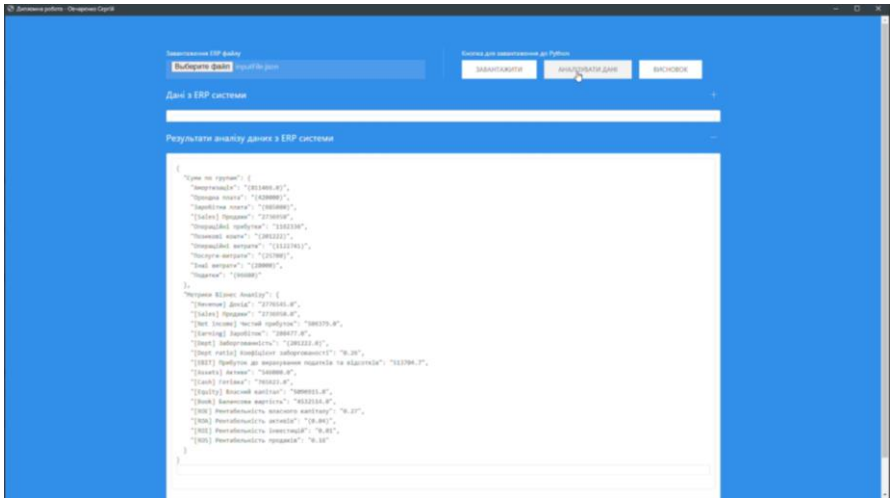


Рисунок 2.4 – Згенеровані дані ERP системою

Після завантаження файлу, згенерованого ERP системою та аналізу даних, за допомогою алгоритму розрахунку звітності підприємства, написаного на мові програмування Python, отримуємо висновок рентабельності підприємства (див. рис. 2.5).

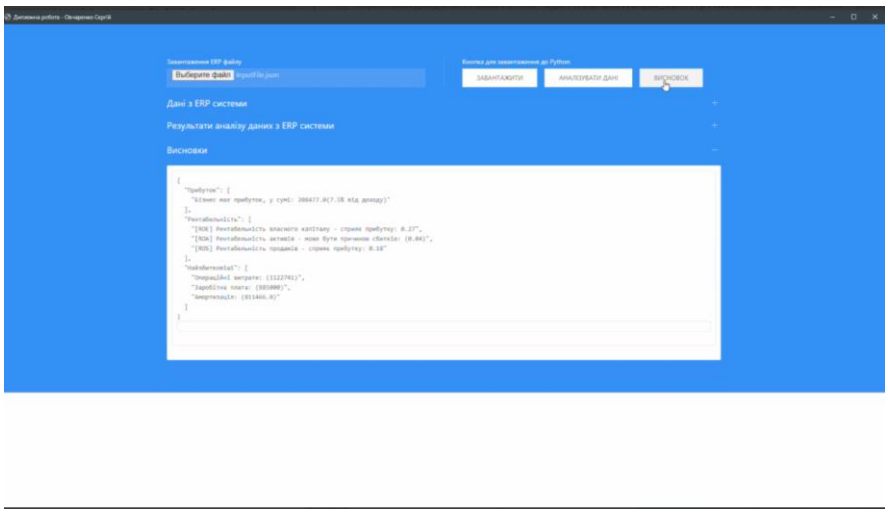


Рисунок 2.5 – Рентабельність підприємства

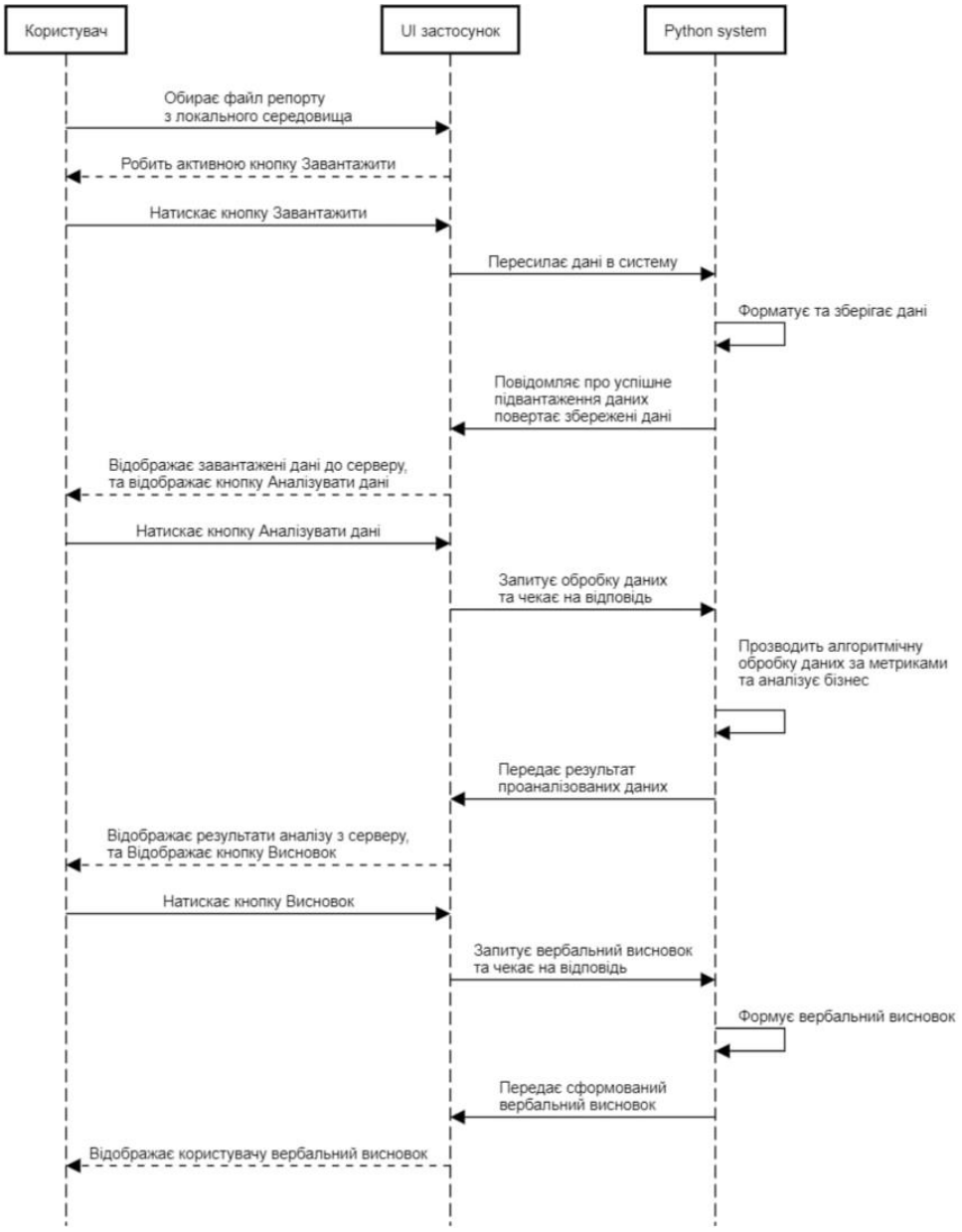


Рисунок 2.6 – Діаграма Sequence послідовності дій

Odoo модулі можуть додавати абсолютно нову бізнес-логіку, так і збільшувати існуючу. Модуль може бути зроблений для того, щоб додати план рахунків відповідної країни таким чином, щоб Odoo міг підтримувати саме ваш тип бухгалтерського обліку.

У той же час, інший модуль додає підтримку відображення стану в режимі реального часу [21].

Таким чином, все в Odoo починається і закінчується модулями склад модуля.

Модуль Odoo (див. рис. 2.7) може містити такі елементи:

– бізнес-об’єкти, оголошені як класи Python, Odoo механічно зберігає ці джерела за допомогою їхньої конфігурації;

– дані, файли XML чи CSV, які декларують метадані (подання чи робочі процеси), дані зміни (параметризація модулів), демонстраційні дані тощо;

– веб-контролери (Web Controllers), обробляє запити від веб-браузерів;

– статичні джерела, зображення, CSS чи javascript-файли, використовувані веб-інтерфейсом чи сайтом.

У файлі _init_.py нам доведеться імпортувати або зареєструвати або навіть сказати, ініціалізувати два елементи. По-перше, ви можете вказати каталоги, що містять файли python, які ми застосовували в модулі Odoo. По-друге, визначає файли python, які природно існують у модулі Odoo (крім папок), але нам не слід розміщувати python файлу в кореневому шляху модуля.

Файл _manifest_.py у _init_.py. Від того, що це провідний файл у модулі Odoo, він буде природно отримувати доступ із фрейм ворку Odoo для реєстрації модуля. _manifest_.py

Кожен модуль Odoo повинен мати файл _manifest_.py з точно цим іменем у корені модуля. Файл маніфесту описує важливу інформацію про ваш модуль. Наприклад, ім’я модуля та багато інших необов’язкових полів, які ви можете вказати.

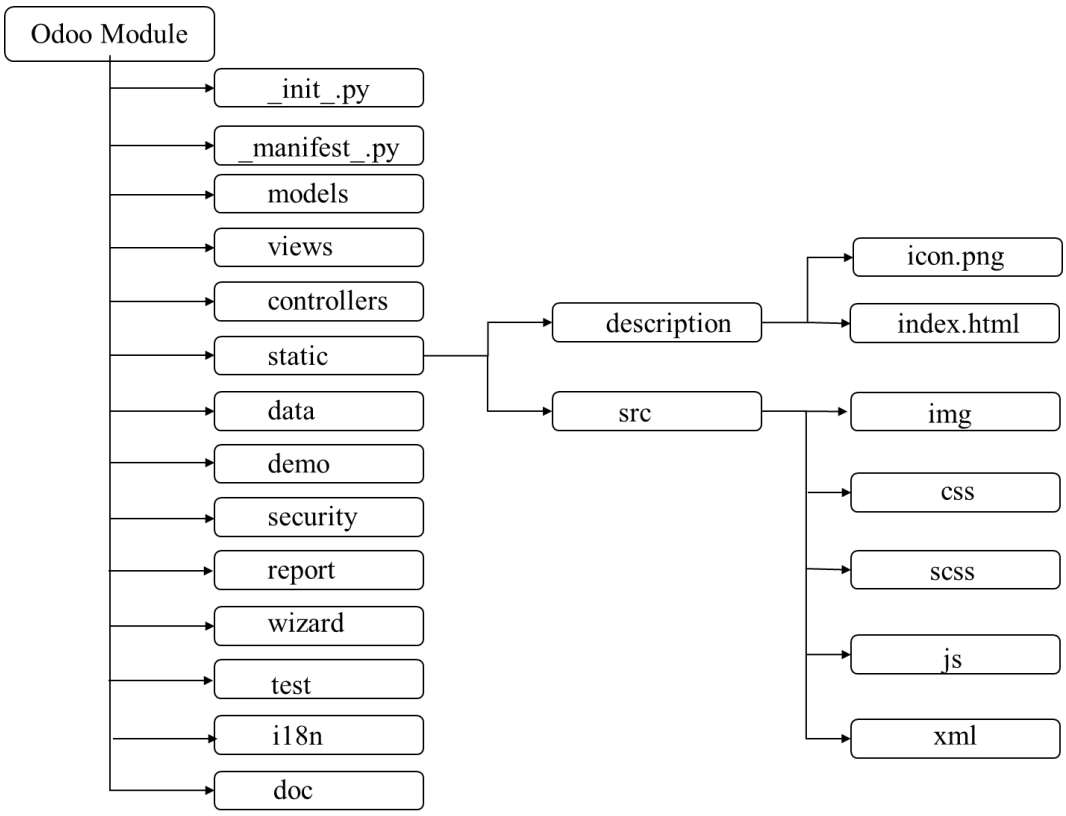


Рисунок 2.7 – Стандартна структура модулів ERP-системи Odoo

У файлі маніфесту ви можете оголосити такі поля:

– name – назва модуля;

– description – короткий опис модуля;

– version – вкажіть версію модуля;

– license – вкажіть ліцензію на розповсюдження модуля;

– author – ім’я автора модуля;

– website – URL-адреса веб-сайту автора модуля;

– category – вкажіть назву категорії, рекомендується використовувати існуючі категорії;

– depends – вказує список модулів, які встановлюються першими перед установкою модуля;

– data – список файлів даних, які завжди встановлюються або оновлюються разом із модулем;

– demo – список файлів даних, які встановлюються або оновлюються в активному демонстраційному режимі.

Моделі (Models). Каталог моделей містить усі ваші файли python (.py). Ці файли python містять моделі, які ви створюєте або успадковуєте існуючі моделі та основну бізнес-логіку. Усі файли python (.py) у будь-яких каталогах, потрібно імпортувати / зареєструвати у файл _init_.py у цьому відповідному каталозі.

Погляди (Views). Цей каталог містить усі файли .xml, де зберігається перегляд kanban, вигляд форми, меню, дію, шаблон Qweb та багато інших.

Контролери (Controllers). Цей каталог містить файли .py. Контролер – це клас, який походить від базового класу. Усі запити обробляються з браузерів і передаються на сервер. Це як середній міст для спілкування між моделями та поглядами. Усі файли python (.py) у будь-яких каталогах, потрібно імпортувати/зареєструвати у файл _init_.py у цьому шанованому каталозі.

Статичний (Static). Цей каталог містить опис і src два підкаталоги. Опис містить один файл icon.png, який представляє піктограму модуля у списку програм. А інший файл index.html містить документацію щодо модуля. Він наддасть інформацію про функціональність модуля разом із скріншотом. Src – містить багато підкаталогів:

- img – ця папка зберігає зображення, які ми використовуємо в модулі;
- css – зберігає один або кілька файлів CSS для застосування стилів;
- scss – зберігає один або кілька файлів SCSS для застосування стилів;
- js – містить один або кілька файлів JavaScript;
- xml – ця папка містить XML-файл, що містить збережений код QWeb, за допомогою JS він відобразиться.

Дані (Data). Файли .xml вмісту цього каталогу містять деякі заздалегідь визначені дані. Файли даних завжди завантажуються під час встановлення модуля.

Демо (Demo). Так само, як вище цього файлу вмісту каталогу .xml містять демонстраційні дані. Демо-файли завжди завантажуються під час встановлення модуля.

Безпека (Security). Ця папка містить два файли. Файл контролю доступу та файл правил запису. Контроль доступу: визначає права доступу моделей. Отже, вам потрібно створити файл ir.model.access.csv, який містить права доступу до моделей.

Правила запису. Взагалі правила запису – це умови. Це повинно бути задоволено для виконання таких операцій, як створення, оновлення до дозволених. Застосовується запис за записом після застосування контролю доступу. Отже, вам потрібно створити файл security.xml, який містить правила записів.

Доповідь (Report) – ця папка містить усі файли звітів. Тут зберігаються файли двох типів. По-перше, файли python (.py) для синтаксичного аналізу, а по-друге, файли qweb (.xml) для форматування дизайну звітів в Odoo.

Модальне вікно (Wizard). В основному це створить модальне вікно. Ця папка містить файли python та xml. Файл Python містить модель, яка розширює TransientModel. Більше того, ця модель є тимчасовою моделлю, яка буде автоматично видалена після завершення виконання. Крім того, XML-файл містить файли перегляду модальної версії.

Тест (Test). Файли для тестування зберігатимуться в цій папці. Усі тестові приклади – це файли .py та .yml із записом.

Локалізація (i18n). У цьому каталозі зберігаються файли .po та .pot. Ці файли зберігають переклад модуля. Один файл .pot, що зберігає рядки модулів, які ми хочемо перекласти. Тоді як .po файл зберігає весь вміст .pot файлу разом із перекладеним рядком.

Документація (Doc). Цей каталог містить документацію (файл формату .doc) модуля. Він надає інформацію про функціональність модуля.

2.3 Аналіз вимог до компонентів реалізації бізнес-аналізу

Під час проектування дипломної роботи було проведено огляд показників роботи ERP-систем ERP.Netcracker.com та Odoo на предмет отримання фінансової звітності, а саме таких показників як:

– час набуття сформованого звіту в системах;

– кількість виконаних кроків для отримання звіту.

Для отримання звіту в системі ERP.Netcracker.com потрібно виконати наступні кроки:

All reports>Finance>Archive (GAAP)>Billing & Invoicing (GAAP)>Invoicing Status Report

Всі звіти>Фінанси>Архів (GAAP)>Рахунки та виставлення рахунків (GAAP)> Звіт про стан виставлених рахунків>Передати період і статус.

Після отримання звіту потрібно за допомогою зведених таблиць зробити потрібний звіт.

Витрати часу для підготовки звіту становлять 30 хвилин.

На рисунку нижче (рис. 2.8) схематично зображені кроки реалізації для отримання звіту в системі ERP.Netcracker.com.

Рисунок 2.8 – Схема отримання фінансового звіту з рахунків в системі ERP.Netcracker.com

В ERP-системі Odoo схожий звіт можно отримати таким чином:

Invoicing>Reports>Invoice Summary Report

Виставлення Рахунків>Звітність>Роздрукувати Звіт про Рахунки>Обрати період та статус

Приблизний час отримання звіту – 3 хвилини.

Рисунок 2.9 – Схема отримання фінансового звіту з рахунків в системі Odoo

Отже, можна зробити висновок, що майбутній модуль виконується за меншу кількість кроків та за коротший проміжок часу, що підтверджує новизну та актуальність даного проєкту.

3 ВИБРАНИЙ МЕТОД РЕАЛІЗАЦІЇ БІЗНЕС АНАЛІЗУ

3.1 Реалізація задачі та демонстрація її інтегрування

View запропонована HTML (що розширює мову розмітки). Специфікація цієї мови дозволяє описувати поведінку програм, що зчитують дані шаблону та обробляють їх в браузері. Цей View містить записи з полями для вибору даних, які будуть розроблені в моделі.

У даному застосунку використовується доволі популярна модель побудування MVC (Model View Controller). У ролі моделі в нас виступає наш main.py, який виконує всю бізнес логіку застосунку, тобто виконує всі алгоритми та аналізує дані. Контролером виступає наш main.js файл, який є зв’язуючим елементом між нашою моделлю та переглядом, тобто він перехвачує всі дії користувача з перегляду та запитує інформацію з моделі. А переглядом або його ще називають уявленням у нашому застосунку є index.html, який займається візуалізацією нашого застосунку, тобто містить всі компоненти, кнопки, аккардіони та тексти.

Підсумовуючи наш HTML слухає та надає інформацію про дії користувача до JS, а JS вже безпосередньо комунікує з Python, а Python отримує запити від JS оброблює їх та повертає відповіді.

Структура нашого проекту складається з двох основних частин це бекенд виконаний на мові Python, та веденою частиною виступає фронтенд виконаний стандартною зв’язкою з HTML/CSS/JS. Ініціалізацію виконує файл main.py, який з самого початку запускає фронтенд застосунок з деякими параметрами, для того щоб локально на комп’ютері запустився фронтенд застосунок у спеціальній оболонці бібліотеки eel, а потім запущений локально сервер стає у режим прослуховування запитів до нього (див. рис. 3.1).

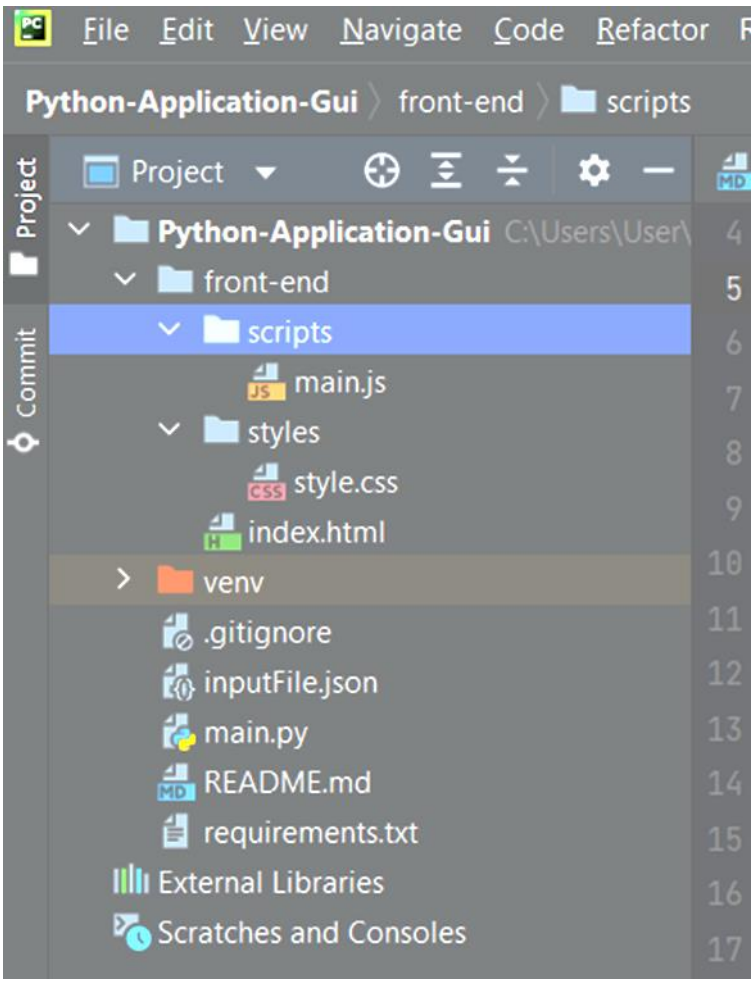


Рисунок 3.1 – Зміст файлу

Також представлені інтерактивні кнопки для підтвердження завантаження даних, аналізу даних та виведення висновків.

```
        <form class="uk-form-stacked uk-column-1-2 uk-column-divider">
            <div>
                <label class="uk-form-label">Завантаження ERP файлу</label>
                <div class="uk-form-controls">
                    <input class="uk-input" type="file" id="inputFile"
name="inputFile" accept="application/json">
                </div>
            </div>
            <div>
                <div class="uk-form-label">Кнопка для завантаження до Py-
thon</div>
                <div class="uk-form-controls main-controls">
                    <input type="button" class="uk-button uk-button-pri-
mary" value="Завантажити" onclick="init(eel.load)">
                    <input hidden id="analyse-button" type="button"
class="uk-button uk-button-secondary" value="Аналізувати дані" on-
click="process(eel.analyse)">
                    <input hidden id="conclusion-button" type="button"
class="uk-button uk-button-secondary" value="Висновок" onclick="fi-
nal(eel.conclusion)">
                </div>
            </div>
        </form>
```

Рисунок 3.2 – Запит на формування звіту

Представлено розділ в меню, який додається в існуючий модуль Завантаження ERP файлу.

```
<div>
    <label class="uk-form-label">Завантаження ERP файлу</label>
    <div class="uk-form-controls">
        <input class="uk-input" type="file" id="inputFile" name="inputFile" accept="application/json">
    </div>
</div>
```

Рисунок 3.3 – Додавання розділу Звітність

Спочатку відбувається імпорт файлів Python з ERP-системи Odoo.

```
from odoo import models, fields, api, _
import xlwt
import io
import base64
from xlwt import easyxf
import datetime
```

Рисунок 3.4 – Імпорт файлів Python з ERP-системи Odoo

Модель супер-класу для перехідних записів, призначена для тимчасового зберігання та регулярного чищення.

Далі в залежності від вибраних параметрів формується звіт у форматі HTML, що генерується за допомогою бібліотек Python. [1].

Файл містить 2 листи: 1 – дані всіх інвойсів (див. рис. 3.5), 2 – дані всіх інвойсів за замовниками (див. рис. 3.6).

```
#   "title": string;
# }
#
# interface IReportData {
#     "amortization": ICashFlow[];
#     "rent": ICashFlow[];
#     "operating-expenses": ICashFlow[];
#     "operating-profits": ICashFlow[];
#     "other-expenses": ICashFlow[];
#     "salaries": ICashFlow[];
#     "sales": ICashFlow[];
#     "services-expenses": ICashFlow[];
#     "taxes": ICashFlow[];
#     "borrowed-funds": ICashFlow[];
# }
#
# interface IBalanceData {
#     "assets": number;
#     "cash-funds": number;
#     "property-valuation": number;
# }
#
# interface IInputFile  {
#     "erp-version": string;
#     "erp-name": string;
#     "report-name": string;
#     "report-version": string;
#     "report-date": string;
#     "report-data": IReportData;
#     "balance-data": IBalanceData;
# }

@eel.expose
def load(input_file):
    global json_file
    # json_file = json.loads(inputFile)
    json_file = input_file

    return input_file
```

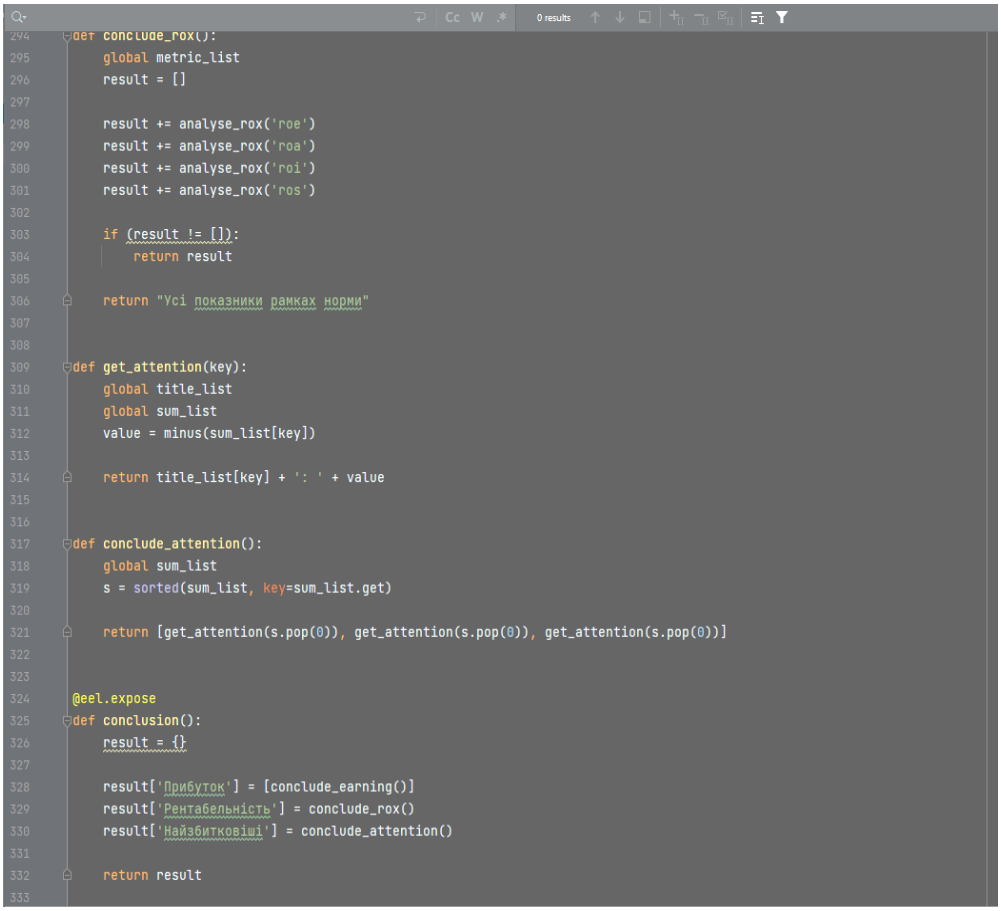
Рисунок 3.5 – Формування листа 1

```
                'borrowed-funds', 'operating-expenses', 'services-expenses', 'other-expenses', 'taxes']
    result = {}

    for item_key in sequence_list:
        result[title_list[item_key]] = minus(sum_list[item_key])
    return result


def getTitleByKey(key):
    return title_list[key] | key


def adapt_metric_list():
    global metric_list
    global title_list
    sequence_list = ['revenue', 'sales', 'net-income', 'earning',
                'dept', 'dept-ratio', 'ebit',
                'assets', 'cash', 'equity', 'book',
                'roe', 'roa', 'roi', 'ros']
    result = {}

    for item_key in sequence_list:
        result[title_list[item_key]] = minus(floor(metric_list[item_key]))

    return result


def adapt_user_result():
    # {'Amounts': sum_list, 'Metrics': metric_list}
    return {'Суми по групам': adapt_sum_list(), 'Метрики Бізнес Аналізу': adapt_metric_list()}


@eel.expose
def analyse():
    calculate_sum()
    calculate_metrics()

    return adapt_user_result()
```

Рисунок 3.6 – Формування листа 2

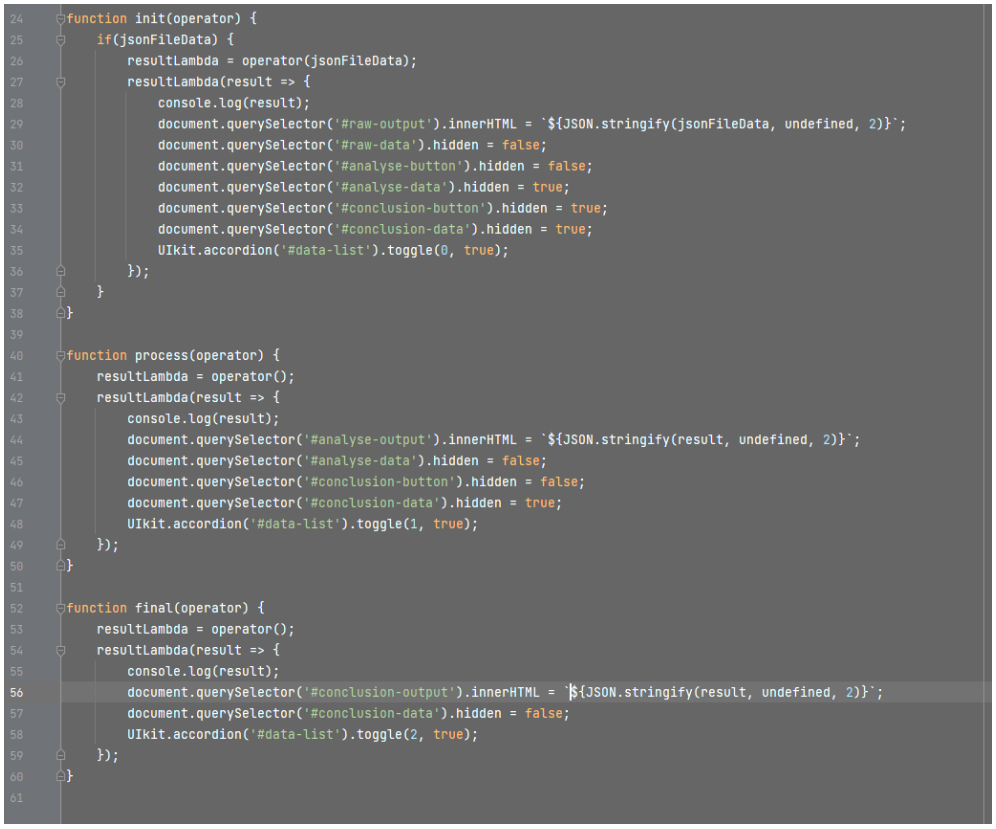
Нижче описано, як виходячи з обраних даних формується звіт (див. рис. 3.7).

На рисунку 3.8 описано як згенеровані дані з потоку UI зберігаються у файл формату HTML.



```python
def conclude_rox():
    global metric_list
    result = []

    result += analyse_rox('roe')
    result += analyse_rox('roa')
    result += analyse_rox('roi')
    result += analyse_rox('ros')

    if (result != []):
        return result

    return "Усі показники рамках норми"


def get_attention(key):
    global title_list
    global sum_list
    value = minus(sum_list[key])

    return title_list[key] + ': ' + value


def conclude_attention():
    global sum_list
    s = sorted(sum_list, key=sum_list.get)

    return [get_attention(s.pop(0)), get_attention(s.pop(0)), get_attention(s.pop(0))]


@eel.expose
def conclusion():
    result = {}

    result['Прибуток'] = [conclude_earning()]
    result['Рентабельність'] = conclude_rox()
    result['Найзбитковіші'] = conclude_attention()

    return result
```

Рисунок 3.7 – Формування звіту



```javascript
function init(operator) {
    if(jsonFileData) {
        resultLambda = operator(jsonFileData);
        resultLambda(result => {
            console.log(result);
            document.querySelector('#raw-output').innerHTML = `${JSON.stringify(jsonFileData, undefined, 2)}`;
            document.querySelector('#raw-data').hidden = false;
            document.querySelector('#analyse-button').hidden = false;
            document.querySelector('#analyse-data').hidden = true;
            document.querySelector('#conclusion-button').hidden = true;
            document.querySelector('#conclusion-data').hidden = true;
            UIkit.accordion('#data-list').toggle(0, true);
        });
    }
}

function process(operator) {
    resultLambda = operator();
    resultLambda(result => {
        console.log(result);
        document.querySelector('#analyse-output').innerHTML = `${JSON.stringify(result, undefined, 2)}`;
        document.querySelector('#analyse-data').hidden = false;
        document.querySelector('#conclusion-button').hidden = false;
        document.querySelector('#conclusion-data').hidden = true;
        UIkit.accordion('#data-list').toggle(1, true);
    });
}

function final(operator) {
    resultLambda = operator();
    resultLambda(result => {
        console.log(result);
        document.querySelector('#conclusion-output').innerHTML = `${JSON.stringify(result, undefined, 2)}`;
        document.querySelector('#conclusion-data').hidden = false;
        UIkit.accordion('#data-list').toggle(2, true);
    });
}
```

Рисунок 3.8 – Формування звіту у HTML за допомогою JS

3.2 Визначення вимог ефективності

Розроблений програмний модуль відповідає функціональним вимогам, а саме, реалізований функціонал:

– механічний збір даних за певний період та статус з ймовірністю їхньої наступної модифікації користувачем;

– зберігання отриманого звіту по зазначеному шляху на локальному диску.

Програмний модуль було розроблено засобами об’єктно-орієнтованої мови Python у середовищі розробки PyCharm.

Для роботи програмного модуля були створені зразки структури інструментарію розгортання пакетів програмного забезпечення та файли з конфігураціями програм за стандартом HTML.

Після виконання розробки провели тестування продукту.

Отже, виконавши всі етапи плану кваліфікаційної роботи, було розроблено програмний модуль звітності рентабельності підприємницької діяльності підприємства, що дозволить удосконалити показники роботи та автоматизувати процес перевірки звітів рентабельності підприємства. Застосування розробленого програмного забезпечення дозволить заощаджувати час на фінансові звіти.

ВИСНОВКИ

Під час реалізації кваліфікаційної роботи було розроблено програмний модуль автоматизованої звітності рентабельності підприємства Report для спрощення отримання та обробки звітності підприємства. Рішення було розроблено для використання працівниками відділу фінансів програмного забезпечення Ethereal ERP та будь-якими користувачами систем ERP.

У процесі реалізації проектного плану було обґрунтовано затребуваність роботи, проаналізовано існуючі аналоги та його переваги та недоліки, визначено функціональні вимоги до програмного модулю і визначено кошти реалізації.

Було виконано проектування робіт, за підсумками якого розроблено календарний план, а також визначено перелік ризиків під час виконання проектного плану.

ПЕРЕЛІК ПОСИЛАНЬ

1. Вирівнювання тексту в xlwt за допомогою easyxf. Питання – CodeRoad. URL: https://coderoad.ru/19900972/%D0%B2%D1%8B%D1%80%D0%B0%D0%B2%D0%BD%D0%B8%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5-%D1%82%D0%B5%D0%BA%D1%81%D1%82%D0%B0-%D0%B2-xlwt-%D1%81%D0%BF%D0%BE%D0%BC%D0%BE%D1%89%D1%8C%D1%8E-easyxf (дата звернення: 28.09.2022).

2. Данжу Д. Путь Python. Чорний пояс з розробки, масштабування, тестування та розгортання. Санкт-Петербург: Пітер, 2020. 258 с.

3. Знайомство з нотацією IDEF0 та приклад використання. Habr. URL: https://habr.com/ua/company/trinion/blog/322832/ (дата звернення: 30.09.2022).

4. Методологія функціонального моделювання IDEF0. Керівний документ IDEF0-2000. Москва: Держстандарт Росії, 2000. 115 с.

5. Одиночкина С. В. Основи технологій XML. Навчальний посібник. Санкт-Петербург: НДУ ІТМО, 2013. 56 с.

6. О'Лірі Д. ERP системи: сучасне планування та управління ресурсами підприємства. Вибір, використання, експлуатація. Москва: ТОВ «Вершина», 2004. 272 с.

7. Плескач В. Л., Затонацька Т. Г. Інформаційні системи і технології на підприємствах: підручник. Київ: Знання, 2011. 718 с.

8. Побудова ефективної системи управління / пер. з англ. Москва: Альпіна Бізнес Букс, 2008. 346 с.

9. Програмний модуль. Вікіпедія. URL: https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B8%D0%B9_%D0%BC%D0%BE%D0%B4%D1%83%D0%BB%D1%8C#:~:text=%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C%20%E2%80%94%20%D1%84%D1%83%D0%BD%D0%BA%D

1%86%D1%96%D0%BE%D0%BD%D0%B0%D0%BB%D1 (дата звернення: 10.10.2022).

10. Чорненька Л. В. Корпоративні інформаційні системи. Baan ERP : навч. посібник. Санкт-Петербург: Вид-во Політехнічного ун-ту, 2008. 331 с.

11. 2020 Clash Of The Titans: SAP Vs. Oracle Vs. Microsoft Vs. Infor. URL: https://www.panorama-consulting.com/resource-center/clash-of-the-titans-2020-sap-vs-oracle-vs-microsoft-dynamics-vs-infor/ (дата звернення: 03.11.2022).

12. McFarland D. S. CSS3 The Missing Manual. O'Reilly Media, Inc., 2013. 638 p. ISBN-13:978-1449325947.

13. Freeman E., Robson E., Bates B., Sierra K. Head First Design Pattern. O'Reilly Media, 2004. 694 p. ISBN-13:978-0596007126.

14. Freeman A. Pro Entity Framework Core 2 for ASP.NET Core MVC. Apress, 2018. 650 p. ISBN-13:978-1-4842-3434-1.

15. Klaus-Dieter G. Integrated Business Information Systems: A Holistic View of the Linked Business Process Chain ERP-SCM-CRM-BI-Big Data. Springer, 2017. 206 p. ISBN 978-3-662-53290-4.

16. Halpin T. Object-Role Modeling Fundamentals: A Practical Guide to Data Modeling with ORM. Basking Ridge: Technics Publications, 2015. 151 p. ISBN-13:978-1634620741.

17. Hsieh C. Y., Chen, C. T. Patterns for Continuous Integration Builds in Cross-Platform Agile Software Development. *J. Inf. Sci. Eng.* 2015. 31(3). P. 897–924.

18. Lincoln C. R. Process Improvement Essentials-CMMI, ISO 9001, Six Sigma. *Software Quality Professional*. 2007. *9*(4). P. 48.

19. McConnell S. Software estimation: demystifying the black art. *Microsoft press*. 2006.

20. Mike McGrath. HTML5 in easy steps. In Easy Steps, 2011. 240 p. ISBN-13:978-1840787542.

21. Module Structure. Open Source ERP and CRM | Odoo. URL: https://www.odoo.com/documentation/14.0/howtos/backend.html#module-

structure (дата звернення: 05.10.2022).

22. Patwardhan A., Kidd J., Urena T., Rajgopalan A. Embracing Agile methodology during Developer Internship Program. 2016. arXiv preprint arXiv:1607.01893.

23. Sacks M. Devops principles for successful web sites. In *Pro Website Development and Operations*. Apress, Berkeley, CA. 2012. P. 1–14.

24. SAP ERP. Побудова ефективної системи управління / пер. з англ. Москва: Альпіна Бізнес Букс, 2008. 346 с.

25. Schenker Gabriel N. Learn Docker – Fundamentals of Docker 19.x: Build, test, ship, and run containers with Docker and Kubernetes / 2nd Edition. Packt Publishing Ltd., 2020. 574 p. ISBN 978-1-83882-747-2.

26. Wallace T. F., Kremzar M. H. ERP: Making It Happen: The Implementers' Guide To Success With Enterprise Resource Planning. *John Wiley & Sons, Inc*., 2001. 385 p. ISBN 978-0-471-39201-9.

27. What is TransientModel class. ODOO NINJA. URL: http://www.odooninja.com/what-is-transientmodel-classes/ (дата звернення: 12.11.2022).

ДОДАТОК А

Лістинг програмного коду

A.1 Python

```
import eel
import math
import json
import functools
from functools import reduce

eel.init('front-end')

metric_list = {}
sum_list = {}
json_file = {}

title_list = {
    'revenue': '[Revenue] Дохід',
    'sales': '[Sales] Продажи',
    'net-income': '[Net income] Чистий прибуток',
    'earning': '[Earning] Заробіток',
    'dept': '[Dept] Заборгованність',
    'dept-ratio': '[Dept ratio] Коефіцієнт заборгованості',
    'ebit': '[EBIT] Прибуток до вирахування податків та
відсотків',
    'assets': '[Assets] Активи',
    'cash': '[Cash] Готівка',
    'equity': '[Equity] Власний капітал',
    'book': '[Book] Балансова вартість',
    'roe': '[ROE] Рентабельність власного капіталу',
    'roa': '[ROA] Рентабельність активів',
    'roi': '[ROI] Рентабельність інвестицій',
    'ros': '[ROS] Рентабельність продажів',
    'amortization': 'Амортизація',
    'rent': 'Орендна плата',
    'salaries': 'Заробітна плата',
    'operating-profits': 'Операційні прибутки',
    'borrowed-funds': 'Позикові кошти',
    'operating-expenses': 'Операційні витрати',
    'services-expenses': 'Послуги-витрати',
```

```
    'other-expenses': 'Інші витрати',
    'taxes': 'Податки'
}

### interface
# interface ICashFlow {
#   "amount": number;
#   "description": string;
#   "title": string;
# }
#
# interface IReportData {
#     "amortization": ICashFlow[];
#     "rent": ICashFlow[];
#     "operating-expenses": ICashFlow[];
#     "operating-profits": ICashFlow[];
#     "other-expenses": ICashFlow[];
#     "salaries": ICashFlow[];
#     "sales": ICashFlow[];
#     "services-expenses": ICashFlow[];
#     "taxes": ICashFlow[];
#     "borrowed-funds": ICashFlow[];
# }
#
#     "sales": ICashFlow[];
#     "services-expenses": ICashFlow[];
#     "taxes": ICashFlow[];
#     "borrowed-funds": ICashFlow[];
# }
#
# interface IBalanceData {
#     "assets": number;
#     "cash-funds": number;
#     "property-valuation": number;
# }
#
# interface IInputFile  {
#     "erp-version": string;
#     "erp-name": string;
#     "report-name": string;
#     "report-version": string;
#     "report-date": string;
#     "report-data": IReportData;
#     "balance-data": IBalanceData;
# }
```

```
@eel.expose
def load(input_file):
    global json_file
    # json_file = json.loads(inputFile)
    json_file = input_file

    return input_file


def get_amount(cash_flow):
    return cash_flow['amount']


def calculate_sum():
    global sum_list
    global json_file
    sum_list = {}
    for key in json_file['report-data'].keys():
        sum_list[key] = sum(map(get_amount, json_file['report-
data'][key]))


def calculate_assets():
    global json_file
    global metric_list
    metric_list['assets'] = json_file['balance-data']['assets']


def calculate_book():
    global metric_list
    global json_file
    metric_list['book'] = json_file['balance-data']['property-
valuation']


def calculate_cash():
    global json_file
    global metric_list
    metric_list['cash'] = json_file['balance-data']['cash-
funds']



def calculate_dept_ratio():
    # відношення позикових засобів до власного капіталу
    global metric_list
```

```
    metric_list['dept-ratio'] = abs(metric_list['dept'] /
metric_list['cash'])


def calculate_earning():
    global sum_list
    global metric_list
    metric_list['earning'] = sum(sum_list.values())


def calculate_ebit():
    # ROS = EBIT / revenue * 100
    # EBIT = ROS * revenue
    global metric_list
    metric_list['ebit'] = (metric_list['ros'] *
metric_list['revenue'])


def calculate_equity():
    # equity = capital - dept + cash
    global metric_list
    metric_list['equity'] = metric_list['book'] +
metric_list['dept'] + metric_list['cash']


def calculate_dept():
    global sum_list
    global metric_list
    metric_list['dept'] = sum_list['borrowed-funds']


def calculate_net_income():
    global sum_list
    global metric_list
    metric_list['net-income'] = metric_list['earning'] -
sum_list['taxes'] - metric_list['dept']


def calculate_revenue():
    global sum_list
    global metric_list
    metric_list['revenue'] = metric_list['sales'] +
sum_list['operating-profits'] + sum_list['operating-expenses']


def calculate_roa():
```

```
    # ROA = dept / equity
    global metric_list
    metric_list['roa'] = metric_list['dept'] /
metric_list['equity']


def calculate_roe():
    # Рентабельність власного капіталу Colgate Dupont =
    # (Чистий прибуток / Продаж) x (Продажі / Загальна сума
активів) x (Загальна сума активів / Власний капітал).
    global metric_list
    metric_list['roe'] = (metric_list['earning'] /
metric_list['sales']) * (
(metric_list['assets'] / metric_list['cash'])


def calculate_roi():
    # ROI = (revenue - sales) / sales
    global metric_list
    metric_list['roi'] = (metric_list['revenue'] -
metric_list['sales']) / (metric_list['sales'])


def calculate_ros():
    # ROS = Net income (before dept and tax) / Sales
    global metric_list
    metric_list['ros'] = (metric_list['net-income'] /
metric_list['sales'])


def calculate_sales():
    global sum_list
    global metric_list
    metric_list['sales'] = sum_list['sales']


def calculate_metrics():
    global metric_list
    metric_list = {}
    calculate_earning()
    calculate_dept()
    calculate_sales()
    calculate_revenue()
    calculate_assets()
    calculate_cash()
    calculate_book()
```

```
    calculate_equity()
    calculate_net_income()

    calculate_dept_ratio()

    calculate_roe()
    calculate_roa()
    calculate_roi()
    calculate_ros()
    calculate_ebit()


def minus(value):
    if (value < 0):
        return '(' + str(abs(value)) + ')'

    return str(value)


def adapt_sum_list():
    global sum_list
    global title_list
    sequence_list = ['amortization', 'rent', 'salaries',
'sales',
                'operating-profits',
                'borrowed-funds', 'operating-expenses',
'services-expenses', 'other-expenses', 'taxes']
    result = {}

    for item_key in sequence_list:
result[title_list[item_key]] = minus(sum_list[item_key])
    return result


def getTitleByKey(key):
    return title_list[key] | key


def adapt_metric_list():
    global metric_list
    global title_list
    sequence_list = ['revenue', 'sales', 'net-income',
'earning',
                'dept', 'dept-ratio', 'ebit',
                'assets', 'cash', 'equity', 'book',
                'roe', 'roa', 'roi', 'ros']
```

```
    result = {}

    for item_key in sequence_list:
        result[title_list[item_key]] =
minus(floor(metric_list[item_key]))

    return result


def adapt_user_result():
    # {'Amounts': sum_list, 'Metrics': metric_list}
    return {'Суми по групам': adapt_sum_list(), 'Метрики Бізнес
Аналізу': adapt_metric_list()}


@eel.expose
def analyse():
    calculate_sum()
    calculate_metrics()

    return adapt_user_result()


def floor(number):
    return math.floor(number * 100) / 100.0


def percent_income():
    global metric_list
    return floor(metric_list['earning'] / metric_list['revenue']
* 100)


def conclude_earning():
    global metric_list
    if metric_list['earning'] > 0:
        return "Бізнес має прибуток, у сумі: " +
str(metric_list['earning']) + '(' + str(
            percent_income()) + '% від доходу)'
    elif metric_list['earning'] < 0:
        return "Бізнес зазнає збитків, у сумі: " +
minus(metric_list['earning'])

    return "Бізнес має нульовий результат (0% від доходу)"
```

```
def analyse_rox(key):
    global metric_list
global title_list
    if metric_list[key] < 0:
        return [title_list[key] + " - може бути причиною
збитків: " + minus(floor(metric_list[key]))]
    if metric_list[key] > 0.15:
        return [title_list[key] + " - сприяє прибутку: " +
minus(floor(metric_list[key]))]
    return []


def conclude_rox():
    global metric_list
    result = []

    result += analyse_rox('roe')
    result += analyse_rox('roa')
    result += analyse_rox('roi')
    result += analyse_rox('ros')

    if (result != []):
        return result

    return "Усі показники рамках норми"


def get_attention(key):
    global title_list
    global sum_list
    value = minus(sum_list[key])

    return title_list[key] + ': ' + value


def conclude_attention():
    global sum_list
    s = sorted(sum_list, key=sum_list.get)

    return [get_attention(s.pop(0)), get_attention(s.pop(0)),
get_attention(s.pop(0))]


@eel.expose
def conclusion():
    result = {}
```

```
    result['Прибуток'] = [conclude_earning()]
    result['Рентабельність'] = conclude_rox()
    result['Найзбитковіші'] = conclude_attention()

    return result


eel.start('index.html', size=(1200, 720))
```

A.2 JavaScript

```
var jsonFileData;
(function(){

    function onChange(event) {
        var reader = new FileReader();
        reader.onload = onReaderLoad;
        reader.readAsText(event.target.files[0]);
        document.querySelector('#analyse-button').hidden = true;
        document.querySelector('#analyse-data').hidden = true;
        document.querySelector('#conclusion-data').hidden =
true;
        document.querySelector('#conclusion-data').hidden =
true;
    }

    function onReaderLoad(event){
        console.log(event.target.result);
        jsonFileData = JSON.parse(event.target.result);
    }
    setTimeout(() => {

document.querySelector('#inputFile').addEventListener('change',
onChange);
    }, 1000);

}());

function init(operator) {
    if(jsonFileData) {
        resultLambda = operator(jsonFileData);
```

```
        resultLambda(result => {
            console.log(result);
            document.querySelector('#raw-output').innerHTML =
`${JSON.stringify(jsonFileData, undefined, 2)}`;
            document.querySelector('#raw-data').hidden = false;
            document.querySelector('#analyse-button').hidden =
false;
            document.querySelector('#analyse-data').hidden =
true;
            document.querySelector('#conclusion-button').hidden
= true;
            document.querySelector('#conclusion-data').hidden =
true;
            UIkit.accordion('#data-list').toggle(0, true);
        });
    }
}

function process(operator) {
    resultLambda = operator();
    resultLambda(result => {
        console.log(result);
        document.querySelector('#analyse-output').innerHTML =
`${JSON.stringify(result, undefined, 2)}`;
        document.querySelector('#analyse-data').hidden = false;
        document.querySelector('#conclusion-button').hidden =
false;
        document.querySelector('#conclusion-data').hidden =
true;
        UIkit.accordion('#data-list').toggle(1, true);
    });
}

function final(operator) {
    resultLambda = operator();
    resultLambda(result => {
        console.log(result);
        document.querySelector('#conclusion-output').innerHTML =
`${JSON.stringify(result, undefined, 2)}`;
        document.querySelector('#conclusion-data').hidden =
false;
        UIkit.accordion('#data-list').toggle(2, true);
    });
}
```

A.3 HTML

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="description" content="">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1, shrink-to-fit=no">
    <!-- UIkit CSS -->
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/uikit@3.15.16/dist/css/uikit.
min.css" />

    <!-- UIkit JS -->
    <script
src="https://cdn.jsdelivr.net/npm/uikit@3.15.16/dist/js/uikit.mi
n.js"></script>
    <script
src="https://cdn.jsdelivr.net/npm/uikit@3.15.16/dist/js/uikit-
icons.min.js"></script>

    <title>Дипломна робота - Овчаренко Сергій</title>
    <link rel="stylesheet" href="styles/style.css">
    <script type="text/javascript" src="/eel.js"></script>
    <script type="text/javascript"
src="scripts/main.js"></script>

</head>

<body class="uk-section uk-section-primary">
    <div class="uk-container">
        <form class="uk-form-stacked uk-column-1-2 uk-column-
divider">
            <div>
                <label class="uk-form-label">Завантаження ERP
файлу</label>
                <div class="uk-form-controls">
                    <input class="uk-input" type="file"
id="inputFile" name="inputFile" accept="application/json">
                </div>
            </div>
            <div>
                <div class="uk-form-label">Кнопка для
```

```
завантаження до Python</div>
                <div class="uk-form-controls main-controls">
                    <input type="button" class="uk-button uk-
button-primary" value="Завантажити" onclick="init(eel.load)">
                    <input hidden id="analyse-button"
type="button" class="uk-button uk-button-secondary"
value="Аналізувати дані" onclick="process(eel.analyse)">
                    <input hidden id="conclusion-button"
type="button" class="uk-button uk-button-secondary"
value="Висновок" onclick="final(eel.conclusion)">
                </div>
            </div>
        </form>

        <ul uk-accordion id="data-list">
            <li id="raw-data" hidden>
                <a class="uk-accordion-title" href="#">Дані з
ERP системи</a>
                <div class="uk-accordion-content"><pre id="raw-
output"></pre></div>
            </li>
            <li id="analyse-data" hidden>
                <a class="uk-accordion-title"
href="#">Результати аналізу даних з ERP системи</a>
                <div class="uk-accordion-content"><pre
id="analyse-output"></pre></div>
            </li>
            <li id="conclusion-data" hidden>
                <a class="uk-accordion-title"
href="#">Висновки</a>
                <div class="uk-accordion-content"><pre
id="conclusion-output"></pre></div>
            </li>
        </ul>
    </div>
</body>

</html>
```