НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено:
Завідувач кафедри
_____Олександр Коваль
«___»_____2021 р.

Дипломна робота

на здобуття ступеня бакалавра
спеціальності 122 «Комп'ютерні науки»
освітня програма «Комп'ютерний моніторинг та геометричне моделювання процесів і систем»

на тему: «Створення компонент бухгалтерського обліку на базі ERP системи Odoo 13 Community Edition»

Виконав (-ла):
студент (-ка) IV курсу, групи ТМ-71
Тимошенко Іван Юрійович _____

Керівник:
професор кафедри АПЕПС, д.т.н.
Федорова Наталія Володимирівна _____

Рецензент:
доцент кафедры ТЕУТ і АЕС, к.т.н
Сірий Олександр Анатолійович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2021 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

спеціальність 122 Комп’ютерні науки

освітня програма «Комп’ютерний моніторинг та геометричне моделювання процесів і систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр Коваль
(підпис)
”___”_____2021р.

ЗАВДАННЯ
на дипломну роботу студенту

Тимошенко Іван Юрійович
(прізвище, ім’я, по батькові)

1. Тема роботи Створення компонент бухгалтерського обліку на базі ERP системи Odoo 13 Community Edition
керівник роботи Федорова Наталія Володимирівна, д.т.н, професор
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”24” травня 2021р. № **1267-с**

2. Строк подання студентом роботи ”08” червня 2021
3. Вихідні дані до роботи Мова програмування Python, Odoo 13 Community Edition, середовище розробки PyCharm, Git, Jira, Kubernets

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) провести аналіз бухгалтерського обліку, створити модуль бухгалерського обліку, проаналізувати засоби розробки системи Odoo 13 Community Edition

5. Перелік ілюстративного матеріалу мета розробки, система Odoo 13 Community Edition, засоби розробки, Презентація модуля, розгортання, структура проекту, інструкція користувача, висновки

6. Консультанти розділів роботи

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

7. Дата видачі завдання ” 10 ” жовтня 2020 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	9.12.20	
2.	Вивчення та аналіз задачі	15.12.20 – 29.01.21	
3.	Розробка архітектури та загальної структури системи	30.01.21 – 20.02.21	
4.	Розробка структур окремих підсистем	21.02.21 – 24.03.21	
5.	Програмна реалізація системи	25.03.21 – 24.04.21	
6.	Оформлення пояснювальної записки	25.04.21 – 10.05.21	
7.	Захист програмного продукту	11.05.21	
8.	Передзахист	25.05.21	
9.	Захист	16.06.21	

Студент _____ Тимошенко І.Ю.. ____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Федорова Н.В. _____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Дипломну роботу розроблемно на 61 сторінках, в ній використано 22 рисунки і 1 діаграму. Також включає в себе 21 бібліографічних найменувань за “Списком використаних джерел” та 3 додатки.

Метою даної дипломної роботи є опис процесу створення модуля на базі системи Odoo 13 Community Edition та розробка бухгалтерського модуля.

В ході реалізації модуля було використано віддалений сервер, віртуальну машину, IDE PyCharm, документацію Odoo для розробників.

Ключові слова: ERP система, CRM, бухгалтерський облік, Odoo 13 Community Edition, розробка

ABSTRACT

The developed work is developed on 61 pages, in it 22 drawings and 1 diagram are used. It also contains 21 bibliographic names for "List of used sources" and 3 additions.

The purpose of this thesis is to describe the process of creating a module based on the system Odoo 13 Community Edition and the development of an accounting module.

During the implementation module the remote server, the virtual machine, IDE PyCharm, Odoo documentation for developers were used.

Keywords: ERP system, CRM, accounting, Odoo 13 Community Edition, development

ЗМІСТ

ВСТУП

Сьогодні жодна людина не може уявити свого життя без гаджетів, додатків та інтернет-ресурсів. Нові технології мають на меті покращити життя суспільства та кожної окремої людини. Розроблюються нові технології, покращуються старі додатки, автоматизується все більше процесів. Більша частина населення щоденно користується здобутками сучасних технологій.

Бухгалтерський облік інколи стає складною операцією для людина, яка не має відповідного досвіду у роботі з доходами, звітами, податками. В таких випадках доводиться витрачати велику кількість часу і ресурсів на пошуки контори, яка буде займатися питаннями бюрократії.

Останнім часом все більше набирають популярність веб- та десктоп-додатки, CRM (Системи управління взаємовідносинами з клієнтами) / ERP (організаційна стратегія інтеграції виробництва і операцій, управління трудовими ресурсами, фінансового менеджменту і управління активами), які покликані спростити питання управління доходами та фінансами для кожного.

Odoo – система для будь-яких потреб. Додаток з відкритим кодом та ліцензією LGPL доступна для редагування та доробки будь-якій людині. Система має зручний інтерфейс, кросплатформленість та доступна на пристроях різних габаритів, що дозволяє користуватись системою будь-де та у будь-який час [1].

Компонент бухгалтерського модуля дозволяє використовувати всі можливості Odoo та надавати користувачеві максимальну автоматизацію процесів роботи з податками та доходами. Все що раніше потребувало велику кількість часу перетворюється на операцію в декілька кроків, також це не потребує жодної підготовки, за рахунок чого система стає доступним для кожного.

Також це дозволяє відмовитись від великої кількості паперів, адже архів з усіма нотатками завжди поруч. Система Odoo дозволяє зберігати необхідну інформацію у базі даний стільки часу, скільки потрібно. Відмова від паперових аналогів стає не

тільки зручним у користуванні, а також економить місце у житловому приміщенні і сприяє покращенню екологічного стану планети, адже це питання стоїть сьогодні дуже гостро.

Отже, бухгалтерський модуль ERP Odoo надає великі можливості користувачеві, а також сприяє покращенню навколишнього середовища.

Записка до дипломної роботи складається з п’яти розділів - вони надають необхідну кількість інформації для повного розуміння створеної роботи та розробці модулів в системі Odoo 13 Community Edition загалом. У першому розділі описується задачу створення модуля бухгалтерського обліку. Другий розділ описує систему Odoo 13 та її аналоги. У третьому розділі описуються засоби розробки, що були використані при розробці. Четвертий розділ складається з опису програмної реалізації, а у п’ятому опис роботи користувача з розробленим модулем.

1. Задача створення модуля бухгалтерського обліку

Модуль бухгалтерського обліку аналізує загальний стан бізнесу, дозволяє переглянути рахунки компанії, подробиці заборгованості, опис стану компанії. Програмний продукт дозволяє отримувати точну картину бізнесу завдяки звітам, які створюються в реальному часі. Це дає можливість автоматизувати повторювані дії, а також швидко вирішувати проблеми, такі як: створення звітів в податкову, аналіз зміни оборотних коштів, або таких проблем як борги клієнтів та перелік несплачених замовлень.

Користувач системи використовує модуль замість команди аналітиків та бухгалтерів, або в якості допомоги для вирішення їх задач. Модуль надає чітку та корисну інформацію за допомогою збору даних в загальній базі даних компанії, яка відправляється з клієнта на сервер та належним чином оброблюється. Інтерфейс для заповнення і відображення даних спільний, тож користувач має можливість і заповнювати і переглядати заповнені дані.

1.1 Технічні характеристики

Інтерфейс користувача має надавати можливість швидко, зручно та з будь-якого пристрою заповнювати необхідну інформацію, а також відображати її у разі необхідності. Інтерфейс також має відповідати сучасним стандартам дизайну та бути локалізованим на українську мову. Додати можливість експортувати дані з інших систем. Треба переконатись, що дизайн зрозумілий для будь-якого користувача цільової аудиторії. Також важливо, щоб дизайн не відрізнявся в різних браузерах та

на різних пристроях. Забезпечити валідацію даних, щоб уникнути потрапляння неправильного формату даних на сервер, або так званих SQL-ін’єкцій.

База даних повинна зберігати всю інформацію про користувача, інформацію довідників, документів, звіти, податковий облік та іншу інформацію, яку користувач надсилає на сервер. Інформація зберігається в архіві впродовж необмеженого терміну, кількість інформації обмежується лише дисковим простором сервера. Також запити в базу даних мають бути оптимізованими, сама база проіндексована, створені відповідні абстракції. Варто зауважити, що система надає можливість користування багатьма користувачами, через що можливі випадки перенавантаження даних. Також має бути можливість експортувати базу даних для забезпечення контрольних точок.

Модуль бухгалтерського обліку віддає інтерфейс користувачу, оброблює інформацію надіслану користувачем на сервер (створення звіту, впорядкування довідників і документів), виконує запис у базу даних, готує записи з бази даних для відображення на користувацькому інтерфейсі. Проект має монолітну архітектуру, тобто окремо винесено лише базу даних: фронт-енд та бек-енд частина застосунку знаходиться в межах одного проекту.

1.2 Користувацькі можливості

Модуль повинен мати змогу швидко актуалізувати форми бланків до чинного законодавства, для чого необхідно реалізувати декілька шаблонів бланків та додати можливість створювати власні шаблони, щоб система швидко адаптувалась під нові вимоги.

Реалізувати експорт даних у форматі .xlsx та .xml, щоб користувач міг передати необхідну інформацію на інший пристрій, або в іншу програму чи систему. Це забезпечує повну гнучкість системи.

Звіти мають бути розширені та мати всі найбільш поширені варіанти, а саме: журнали, бухгалтерські книги, головна бухгалтерська книга, прибуток, збиток, баланси рахунків, пересування коштів, банківська книга, касова книга, щоденна книга, партнерська книга, аудит журналу, активи, податковий облік. На стороні клієнта звіти створюються шляхом вказування початкової та кінцевої дати періоду формування звіту. На стороні сервера оброблюються дані з клієнта, отримуються всі необхідні дані з бази даних та виконуються необхідні розрахунки для формування звіту. Звіт має зберігатись в архіві та бути доступним необмежену кількість часу.

Звіт про прибутки і збитки має містити чистий прибуток компанії у звітний період (дата початку і кінця вибирається користувачем) за вирахуванням витрат. Також можна вибирати чи відображати всі записи, або тільки опубліковані. Звіт має бути персоналізованим: додати логотип, фон, вибрати тип макету. Звіт можна надрукувати, або завантажити у форматі pdf.

Звіт по балансу рахунків показує активи, пасиви і капітал бізнесу. Капітал розраховується через вирахування обов’язкових витрат з активів.

Пересування коштів показує зміни балансу і доходів і вплив цих змін на кошти та їх еквівалент. Тобто це потік грошей компанії.

Звіт по банківській книзі демонструє всі банківські операції компанії: включаючи початковий баланс, або без нього. Початковий баланс вказується від вибраної користувачем дати. Банківські рахунки можуть бути відсортовані по даті журналу, або партнеру. Також передбачити можливість включити до звіту відображення сум дебіту, кредиту та балансу [2].

Щоденна книга містить всі операції компанії у вибраний день.

Модуль також має надавати можливість перегляду транзакції партнерів вашої компанії (якщо баланс відкритий на перегляд). В звітах партнерів відображаються всі транзакції партнера, а також всі кредити і дебіти у вибраних журналах.

Головна книга містить всі грошові операції на всіх рахунках в обраний період, або в кожному окремому рахунку у журналі компанії.

Податковий облік включає в себе звіт з продажу та закупівлі за звітний період. В модулі має бути передбачено також суму податків за цей обраний період, базову суму податку, групування по продажам і закупкам.

Аудит має показувати всі операції з грошима вашого бізнесу, поставлених у відповідність до бухгалтерських журналів обліку. Журнали для відображення обираються користувачем, або складаються на основі інших обраних звітів за вказаний період.

Активи мають бути відображені у вигляді графіку, або зведеному аналізі.

Всі звіти мають містити функцію друку, а також імпорту або експорту даних. Звіти можуть бути заплановані та шаблонні. Вигляд звіту персоналізується під кожну компанію, також є можливість вибрати або змінити шаблон звіту. Значення податків встановлюється у конфігурації модуля.

Користувач має можливість переглядати необхідні документи, такі як виписка, рахунок, зарплата. У пункті конфігурації також налаштовуються умови виплат: наприклад строки, вказані для клієнтів.

1.3 Вимоги до конфігурації

В налаштуваннях модуля має бути представлена можливість створення та редагування інкотермів. Інкотерми – міжнародні комерційні виплати. Це спеціальні короткі абревіатури, які використовуються для визначення ризиків пов’язаних, наприклад, з транспортуванням та доставкою продукції [3].

Користувач має можливість створювати нові журнали одного з вказаних типів:

- Закупівля
- Продаж
- Готівка
- Загальні

Банківські рахунки також можуть додаватись, або видалятись. Вони надають можливість переглядати рух коштів по вказаним рахункам компанії.

Також потрібно передбачити можливість створювати та редагувати інформацію облікового запису користувача: змінювати фото, ім’я, особисту інформацію

1.4 Вхідні дані

Перед початком розробки було розгорнуто DEVOPS-система на засобах Jira, Git, Kubernets. Завдання на написання коду та контроль виконаних завдань виконується в DEVOPS системі.

Git Flow організовано таким чином, що будь-які дані, які потрапляють до репозиторію будуть запускати скрипти, що в свою чергу запустять команди зборки проекту. Після проведення зборки роботами Kubernets код потрапляє на сервер, який в свою чергу перезапускається та повідомляє розробнику про готовність внесених змін. Також реалізовано доступ до серверу шляхом підключення через VPN до локальної мережі сервера за допомогою SSH. Таким чином дані можуть редагуватись безпосередньо на сервері, але при наступній зборці проекту будуть замінені.

Засоби Jira надають зручний доступ до задач, що створює менеджер проекту, а також до необхідних документів з переліком доступів та документації. В Jira було створено дошку з задачі на базі Kanban. Задачі створювались зі станом «готові до розробки», після чого відповідальна особа, що приймає завдання на себе, має змінити стан на «виконується». Після виконання завдання потрапляє до тестування, де отримує статус «проходить тестування», у випадку виявлення проблем задача повертається до розробника з описом проблеми, яку розробник має виправити та відправити назад у тестування. Виконана задача може бути закрита [4].

5.3 Висновки до розділу

Отже, в цьому розділі було розглянуто основні пункти завдання на створення бухгалтерського модуля на базі системи Odoo 13 Community Edition, початкові дані та вимоги до функціоналу. Було описано шлях, який проходить кожна задача а також етап виконання задачі розробником.

Технічні характеристики, які мають бути виконані у модулі, вимоги до можливих налаштувань, які користувач може змінювати у системі, а також опис кінцевих можливостей користувача є достатніми для початку вивчення предметної області та розроблення архітектури проекту, після чого можна розпочати розробку бухгалтерського модуля.

2. Опис та аналоги системи Odoo 13 Community Edition для роботи з бухгалтерським модулем

2.1 Опис системи Odoo 13 Community Edition

Odoo 13 Community edition – безкоштовна веб-програма для автоматизації управління бізнесом. Odoo це відкрита платформа з широким спектром можливостей: управління виробництвом, торгівля, планування і багато іншого. Існує велика кількість різних модулів, які розширюють функціонал платформи. Модулі надають різний функціонал, такий як: CRM, налаштування сайту компанії, управління закупками, продажами, проектами, інтеграція з різними соціальними мережами [4].

Odoo має відкрите джерело коду, тобто платформа може редагуватись будь-якими розробниками, розширювати функціонал, додавати нові моделі у базу даних та швидко адаптуватись під необхідні зміни оточення.

Система має модульний тип, що забезпечує гнучкість та прозорість інтеграції. Існує багато модулів, що забезпечують велике різноманіття функціоналу. Отже кожна компанія знайде для себе модуль для власних потреб.

Завдяки правильно створеній архітектурі та хорошій кодовій базі Odoo 13 Community Edition модулі займають невелику кількість пам’яті.

Локалізовано під декілька мов та може розширюватись перекладами користувача. Файли перекладів поставляються не виконуваним кодом, а файлами офіційних перекладів інтерфейсу.

Оскільки система створена у вигляді веб-додатку, не потребує скачувань та встановлення. Для роботи користувача необхідно мати тільки браузер та стабільний інтернет. Також це забезпечує доступність для будь-якого пристрою, будь-якого розміру незалежно від операційної системи.

Серед переваг модульної системи також варто зазначити швидкодію, адже користувачу не потрібно завантажувати код, який не буде використовуватись. Модулі знаходяться в публічному маркетплейсі, легко знаходяться та швидко встановлюються. Так само просто можна видалити модуль.

Найбільшою проблемою системи Odoo 13 Community Edition визначають відсутність адаптації під ринок України. Відсутні місцеві фінанси і бухгалтерія. Також зазначають, що фунціонал бухгалтерського обліку потребує допрацювання та вдосконалення

Переваги:

- Безкоштовна
- Легко персоналізується
- Доступ з будь-яких пристроїв
- Доступно багато перекладів

Недоліки:

- Мало відома через що важко знайти компетентних розробників

2.2 Аналоги

2.2.1 1С-Бухгалтерія

«1С: Бухгалтерія» - найбільш популярна програма для роботи з бухгалтерськими модулями, яка займається автоматизацією обліку. Сам продукт і інші сервіси, які можна до нього підключити, дають будь-якій фірмі ефективні інструменти для вирішення бухгалтерських завдань. Фірма «1С» постійно займається вдосконаленням своїх продуктів та модулів, для того щоб запропонувати універсальне рішення для бухгалтерії. Ця програма - це якісній інструмент бухгалтера

або власника, який надає можливість вести бухгалтерський і податковий облік, а також створювати необхідні звіти та документи. Облік ведеться відповідно до законодавству та потреб компанії, економить час та зусилля працівників, надає інструменти для аналізу стану компанії [5].

«1С: Підприємство» це багатофункціональне рішення. Система дає можливість в комплексі вирішувати поставлені перед бухгалтером задачі. Розмір підприємства не грає ролі, а також характер виробництва.

Містить різні перевірки даних, які забезпечують більшу надійність при виготовленні звітів. Специфічні алгоритми забезпечують перевірки правильності заповнення даних.

Переваги:

- Найбільш досвідчена компанія на ринку
- Великий обсяг користувачів
- Керування знижками
- Має додаток для комп’ютера

Недоліки:

- У відповідності до чинного законодавства заборонена для використання в Україні
- Платна базова версія
- Довготривала адаптація під зміни форм податкового обліку

2.2.2 «БухСофт»

БухСофт – онлайн сервіс, який не потребує завчасного встановлення і доступний для будь-якого пристрою з доступом в інтернет. Призначена для ведення обліку як податкового, так і бухгалтерського, кадрового, оперативного, складського, тощо. Також присутні сервісі для транспортування звітів у відповідні органи.

Однією з важливих переваг визначається цілодобова технічні підтримка російською мовою, а також простота у використанні. Порівняно з попередніми програмами націлено виключно на ведення бухгалтерського обліку, тобто має набагато менший функціонал, що робить програму надійнішою та простішою у вивченні.

Переваги:

- Має онлайн версію, що дозволяє працювати з програмою без попереднього скачування
- Має безкоштовний пробний період
- Цілодобова підтримка

Недоліки:

- Платна
- Оновленнями керує розробник, що означає підтримку сервісом лише типових проектів, що не потребують налаштувань під специфічні потреби.

2.2.3 «SAP»

Одне з найсильніших підприємств на ринку. Німецька компанія вважається головним конкурентом 1С. ERP-система дозволяє автоматизувати виробництво, фінанси, управління складами і персоналом, а також бухгалтерський облік [6].

Це рішення дозволяє професійно автоматизувати діяльність компанії та надає велику кількість функціоналу, хоча рішення розробників інтерфейсу інколи викликає питання в користувачів.

Переваги:

- Надійність
- Адаптована під український ринок

Недоліки

- Дуже дорога
- Висока заробітня плата компетентних розробників

2.3 Висновки до розділу

В розділі було описано систему Odoo 13 Community Edition та наведено обгрунтування вибору системи для розробки бухгалтерського модуля. В порівнянні з конкурентами система є гнучкою, піддається персоналізації та адаптуванню під необхідні вимоги користувача, також система є дозволеною для використання в межах України.

Безкоштовна версія має менше функціоналу, проте за рахунок створення власного модуля набір можливостей може бути реалізований згідно з технічним завданням.

Система може використовуватись на різних пристроях, які мають доступ до інтернету, а також зберігати багато додаткової інформації, що робить вибір на користь Odoo 13 Community Edition оптимальним.

3. Засоби розробки

3.1 Вибір технологій та їх обґрунтування

Серверна частина модуля реалізовано мовою програмування Python. Графічний інтерфейс створено за допомогою вбудованих засобів Odoo та мовою розмітки xml, а також розширено мовою програмування JavaScript, мовою розмітки HTML і каскадною таблицею стилів CSS. HTML, CSS та JavaScript наразі є єдиними засобами, що виконуються та розпізнаються браузером, тому цей вибір є безальтернативним для будь-якого веб-додатку. Також система Odoo використовує популярний фреймворк Vue.js для реалізації інтерфейсу користувача.

Система Odoo надає користувачу готові стилі та шаблони інтерфейсів користувача. Але за потребою це можна змінювати або доповнювати персоналізованим кодом CSS. Додатковий код стилів дозволяє змінювати існуючий дизайн, а завдяки розмітки HTML на сторінку можна додавати нові блоки, або навіть створювати нові сторінки. Все це дозволяє як перероблювати під власні потреби існуючі можливості Odoo, так і додавати нові.

Оскільки модулі і сама система Odoo розроблена мовою програмування Python, тому вибір був продиктований розробниками системи, адже створювати інтеграцію з іншою мовою програмування зайняло б багато часу і ми вважаємо, що це недоцільно, оскільки Python ідеально підходить для розробки бухгалтерського модуля. Ця мова є простою для розуміння, має багато пакетів для роботи з математичним аналізом. Для розробки було обрано версію 3.6, оскільки це найбільш стабільна версія при роботі з даною платформою.

Для перекладів було використано бібліотеку i18n, яка надає зручні можливості для визначення перекладів слів. Дана бібліотека є безкоштовною і присутня в

переліку пакетів Python. Переклади працюють шляхом створення словника – асоціативного масиву (ключ-значення), де ключем є слово англійською мовою, а переклад вказується будь-якою необхідною мовою. Було обрано ручний переклад, оскільки автоматичний не відзначається високою точністю перекладів довгих речень, а також такі бібліотеки погано працюють з лексичним контекстом [7].

Бібліотеки XML-RPC Library та JSON-RPC Library використовуються для взаємодії з сервером та передачі даних. Ці механізми дозволяють легко і надійно викликати процедури і функції, присутні на сервері. Дані концепції не мають великих можливостей, за рахунок чого стають простішими у використанні і тестуванні.

В системі Odoo використовується СУБД PostgreSQL, тому для розробки модуля була обрана саме ця СУБД. Оскільки ця PostgreSQL є реляційною СУБД, досягається краща цілісність даних та структура бази даних. Головна відмінність реляційних баз даних полягає в наявності чітко структурованих таблиць, які в свою чергу мають стовпці і рядки. В кожній таблиці знаходиться інформація про об’єкт, представлений в предметній області. Доступ до будь-якого значення може бути отриманий шляхом запиту, реорганізовувати базу даних при цьому не потрібно. Рядок в таблиці може мати унікальний ідентифікатор – первинний ключ, а також таблиці можуть бути пов’язані між собою за допомогою зовнішніх ключів.

3.2 Мова програмування Python

Python це об’єктно-орієнтовна, інтерпретована мова програмування високого рівня. Мова розроблена голландським розробником Гвідо Ван Россумом. Python має динамічну семантику, простий синтаксис та багато різних пакетів, що робить мову дуже популярною для швидкої розробки додатків та робить код елегантним та простим у читанні [8].

Висока продуктивність та відсутність компілятору робить з мови дуже зручний засіб для написання сценаріїв на стороні сервера.

Мова розроблена таким чином, що його дуже легко розуміти і читати. Сам по собі власний синтаксис мови досить скромний, більшість додаткових можливостей досягається шляхом підключення додаткових модулів-пакетів. Завдяки цьому розробник може більше сконцентруватись на питанні вирішення проблеми, ніж на самому інструментові, адже зазвичай написання коду це лише 20% від розробки, більшість часу має витрачатись на пошук оптимального рішення та побудови алгоритму. Python чудово надає ці можливості.

Однією з найважливіших переваг мови програмування Python є кросплатформеність, адже мова однаково виконається на всіх операційних системах, що потенційно позбавляє великої кількості проблем, так як згідно технічного завдання необхідно створити рішення, яке б запускалось на будь-якому пристрої.

Python не надає високої швидкодії програмі, адже мова не є компільованою, для рішень, які потребують пришвидшення, наприклад, створення операційної системи, або драйверів для пристрою, слід звернути увагу на менш високорівневу мову, так як C, C++. Для нашого завдання швидкодія не є пріоритетом, тому робимо висновок, що недоліки Python не є критичними для даного завдання.

Також мова має багато прихильників та велику кількість розробників та користувачів, завдяки чому можна знайти готові рішення або поради щодо розв’язання задачі в інтернеті. Мова підтримується та часто оновлюється, що забезпечує впевненість у тому, що у майбутньому не буде проблем з обслуговуванням програмних продуктів.

3.3 Мова програмування JavaScript

JavaScript – мова сценаріїв, що застосовується для забезпечення динаміки веб-додатків та впровадження дій користувача, передачі даних на сервер. Також за допомогою JavaScript програмуються всі дії, що виконуються без перезавантаження сторінки.

Мова є об’єктно-орієнтованою, динамічно типізованою, підтримує всі стилі програмування. Стандартом мови є EGMAScript, найбільш розповсюдженою для розробки є стандарт EGMAScript -6, але не всі браузера ще підтримують цей стандарт, через що використовуються транспілятори, наприклад Babel, які транспілюють ES-6 код в більш старий стандарт. Оскільки в системі Odoo 13 Community Edition відсутні засоби для збору проектів, таких як Webpack, Gulp, Grunt… транспілятори не були підключені в загальну зборку, також не створювався загальний файл скриптів. Замість цього було використано архітектуру розбивання файлів для кожної сторінки і використано стандарт EGMAScript-5, який підтримується у всіх сучасних браузерах. Такий підхід дозволяє не перейматись за коректну роботу програми у різних версіях браузерів [9].

JavaScript це динамічна мова програмування, як і Python, тому можна стверджувати, що в проекті відсутня сувора типізація змінних.

JavaScript використовується для розробки веб-додатків, як клієнтської так і серверної частини додатку, мобільних додатків, комп’ютерних додатків. Але найбільше розповсюдження отримав для реалізації клієнта веб-сайтів і веб-додатків, оскільки це єдина мова що підтримуються всіма популярними браузерами.

Без використання JavaScript веб-рішення мали б лише мінімальну динаміку, були б повністю відсутні математичні розрахунки на стороні клієнта. HTML і CSS інколи називають мовами програмування, проте це помилкова думка, адже вони не привносять майже ніякої динаміки до програм.

В сучасних підходах розробки за допомогою JavaScript клієнт створюється повністю, без використання HTML і CSS, замість цього використовується новий формат JSX, який в ході транспілювання коду перетворюється в розмітку сторінки. В проекті використовується JavaScript-фреймворк Vue.js, який надає можливості для реалізації сучасного підходу веб-додатку. За цим підходом більшість коду і логіки для генерування сторінки відбувається на комп'ютері користувача, що дозволяє зменшити навантаження на сервер, тим самим зменшивши ймовірність несанкціонованого припинення виконання програмного коду.

3.4 Мова розмітки сторінки HTML

Hyper Text Markupe Language – мова гіпертекстової розмітки, єдиний формат, який читається браузерами, через що використовується для формування «скелету» будь-яких веб-рішень. Завдяки HTML визначаються будова виводу тексту і його формат [10].

За допомогою широкого набору елементів (тегів) дозволяє виводити на екран будь-яку інформацію, як текст, так і медіа-файли, графіки. В браузері розмітка HTML формує Document Object Model (деревоподібну структуру з елементами веб-сторінки), з якою може взаємодіяти як JavaScript, так і каскадна таблиця стилів CSS.

В проекті HTML використовується для розширення існуючих компонентів користувацького інтерфейсу, які не були передбачені в Odoo 13 Community Edition.

Сучасна версія HTML5 надає велику кількість API для взаємодії з браузером, таким чином можна створювати нескладні графіки, використовувати стандартні поля для зручного та надійного вводу інформації користувачем на будь-якому пристрої, виконувати аудіо- та відео-медіа.

Також в Odoo 13 Community Edition присутні функції CMS, завдяки чому користувач може створити власний сайт з автоматично згенерованим HTML-кодом.

Правильно написаний HTML код, який проходить валідацію, піднімає сайт в результатах пошуку в пошукових системах.

3.5 Каскадна таблиця стилів CSS

Завдяки CSS можна створити стилі для веб-сторінки, тобто визначати зовнішній вигляд HTML елементів за допомогою таблиці стилів, які у браузері трансформуються у CSSOM (деревоподібна ієрархічна структура), що в свою чергу поєднується з DOM у браузері та відображається з визначеним виглядом на сторінці користувача.

На сьогоднішній день існує багато готових рішень для CSS: фреймворки та UI Kit, які значно спрощують роботу зі стилями, а в деяких випадках і повністю замінюють CSS-код. Дуже зручним та поширеними на сьогодні є препроцесори, такі як Scss, SASS, Less. В проекті за відсутності необхідних компіляторів сучасні рішення розробки не були використині, проте більшість стилів вже завчасно створені розробниками системи Odoo 13 Community Edition, тож додаткові засоби розробки могли б зробити проект невиправдано важким, тому застосування вбудованих в браузер засобів було найкращою ідеєю.

Існує декілька форматів мови, на сьогодні найбільш популярний CSS3, який є стандартом, що визначається специфікацією W3C, як і HTML. CSS3 надає багато нових технологій для роботи з таблицею стилів, найбільш поширені серед нових підходів є FlexBox, переважно який і був задіяний для створення дизайну веб-додатка [11].

3.6 Мова програмування SQL

SQL це мова структурованих запитів. Це декларативна мова програмування для здійснення запитів до реляційної бази даних. Використовується для побудови предметної області у вигляді таблиць, модифікації баз даних, оновлення її структури та даних. Оскільки SQL це мова для запитів, вона потребує системи керування базами даних.

SQL хоч і не мають широкого набору функцій, та красномовства, як інші мови, проте вважається мовою програмування, оскільки надає набір правил для комп'ютера, які той має виконати. Мова не компілюється, тобто це мова сценаріїв, яка інтерпретується і безпосередньо виконується пропускаючи етап компіляції.

Сучасні засоби мови SQL дозволяють використовувати багато різних команд, наприклад створення збережених процедур, тригерів, проводити індексування бази даних для оптимізації пошуку даних по реляційним базам даних – все це робить мову неймовірно популярною серед розробників. Основними командами завжди залишаються 3: Select, update, delete – вибрати, оновити, видалити. Також доступні команди для об'єднання таблиць, об'єднання результатів запитів та шифрування даних, що забезпечує надійну передачу даних. Запити виконуються за допомогою звернення до унікальних ідентифікаторів таблиць та зовнішніх ключів [12].

В проекті було обрано саме цю мову, оскільки вона перевірена часом, а сувора структура реляційних баз даних забезпечує сталу архітектуру та при правильному підході надає надійну обробку транзакцій та цілісніть даних, що є пріоритетом при роботі з бухгалтерським модулем. Також SQL добре себе проявляє в роботі з великою кількістю запитів до бази даних.

3.7 СУБД PostgreSQL

PostgreSQL - об'єктно-реляційна СУБД з відкритим кодом. що використовує та розширює мову SQL у поєднанні з багатьма функціями, які дозволяють зберігати та масштабувати дані будь-якої складності та навантаження.

Об’єктно-реляційна архітектура СУБД надає їй переваги перед іншими популярними рішеннями: підтримка користувацьких об’єктів, користувацькі типи даних, різноманітні операції та функції – все це робить PostgreSQL дуже гнучким у порівнянні з конкурентами рішенням. Підтримка великої кількості типів даних робить з PostgreSQL СУБД номер 1 за вибором у розробників.

PostgreSQL має перевірену досвідом архітектуру, вона надійна, зберігає цілісність даних, пропонує користувачам надійний набір функцій, постійно вдосконалюється суспільством розробників завдяки принципу відкритості до редагування, що забезпечує постійні інновації та оновлення. Завдяки цьому система використовується у багатьох проектах та з роками набирає все більшої популярності [13].

Більшість сучасних розробників обирають PostgreSQL замість MySQL, оскільки вважається що в MySQL СУБД є неправильні рішення, які закладені при проектуванні системи, тому при масштабуванні спливають певні проблеми, наприклад при роботі з репліками в storage engine.

В проекті вибір в сторону PostgreSQL було зроблено за рахунок дозволеній кількості даних для збереження, адже передбачається зберігання великих архівів даних, що може спричинити до краху бази в інших СУБД, також це надає можливість створювати багато необхідних колонок для зберігання даних, адже математичні розрахунки та нюанси роботи з бухгалтерськими звітами несуть за собою також великий об’єм таблиць у базі даних. Цього було досягнуто шляхом розбивання великих файлів на велику кількість невеликих за об’ємом.

3.8 Середа розробки PyCharm

Найбільш поширена IDE для написання коду мовою програмування Python. Серед найбільших переваг відзначається функція автодоповнення зі штучним інтелектом, яка не тільки передбачує можливі варіанти написання коду, але також одразу вказує на помилки розробника, підсвічуючи їх та пропонує можливе виправлення. Оснащена автоматичним рефакторингом (перейменування назви змінної може відбуватись одразу в усіх місцях її використання) [14].

Підтримує більшість розширень та фреймворків, дозволяє зручно редагувати не тільки Python, але й HTML, CSS, JavaScript, XML, SQL, а також бібліотеку NumPy, яка є необхідною для роботи з обчисленнями.

Має корисну функцію для створення віртуального середовища з потрібним інтерпритатором. Тобто якщо в користувача встановлена версія Python 3.9, а робота з кодом потребує більш низьку, то PyCharm дозволяє легко виходити з подібної ситуації шляхом створення персоналізованого середовища виконання з усіма необхідними параметрами та додатками, завдяки чому проект вдалось виконати на більш стабільній версії Python без глобальних проблем.

В проекті було обрано використовувати цю IDE, оскільки вона є безкоштовною та надає повну підтримку всім вибраним засобам розробки, тобто це є універсальне рішення з широким вибором функціоналу.

3.9 Інструмент управління pgAdmin 4

pgAdmin 4 - інструмент, що надає графічний інструмент для роботи з базою даних PostgreSQL, надає можливість швидкого запуску та налаштування,

персоналізації. Зручний інструмент як для початківців, так і для досвідчених розробників. Без досвіду роботи з PostgreSQL допомагає швидко зрозуміти принципи СУБД та налаштувати базу даних, запустити її для подальшого використання.

Надає можливість зручно використовувати всі можливості СУБД: створити користувачів, базу даних, таблиці, сервер. Система створена мовами програмування Python та JavaScript бібліотекою jQuery.

В офіційній документації Odoo 13 Community Edition наведено приклад для створення та початку роботи з системою шляхом використання pgAdmin 4.

3.10 Висновки до розділу

В цьому розділі було наведено всі інструменти, які використовуються при створенні модуля на базі системи Odoo 13 Community Edition, наведено обгрунтування кожного вибору.

Основна бізнес-логіка написано мовою програмування Python, складові користувацького інтерфейсу створені шляхом комбінування засобів HTML, CSS, JavaScript. База даних створена за допомогою СУБД PostgreSQL.

Версії кожної з використаних технологій було обрано найбільш стабільні, оскільки продукт має підтримуватись багато років. Вибір багатьох інструментів був продиктований розробниками Odoo, деякі інструменти необхідні для використання через специфіку області те середи роботи системи.

4. Опис програмної реалізації

4.1 Структура програмного забезпечення

Odoo 13 Community Edition використовує трьохрівневу архітектуру (бізнес-логіка, вигляд та база даних розділені). Зовнішній вигляд побудований за допомогою класичної комбінації HTML, CSS, JavaScript. Бізнес логіка описана винятково мовою програмування Python. Сховище підтримує лише PostgreSQL.

За визначенням документації Odoo, модуль – сукупність функцій та даних, об’єднаних для досягнення спільної мети. Клієнтські та серверні розширення упаковані як модулі, які можна додати до бази даних. Модулі можуть додавати суттєво новий функціонал, або вдосконалювати існуючий. Таким чином можна додати правила бухгалтерського обліку для певної країни, або створити функціонал карт місцевих парків в реальному часі. Вся будова Odoo заснована на концепції модулів. Модулі можуть називатись як addons, а шляхи до них розробник вказує завдяки глобальній змінній addon_path.

Сама по собі Odoo поділяється на дві версії: Community (з відкритим кодом) та Enterprise Edition (ліцензійована), які відрізняються доступним функціоналом, підтримкою та частотою оновлення. Платна версія містить більший набір модулів, або надбудов над існуючими модулями. Це стало причиною появи розробників для Odoo 13 Community Edition, які імплементують оновлення з ліцензійної версії у безкоштовну.

У документації Odoo зазначено, що першим етапом розробки нового модуля є створення нової підпапки у папці /addons, або будь-якій іншій, якщо замінити стандартний шлях до модулів, який визначається змінною addon_path. Назва нової папки має містити технічну назву модуля.

4.2 Початок роботи

Для початку роботи з Odoo 13 Community Edition необхідно мати комп’ютер з доступом в інтернет, IDE для роботи з кодом (рекомендується використовувати PyCharm, оскільки більшість робіт будуть проводитись з кодом Python).

Першим етапом для роботи є встановлення системи контролю версій Git, завдяки якій ми можемо отримати вихідний код, оскільки Community Edition має відкрите джерело. Після цього кроку, можна починати розробку та робити розгалуження гілок, щоб в майбутньому запропонувати для суспільства свій код.

Як вже зазначалось, Odoo вимагає встановленої версії Python 3.6 або вище, якщо ви цього ще не зробили – вам необхідно виконати це для подальшої роботи з системою. Після встановлення Python необхідно завантажити всі залежності та модулі, які потребує список забов’язань. Деякі бібліотеки потрібно буде встановити вручну, оскільки вони не входять до пакетів Python, наприклад, бібліотека для перетворення HTML в pdf формат. Ця бібліотека використовується для створення звітів.

Третім важливим кроком є встановлення бази даних, рекомендується робити це локально. Таким чином після встановлення локальної бази та створення користувачів, варто зазначити налаштування доступів до бази даних в конфігураційному файлі Odoo: всі необхідні початкові таблиці система створить автоматично.

Після проведення всіх необхідний етапів підготовки можна запустити систему. Розробник має побачити в консолі рядки з лог-текстом, які інформують про дії виконувані наразі на сервері. За замовчування Odoo займає на 127.0.0.1 (localhost) порт номер 8069. За цією адресою можна звернутись з будь-якого браузера, який встановлений на комп’ютері [15].

В розробці використовуються засоби перевірки формату та структури коди, тому після виконання необхідний змін обов’язково треба звернути уваги на повідомлення від лінтера.

4.3 Створення модуля

Створюючи в корені нового модуля файл __init__.py ми вказуємо які файли та залежності необхідно імпортувати для роботи нашого нового модуля. Також в корені знаходиться файл __manifest__.py, який є дуже важливим в середовищі Odoo. Цей файл містить всю необхідну інформацію про модуль, перелік його залежностей, файли які модуль має завантажити. Ці файли необхідні для загрузки модуля в список всі доступних додатків.

Файл __manifest__.py містить в собі такі поля:

- Назву модуля
- Опис модуля
- Версія модуля
- Назва категорії, до якої належить модуль
- Резюме модуля
- Інформація про автора
- Залежності
- Необхідні дані для завантаження
- Ліцензія
- Картинка для створення презентації
- Інша додаткова інформація

Детальний опис модуля можна знайти у папці /static/description/index.html

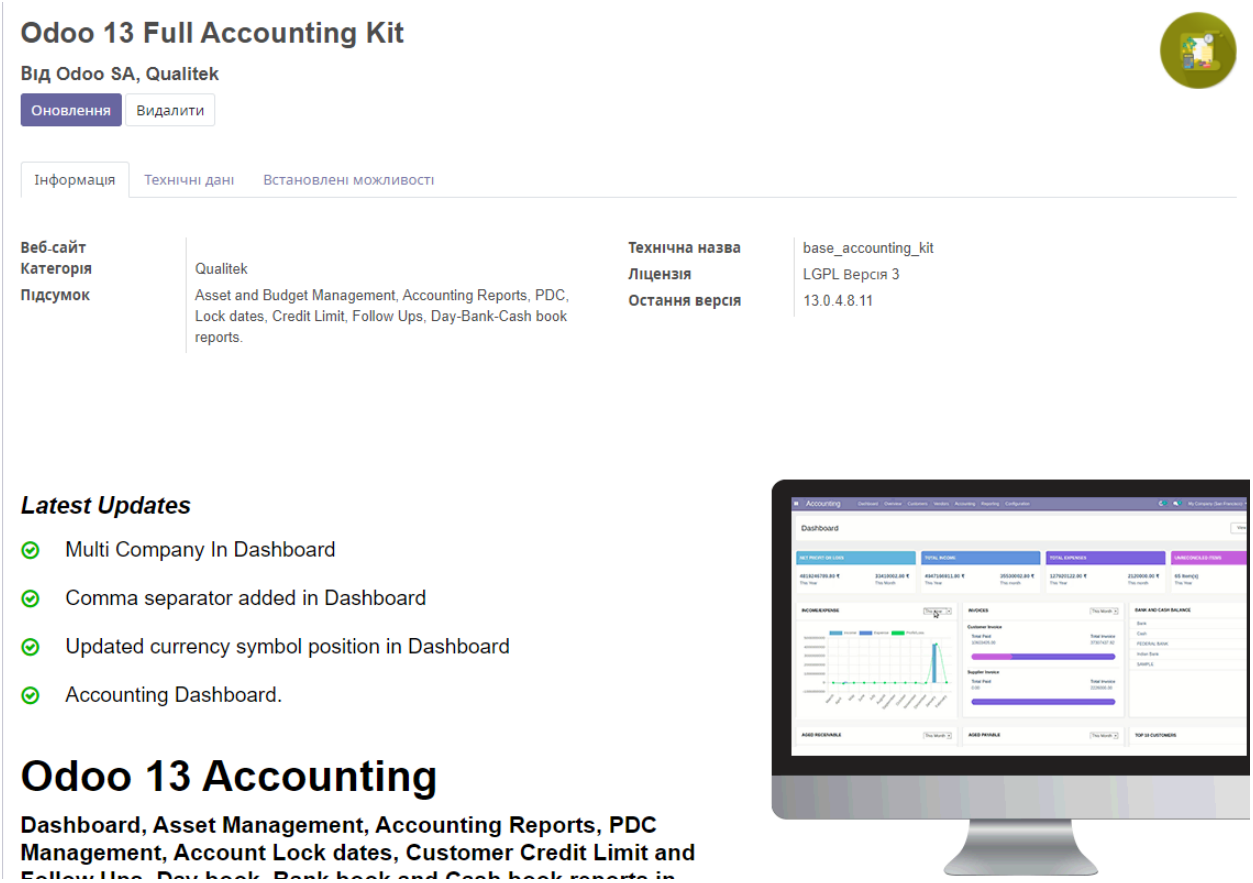


Рисунок 4.1 - Відомості про модуль в окремому index.html файлі

Структура модуля визначається стандартами розробки для Odoo 13 Community Edition і має вигляд:

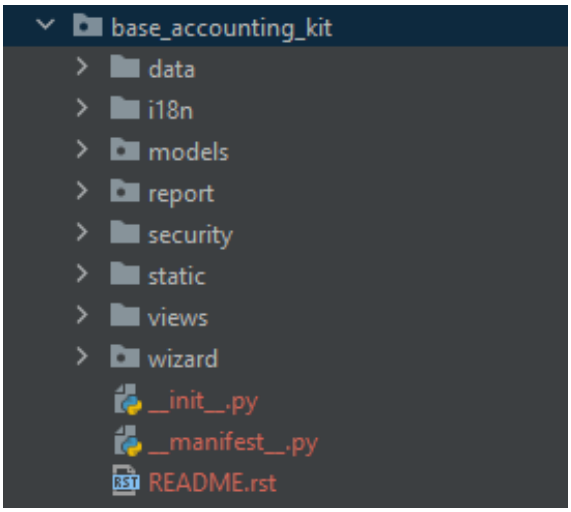


Рисунок 4.2 - Структура модуля

Папка models є необхідної і містить в собі файли з усім функціоналом, що необхідний для роботи модуля, а саме: створення таблиць в базі даних при ініціалізації модуля, оброблення інформації, що надсилається користувачем, описані функції для виконання бухгалтерських розрахунків, а також здійснюється читання та запис у базу даних.

Папка data є опціональною і містить файли формату .xml в яких перераховані дані для демонстрації роботи модуля, тобто це ті дані які відображаються при першому встановлені модуля.

i18n – папка з перекладами слів з англійської мови. Оскільки в системі Odoo не передбачено автоматичного перекладу, ця задача лягає на розробника модуля. Для кожної мови ставиться у відповідність файл з відповідною назвою, наприклад для української мови файл перекладів матиме назву uk_UA і міститиме переклади з англійської на українську у форматі ключ-значення.

Папка report була створена для зручності, вона відсутня у специфікації розробки Odoo. В цій папці зберігаються всі необхідні моделі для роботи зі звітами, які мають формуватись у відповідь на запит клієнта.

Security папка містить файли з даними, що визначають списки контролю доступів для різних користувачів у форматі .xml

Папка static призначення для зберігання всіх файлів, які використовуються для інтерфейсу користувача: зображення, відео, html, css, JavaScript файли та всі інші необхідні ресурси, які мають виконуватись або відобразитись у браузері.

У Views зберігаються файли у форматі .xml які визначають інтерфейс користувача, пов’язують інтерфейс з діями користувача у браузері: кліки, скроли, переходи по сторінкам

Папка wizard також є опціональною. В цій папці лежить функціонал, який додає підказки в інтерфейс користувача, наприклад виводить на екран текст який описує дії кнопки, коли користувач наводить на нею мишкою.

Також було створено додатковий файл README в якому перелічені основні інструкції по інсталюванню і користуванню модулем.

Розгортання модуля виконується за таким алгоритмом:

1) Запускається сервер

2) Встановлюється бухгалтерський модуль

3) Імпортуються всі необхідні файли, зазначені у файлі __init__.py, який є вхідним файлом програми.

4) Виконуються скрипти у папці models які ініціалізують створення нових таблиць у базі даних, або доповнюють існуючи таблиці

5) Сервер отримує дані з бази даних, та віддає користувачеві інтерфейс з необхідною інформацією

При зміні даних користувачем виконуються JavaScript інструкції які надсилають дані на сервер, де вони мають валідуватися, обробитися і виконати необхідну дію: зробити новий запис в базу даних, отримати дані з бази даних, оновити дані, або видалити.

4.3.1 Створення моделей

Модулю необхідно зберігати дані про свою роботу, системні дані, дані користувача та архіві в якомусь певному місці, для Odoo 13 Community Edition це місце є базою даних, а моделі стають середньою ланкою між шаблонами користувацького інтерфейсу та базою даних.

Моделі представлені в системі на рівні ORM, завдяки чому можна уникнути написання запитів в базу даних вручну, що в свою чергу збільшує безпеку програмного продукту та полегшує розробку. Замість цього всі об’єкти оголошуються як класи Python, які спадкуються від класу Models, отримуючи весь необхідний базовий функціонал для роботи за базою даних [16].

Далі моделі можуть налаштовуватись шляхом встановлення необхідних атрибутів. Єдиним обов'язковим для кожної таблиці атрибутом є атрибут _name, який встановлює ім'я моделі в системі Odoo.

За домовленістю розробки кожна модель розміщається в своєму власному файлі, а всі файли розташовані в одній папці з моделями. На рисунку 4.3 вказано приклад створеної моделі з необхідними атрибутами на рівні ORM [17].

```
class Followup(models.Model):
    _name = 'account.followup'
    _description = 'Account Follow-up'
    _rec_name = 'name'

    followup_line_ids = fields.One2many('followup.line', 'followup_id',
                                        'Follow-up', copy=True)
    company_id = fields.Many2one('res.company', 'Company',
                                 default=lambda self: self.env.company)
    name = fields.Char(related='company_id.name', readonly=True)
```

Рисунок 4.3 - Створена модель

Варто зазначити, що після проведення змін, необхідно перезапустити сервер, щоб побачити результат, оскільки редагуються файли сервера.

Використані в якості атрибутів поля файлів визначають як і де будуть використовуватись дані. Існує 2 загальні види полів: поля що зберігають дані, та поля які використовуються в якості ключів – пов'язують записи у моделях. Деякі поля створюються системою автоматично, без оголошення, наприклад:

- Id – унікальний ідентифікатор моделі
- create_date – дата створення моделі
- create_uid – ім'я користувача, який створив модель
- write_date – дата останньої модифікації
- write_uid – ім'я користувача, який останнім редагував модель

Ці поля є необхідними не тільки для роботи з Odoo, а також рекомендовані для будь-якого проекту, оскільки містять великий обсяг корисної інформації та дані, що допомагають взаємодіяти з моделями.

Для створення модуля частіше використовується не одна, а багато моделей для роботи з великою кількістю даних і користувацьких блоків. І часто виникає питання зв’язку між моделями для доступу до даних між іншими моделями.

Одна з моделей зв’язку між модулями називається ManyToOne, що перекладається з англійської як багато до одного. Ця концепція визначає, що на один екземпляр моделі може посилатись багато об’єктів іншої моделі. Ця модель визначається як метод при створення полів.

Інша модель зв’язку називається ManyToMany «багато до багатьох». Це багаторазовий двонаправлений зв’язок, який підтримує багато об’єктів ключів і багато об’єктів значень.

Остання модель називається OneToMany, яка схожа на ManyToOne, але має зворотній напрямок дії.

Деякі моделі будують зв’язок автоматично, адже фреймворк Odoo має можливість визначати неявно деякі дії та зв’язки між моделями, що робить розробку моделів під системи набагато безпечнішої і легшою.

Зв’язки між моделями є дуже важливими для будь-якого модуля Odoo для досягнення моделювання будь-яких справ. Проте інколи виникає потреба в створенні вираховуваних значень всередині моделі, інколи це буває корисним для редагування даних при користувацькому вводі даних. Зазвичай дані знаходяться безпосередньо в базі даних і отримуються шляхом запитів, без змін, на сервер. Але поля на сервері можуть змінювати в процесі отримання та модифікування даних з бази даних, викликаючи функції з моделі. Для досягнення цього результату необхідно спочатку створити необхідні функції та прив’язати їх для певних атрибутів моделі.

Методи для обчислення динамічних полів використовуються лише приватні функції. Ці методи зазвичай залежать від інших полів в моделі, тому розробнику в

приватній функції необхідно описати всі можливі залежності, щоб при зміні значення залежності виконувався перерахунок обчислювальних значень.

Оскільки функції для обчислення полів моделей є приватними, а інколи виникає потреба в безпосередній зміні цих значень, слід використовувати інверсійні функції. Такі функції змінюють залежності приватних методів та викликають цим метод обчислення поля.

Так як обчислювальні поля встановлюються під час виконання запиту в базу даних, обчислювальні значення не зберігаються в таблицях, проте через це не виконується пошук по цим даним. Для отримання таких даних можна використовувати так званий «склад», в який поміщаються проміжні значення після обчислення і як тільки обчислення добігає кінця – дані поміщаються додатковим запитом у базу даних [18].

При проектуванні моделі необхідно розробити архітектуру таким чином, щоб уникати зайвих обчислювальних полів, так як будь-які математичні розрахунки займають багато часу. Особливо у випадках коли обчислені значення у моделі можуть залежати від інших моделей, тоді виконається багато зайвих перерахунків і запитів до бази, що сильно впливає на швидкодію та продуктивність.

Необхідно зазначити, що подібний результат можна отримати шляхом використання методів змін в інтерфейсі. Проте метод змін не викликає ніяких методів у моделі, через що не завжди приводить до бажаного результату. Проте велика кількість обчислювальних значень призводить до втрати продуктивності.

Для ефективної роботи моделей існують механізми наслідування, які дозволяють значно спростити роботу з моделями та уникнути повторення коду. Також механізми наслідування дозволяють спростити архітектуру проекту.

4.3.2 Робота з користувацьким інтерфейсом

Для роботи з користувацьким інтерфейсом використовуються файли формату XML, або CSV. Формат CSV зручний у випадку, якщо дані мають простий формат, для більш складних варіантів рекомендується використовувати формат XML, який містить HTML теги. Ці файли необхідно перелічити у файлі __manifest__.py та помістити всі разом у папку views. Код з усіх файлів завантажується та виконується послідовно, тому XML та CSV файли можуть залежати та наслідувати одне від одного.

Всі стандартні дії та меню описані в базі даних Odoo 13 Community Edition, тобто всі записи сторінок та перехід між сторінками визначаються в XML файлі, найчастіше використовується схема Меню > дія > сторінка. Ці дії в системі описані за замовчуванням, тому розробнику потрібно використовувати стандартні засоби для створення навігаційної панелі.

Дія це зв’язок між користувацьким інтерфейсом та моделлю. Певні дії користувача (натискання на посилання) відправляють на сервер запит для ініціалізації зазначеної на цю дію події. Дії за замовчуванням ініціалізуються деякими способами:

- Перехід на нову сторінку
- Натискання на кнопки в користувацькому інтерфейсі
- Дії на об’єкті за контекстом

В використовуваних полях доступні такі властивості як: доступ лише для читання, значення за замовчуванням. Такі значення мають інакший вигляд, визначений стилями системи Odoo.

Стандартні засоби Odoo 13 Community Edition надають шаблони для більшості необхідний користувацьких елементів та дій з ними: списки, дерева, форми, пошук. Але стандартні блоки не покривають всіх запитів користувачів. Саме для

таких випадків є можливість додавати HTML, CSS, JavaScript і створювати нові шаблони.

Також в стандартних діях передбачені методи зміни: при зміні даних вони записуються у модель, або переключають певний стан, не вносячи жодних змін до бази даних. Будь-які зміни будуть негайно приведені і дію на стороні клієнта, але пропадуть після перезавантаження сторінки. Для того щоб нові дані потрапили до бази даних необхідно натиснути на кнопку «зберегти», таким чином користувач сам впливає на те, які дані будуть знаходитись у базі даних.

Також розробник може створювати і описувати власні публічні методи, викликаючи їх при певних діях користувача, провокуючи використання приватних методів всередині моделі. Цього можна досягти шляхом додавання унікальних ідентифікаторів для блоків шаблону та прив’язки дії до кожного з ідентифікатора.

4.4 Опис бази даних

Базу даних створено засобами мови запитів SQL, СУБД PostgreSQL у середовищі pgAdmin, яке надає зручний інтерфейс для роботи з PostgreSQL. Всього база даних Odoo 13 налічує 382 таблиці, через що наводити опис кожної таблиці ми вважаємо недоцільним. Хочеться зазначити що всі таблиці створюються самою системою Odoo. Бухгалтерський модуль лише розширює існуючі таблиці, а також створює проміжні представлення, яке не має фізичної матеріалізації, це віртуально таблиця, що з’єдную одну чи декілька таблиць.

В Odoo представлені 2 основних види даних: основні дані та демо. Основні дані визначають технічні та ділові поля модуля, зазвичай вони необхідні для коректної роботи модуля та завантажуються при першому встановленні модуля, також до основних даних належать дані, що необхідні для використання логіки: валюта, одиниці виміру, звіти, податки, схеми. Другим типом даних визначається демо-дані,

які використовуються для проведення демонстрації роботи модуля та несуть суто візуальний характер.

Всі дані знаходиться в одній базі дані, проте розділені в коді. Основні дані завантажуються напряму з бази при першому запуску, демо-дані з’являються при створенні запитів на сервер.

Також рисунку 4.4 представлено таблицю із бази даних бухгалтерського модуля Odoo 13 Community edtition. Наведена таблиця є класичним прикладом таблиці для модуля. Там представлено дані, що створюються автоматично системою, а також користувацькі дані.

Name	Comment
id	
target_move	Target Moves
view_format	Format
enable_filter	Enable Comparison
account_report_id	Account Reports
date_from	Start Date
date_to	End Date
debit_credit	Display Debit/Credit Columns
company_id	Company
create_uid	Created by
create_date	Created on
write_uid	Last Updated by
write_date	Last Updated on

Рисунок 4.4 - Таблиця financial_report «Фінансовий звіт» як приклад типової таблиці в системі Odoo

Name	Comment
id	
name	
depreciation_date	
date	
gross_value	
depreciation_value	
installment_value	
posted_value	
unposted_value	
asset_id	
move_check	
asset_category_id	
partner_id	
state	
installment_nbr	
depreciation_nbr	
company_id	

Рисунок 4.5 - Представлення asset_asset_report

На рисунку 4.5 зображено представлення asset_asset_report (шаблон звітів), яке використовується найбільш часто у програмі. Представлення зберігає в собі такі дані, як

- Унікальний ключ
- Назву звіту
- Дату звіту
- Дату коли звіт буде переміщено в архів (рахується автоматично)
- Ідентифікатор і дані про компанію
- Ідентифікатор і дані про партнерів
- Значення полів з розрахунками

За допомогою такого шаблону можна створити майже будь-який бухгалтерський звіт, якщо якоїсь інформації недостатньо, вона може буде розширена в тілі запиту, або в іншій таблиці чи представлені.

4.5 Засоби забезпечення конфіденційності

На сьогодні дуже гостро стоять питання забезпечення безпеки веб-додатка, оскільки все більше документів переходять в електронний формат. До таких документів також належать податкові звіти, фінансові звіти. В бухгалтерському модулі системи необхідно дуже ретельно вивчити область забезпечення конфіденційної інформації.

При роботи з моделями необхідно визначити певні групи користувачів, які матимуть змогу отримувати, редагувати та видаляти інформацію з бази даних. Odoo забезпечує необхідні механізми для налаштування цих процесів.

Також слід забезпечити безпечне завантаження та передачу даних. В Odoo 13 ці процеси відбуваються здебільшого завдяки .CSV файлів. За домовленістю всі файли, що мають забезпечитись найбільшим захистом розміщають в спеціальний каталог security, а інші дані знаходяться разом з іншими моделями. Двонаправлений доступ між файлами з цих двох каталогів має бути визначним у __manifest__.py файлі, який знаходиться в корені проекту.

Якщо в моделі не визначено хто має доступ до даних, то ця модель за замовчуванням вважається закритою для всіх на редагування і видалення. Доступ до читання має буть-який авторизований користувач. Користувач у режимі розробника матиме змогу безпосередньо передивлятись дані у базі.

Доступи визначаються у спеціальній моделі з переліком можливих доступів та користувачів. Визначеному користувачеві надають для виконання наступні дії: редагування, читання, додавання, видалення.

Також в Odoo є можливість об’єднувати користувачів у групи та надавати необхідні доступи одразу для декількох користувачів, що об’єднані в групу. Найкращою стратегією вважається створення ролі менеджера, якому доступний весь функціонал та користувачів, яким необхідні доступи надає менеджер. Кожен окремий

користувач може мати різні права під час роботи з системою. Насправді кожна група є лише частиною в записі правил безпеки системи.

За визначенням права доступу це спосіб надати користувачам доступ до моделей шляхом отримання доступу з групи, оскільки кожен користувач має належати до якоїсь групи користувачів. Права можуть бути надані як до цілої моделі, так і до окремих полів моделі. Права доступу надають можливість надавати або відхиляти запити на дію користувача. Так, наприклад, якщо користувач без необхідного права на доступ намагається редагувати таблицю, то на рівні моделі цей запит буде відхилено.

З кодової бази систему прав можна обійти шляхом виконання SQL-запитів, що дозволяє обійти ORM і відповідно встановлені в ORM правила, або за допомогою так названого “sudo-mode”, в якому вказується команда sudo.

Права доступу надаються на рівні ORM, таким чином створюючи проблему безпеки шляхом SQL-ін’єкцій, через що необхідно дуже ретельно валідувати дані на front-end частині системи [19].

Подібні ситуації також можуть бути перехвачені шляхом проведення явних перевірок безпеки: перевірка поточного користувача, перевірка чи має користувач поточну групу, права поточної групи, а також шляхом виклику функцій для перевірки доступів до моделі даного користувача.

В Odoo 13 Community Edition також передбачено випадок коли користувач може мати багато компній, відповідно необхідно мати доступи до моделей кожного з підприємства, що потенційно може викликати проблеми у системі безпеки.

В зазначених випадках застосовується більш жорстка система надання прав. Жодна з компаній не знає одна про одну та про доступи користувача до них. Таким чином ми отримуємо систему з 3х незалежних компаній, кожна з який надає суворі правила доступів для користувача. У випадку втрати доступу до одного з підприємств, відновити його через інше підприємство буде неможливо.

В Odoo 13 Community Edition передбачено 2 основні механізму безпеки: обмеження видимості та обмеження прав. Обмеження видимості просто не відбражає

дані для користувача, хоча фактично він може мати безпосередній доступ до моделі. У випадку обмеження прав користувач взагалі не має доступів до моделі та її полів. У діях на сервері код можуть бачати та оновлювати лише користувачі системи, а в XML визначаються користувачі або групи користувачів, яким не буде відображатись та чи інша інформація. Тобто деякий функціонал може бути доступний тільки менеджеру проекту.

4.6 Діаграма прецедентів системи

В Odoo 13 Визначені 2 типи користувачів: звичайний користувач, та користувач з адмін-правами, визначений як супер-користувач. Основною відмінністю у правах є можливість редагування даних повз валідацію на клієнті та сервері, адже для супер-користувача відкривається доступ для бази даних. Такі можливості несуть за собою також і велику відповідальність, адже неправильні дані можуть порушити роботу системи в цілому. Супер-користувач може обходити ORM та створювати запити до бази даних на пряму, тобто обходити будь-які обмеження. Проте не рекомендується застосовувати повні права, оскільки таким чином можна суттєво нашкодити модулю [20].

В інших діях користувач і супер-користувач рівні в своїх правах, оскільки для обох типів передбачено доступ до режиму розробника, який відкриває такі функції як:

- Перегляд бази даних
- Підказки з назвою полів у базі та назвою полів в інтерфейсі
- Доступ до переглядів результатів тестів
- Відключення або обмеження користувацьких дій

Додаткові параметри можуть бути визначені у налаштуваннях захищеності модуля: можна створювати нових користувачів та назначати їм певні права та доступи до пунктів меню модуля.

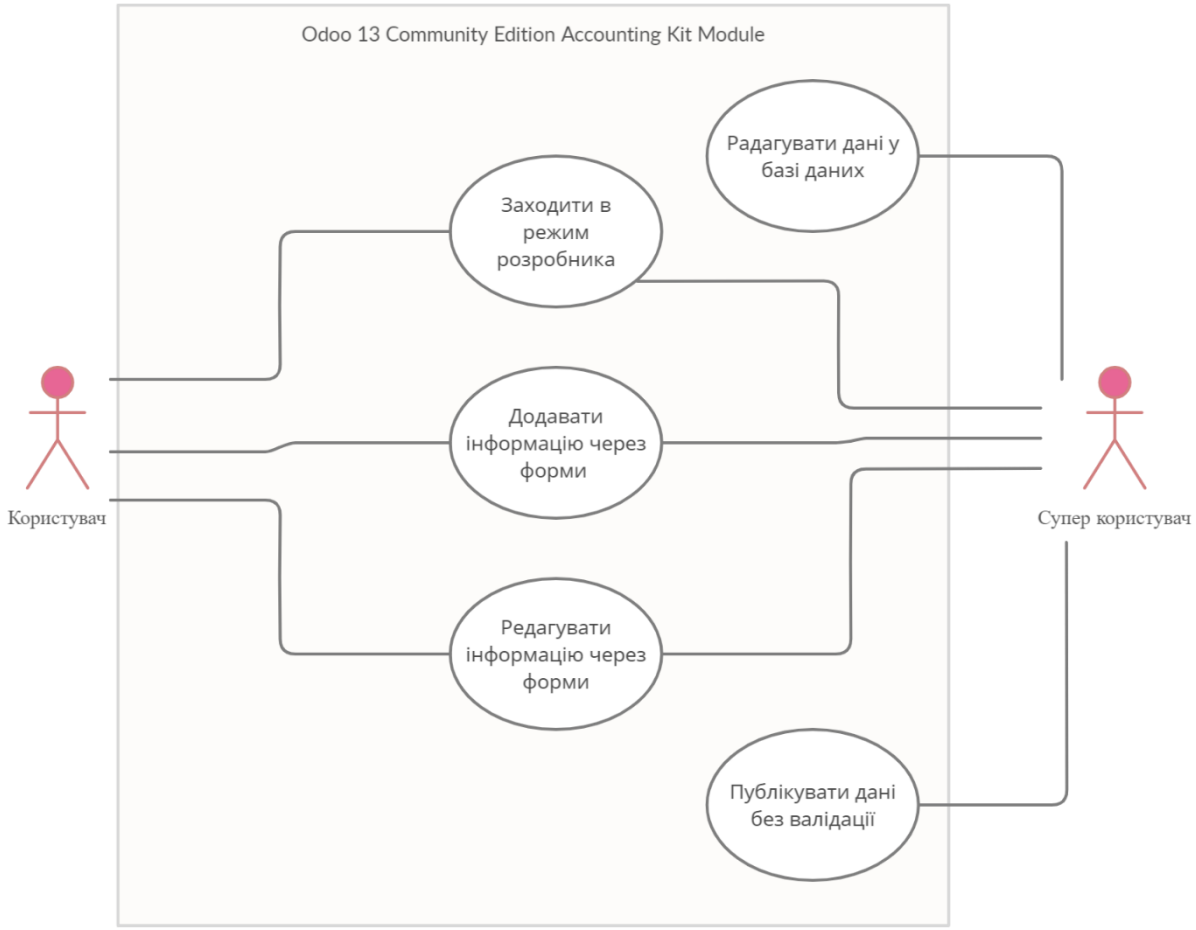


Рисунок 4.6 - Діаграма прецидентів користувачів Odoo.

4.6 Висновки до розділу

В цьому розділі було розглянуто всі необхідні кроки для початку розробки модуля на базі системи Odoo 13 Community Edition. Також було описано структуру проекту та архітектуру каталогів і файлів.

Було детально розглянуто роботу з моделями, шаблонами та базою даних. Наведено обгрунтування вибору того чи іншого підходу до розробки.

Система, окрім описаних у технічному завданні потреб до функціоналу, має задовольняти сучасні потреби користувача: швидкодія, сучасний дизайн, безпека. В розділі було наведено найкращі та найгірші практики при написання коду під систему Odoo 13 Community Edition [21].

Особливої уваги приділено питанням безпеки, оскільки в базі модуля необхідно зберігати та передавати велику кількість конфіденційної інформації.

Під час написання розділу було створено алгоритм створення нового фунціоналу та обгрунтування кроків для досягнення результатів, створено модель прецедентів користувачів. Ці кроки є необхідними для створення якісного продукту.

5 Робота користувача з програмною системою

5.1 Інтерфейс веб-додатку

При вході у модуль користувач потрапляє на головну сторінку, на якій представлена загальна інформація про бізнес у вигляді графіків і таблиць. Ця сторінка відсутня у стандартному наборі шаблонів Odoo 13 Community Edition, вона була створена засобами HTML, CSS, JavaScript. На сторінці передбачено не тільки необхідний вивід інформації, а також динамічні дії користувача. Готову сторінку можна побачити на рисунку 5.1

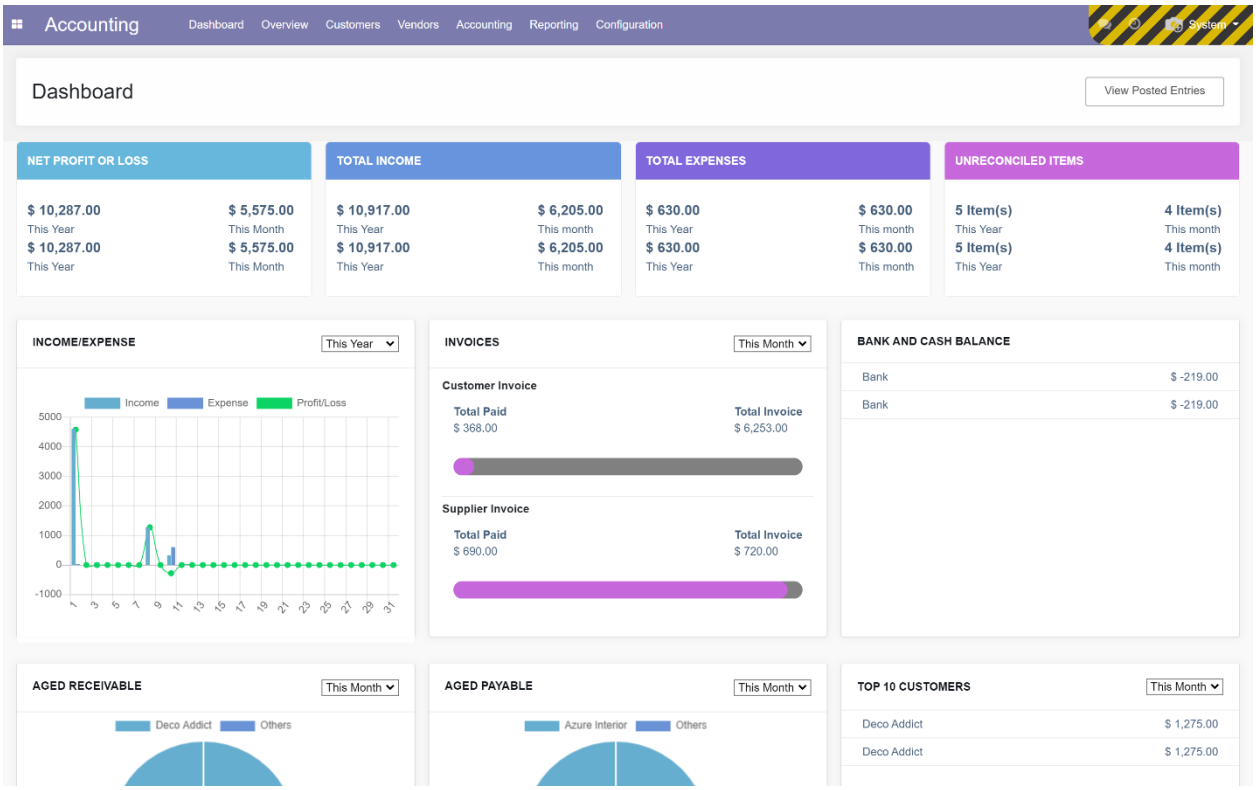


Рисунок 5.1 - Головна сторінка модуля

На рис 5.1 можна побачити наступні елементи інтерфейсу:

- Чистий дохід та збиток
- Сумарний дохід
- Сумарні витрати
- Записи, які потребують вирішення
- Дохід та збиток (графік)
- Фактури
- Баланси

У кожному полі з графіком користувач може вибрати період для відображення даних:

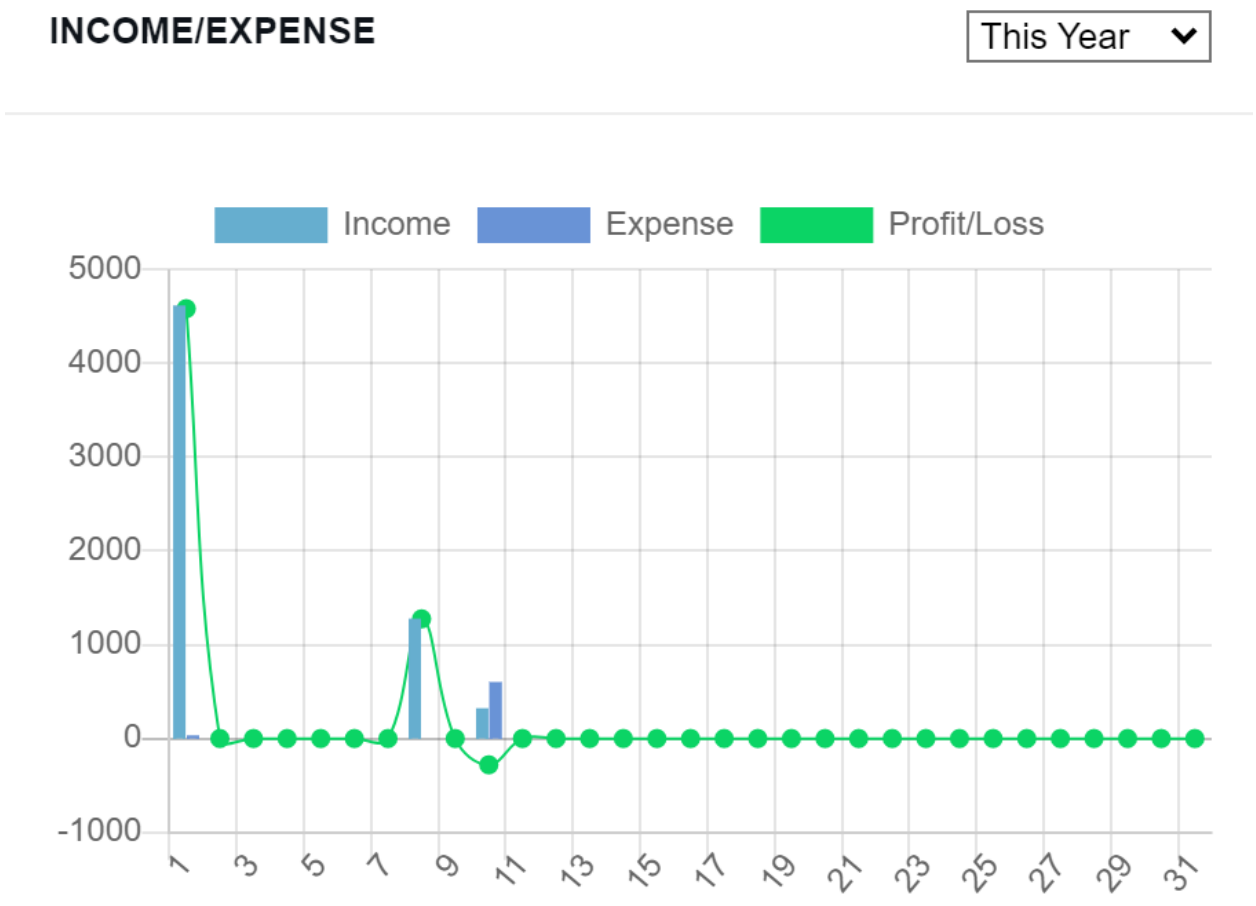


Рисунок 5.2 - Графік з можливістю вибору періоду вибірки даних

На рисунку 5.2 зображено графік доходів та збитків, зображений у вигляді стовпчикової діаграми. В цьому блоці можна інтерактивно наводити мишею на будь-яку дату та більш детально передивлятись необхідні значення. У збитки також включені податки, тому кожен прибуток включає в себе і збиток також

Згори на кожній сторінці знаходиться панель навігації по веб-додатку, яка містить посилання на всі існуючи сторінки. Згідно технічного завдання науково-дослідницької роботи потрібно було розробити такі пункти:

- Податковий облік

За допомогою меню переходимо до сторінки Tax Report та бачимо поп-ап вікно, яке пропонує користувачу ввести дати, для яких необхідно сформувати податковий звіт

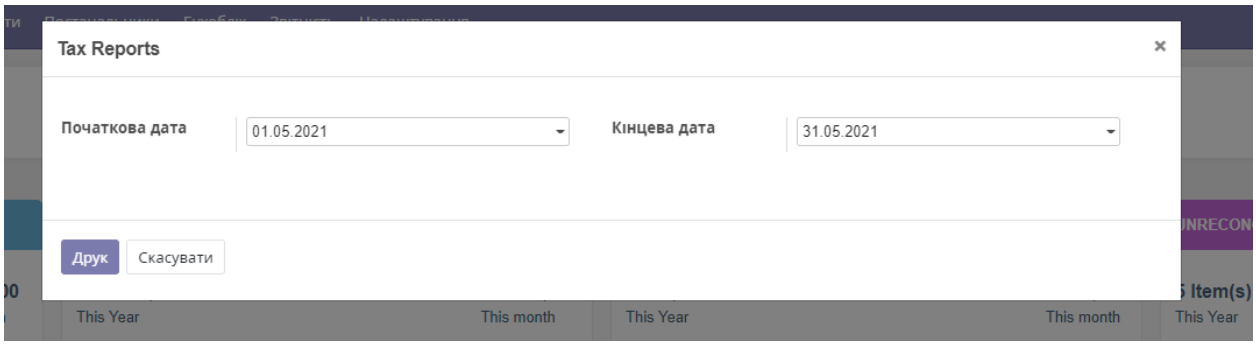


Рисунок 5.3 - Поп-ап вікно податкового звіту з вибраними даними на травень 2021 року

На рисунку 5.3 зображено поп-ап вікно, яке з’являється коли користувач натискає на кнопку “Створити звіт”, в цьому вікні є 2 поля з вибором опцій та динамічним календарем, в якому користувач може обрати необхідний період для створення звіту.

Після натискання на кнопку “Друк” користувач потрапляє на сторінку з сформованим звітом

Рисунок 5.4 - Податковий звіт з переліком доходів та податків

На рисунку 5.4 показано сформований податковий звіт, який можна роздрукувати та подавати у необхідні служби, або зберігати для власної статистики.

- Звіт прибутків та доходів

Переходимо до пунтку меню Profit and Loss, з'являється нове поп-ап вікно яке пропонує визначити період для формування звіту, а також вибрати чи потрібно включати всі записи, чи лише публічні

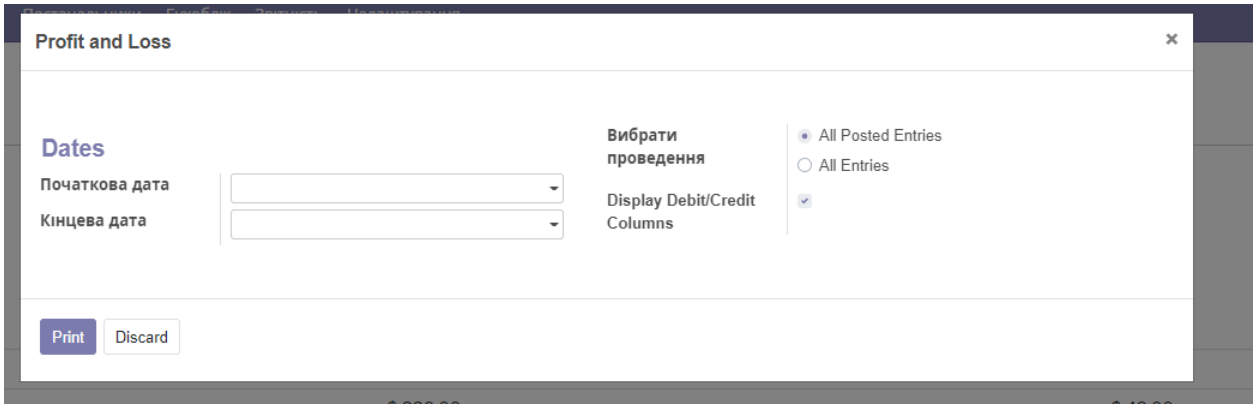
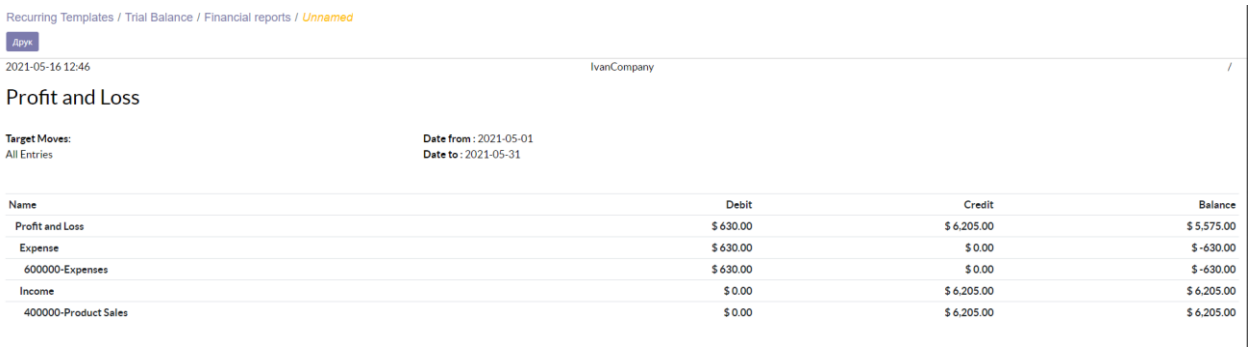


Рисунок 5.5 - Поп-ап вікно звіту пробутків та доходів

На рисунку 5.5 продемонстровано процес заповнення даними поп-ап форми для створення звіту з вибором типу записів, що мають бути включені до звіту.

Після заповнення форми необхідною інформацією користувач потрапляє на сторінку з сформованим звітом, де відображена у формі таблиці вся необхідна інформація


Recurring Templates / Trial Balance / Financial reports / Unnamed

Друк

2021-05-16 12:46 IvanCompany /

Profit and Loss

Target Moves: All Entries

Date from : 2021-05-01
Date to : 2021-05-31

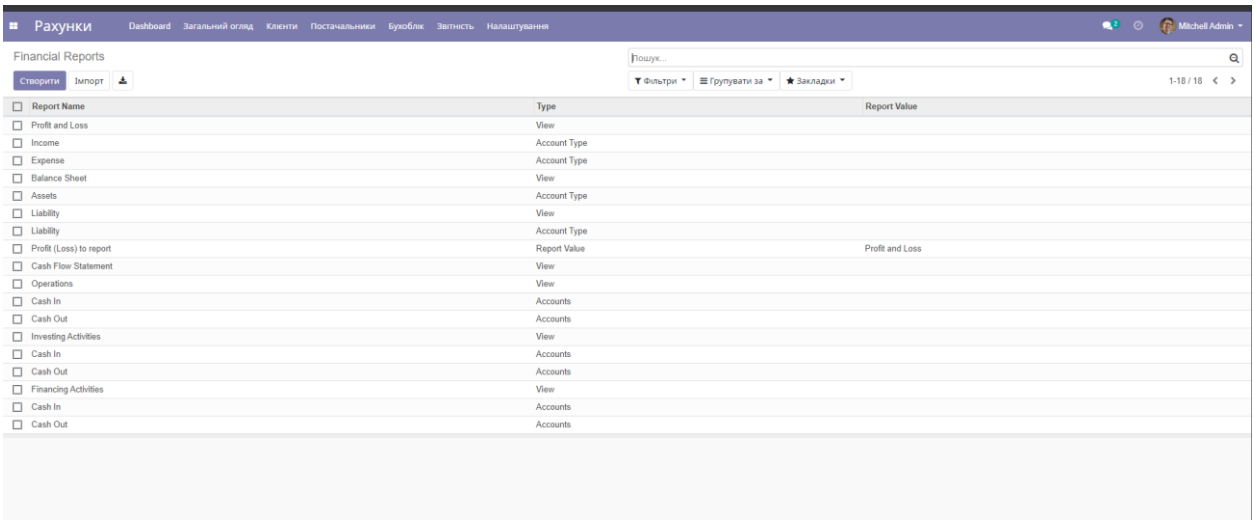
Name	Debit	Credit	Balance
Profit and Loss	$ 630.00	$ 6,205.00	$ 5,575.00
Expense	$ 630.00	$ 0.00	$ -630.00
600000-Expenses	$ 630.00	$ 0.00	$ -630.00
Income	$ 0.00	$ 6,205.00	$ 6,205.00
400000-Product Sales	$ 0.00	$ 6,205.00	$ 6,205.00

Рисунок 5.6 - Звіт прибутків та доходів

На рисунку 5.6 зображено готовий звіт по прибутками та збитками. В звіті у формі таблиці показані всі дані за обраний період та вибрані записи у журналі. Також вказано з яких потоків (дебітових чи кредитових) були задіяні гроші в обороті, а також баланс після проведення операцій. Згори звіту зазначено його назву, дату створення, тип вибраних даних і період, за яких було проведено аудит.

- Імпорт даних

Також згідно технічного завдання необхідно було додати можливість імпортувати дані, для формування звіту. Так, наприклад, на сторінці де зберігаються звіти, можна додавати старі звіти, що було зроблені в іншій програмі, або іншим підприємством, яке перейшло під ваше керівництво


Рахунки Dashboard Загальний огляд Клієнти Постачальники Бухоблік Звітність Налаштування Mitchell Admin

Financial Reports

Створити Імпорт

Пошук...

Фільтри Групувати за Закладки 1-18 / 18

Report Name	Type	Report Value
Profit and Loss	View	
Income	Account Type	
Expense	Account Type	
Balance Sheet	View	
Assets	Account Type	
Liability	View	
Liability	Account Type	
Profit (Loss) to report	Report Value	Profit and Loss
Cash Flow Statement	View	
Operations	View	
Cash In	Accounts	
Cash Out	Accounts	
Investing Activities	View	
Cash In	Accounts	
Cash Out	Accounts	
Financing Activities	View	
Cash In	Accounts	
Cash Out	Accounts	

Рисунок 5.7 - Список всіх сформованих звітів і кнопка «імпорт» для імпортування даних у форматі .xlsx

На рисунку 5.7 можна побачити архів з усіма звітами, відсортований за датою по спаданню від нових до старих записів, також зазначено назву звіту та його тип. Всі звіти, незалежно від дати створення, можна експортувати та імпортувати у форматах .xlsx або .csv, що стає в нагоді у випадку необхідності міграції даних.

У програмі також передбачено можливість перегляду рахунків/чеків після проведення транзакцій

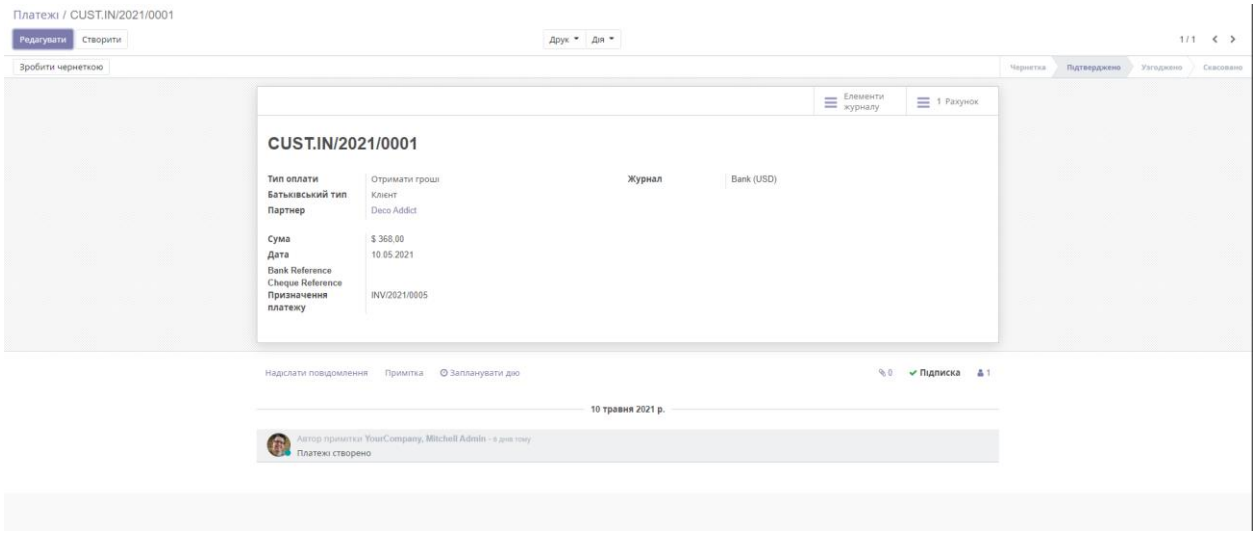


Рисунок 5.7 - Перегляд платежів

На рисунку 5.7 зображено рахунок, який було створено після процедури завершення транзакції. На кінцевому рахунку можна побачити детальну інформацію про проведений платіж.

Важливою складовою, а також однією з найбільших мотивацій для створення модулю є створення шаблонів. Згідно пункту меню Reccuring Templates користувач потрапляє на сторінку шаблонів, де може створити новий шаблон платежів чи звітів, і також редагувати існуючі. Таким чином користувач економить багато часу та зменшує ймовірність забути створити звіт, або передати його у відповідний орган

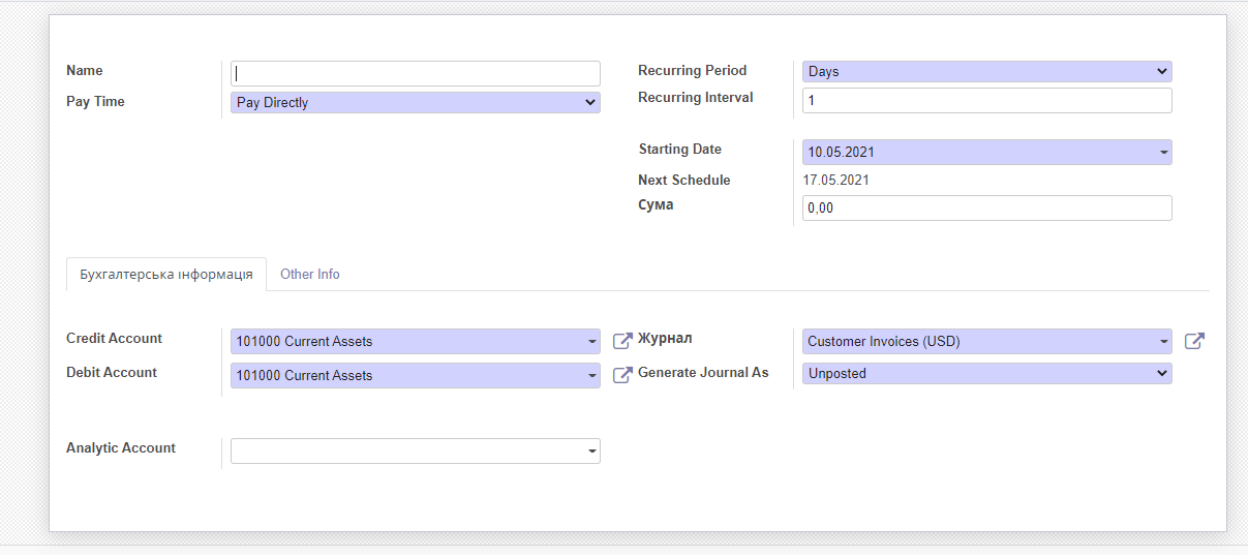
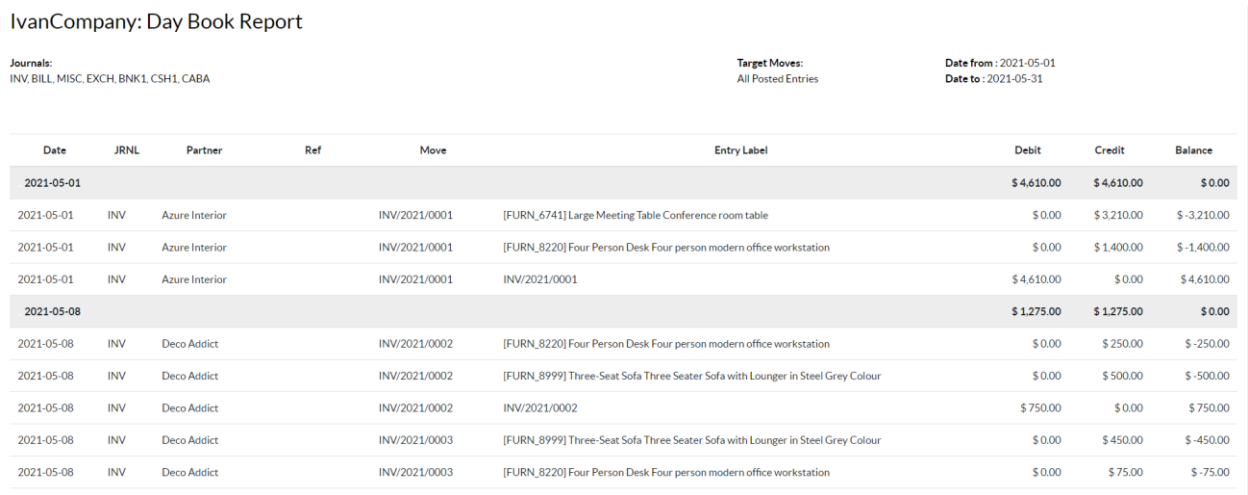


Рисунок 5.8 - Створення нового шаблону для сплати

Актуалізація інформації проводиться шляхом зміни налаштувань: термін оплати, податки, схема оподаткування, що й зображено на рисунку

Також доступні наступні звіти: баланс, рух коштів, банківська книга, книга коштів, щоденна книга, звіти партнерів, загальний звіт компанії, аудит журналу



IvanCompany: Day Book Report

Journals: INV, BILL, MISC, EXCH, BNK1, CSH1, CABA

Target Moves: All Posted Entries

Date from : 2021-05-01
Date to : 2021-05-31

Date	JRNL	Partner	Ref	Move	Entry Label	Debit	Credit	Balance
2021-05-01						**$ 4,610.00**	**$ 4,610.00**	**$ 0.00**
2021-05-01	INV	Azure Interior		INV/2021/0001	[FURN_6741] Large Meeting Table Conference room table	$ 0.00	$ 3,210.00	$ -3,210.00
2021-05-01	INV	Azure Interior		INV/2021/0001	[FURN_8220] Four Person Desk Four person modern office workstation	$ 0.00	$ 1,400.00	$ -1,400.00
2021-05-01	INV	Azure Interior		INV/2021/0001	INV/2021/0001	$ 4,610.00	$ 0.00	$ 4,610.00
2021-05-08						**$ 1,275.00**	**$ 1,275.00**	**$ 0.00**
2021-05-08	INV	Deco Addict		INV/2021/0002	[FURN_8220] Four Person Desk Four person modern office workstation	$ 0.00	$ 250.00	$ -250.00
2021-05-08	INV	Deco Addict		INV/2021/0002	[FURN_8999] Three-Seat Sofa Three Seater Sofa with Lounger in Steel Grey Colour	$ 0.00	$ 500.00	$ -500.00
2021-05-08	INV	Deco Addict		INV/2021/0002	INV/2021/0002	$ 750.00	$ 0.00	$ 750.00
2021-05-08	INV	Deco Addict		INV/2021/0003	[FURN_8999] Three-Seat Sofa Three Seater Sofa with Lounger in Steel Grey Colour	$ 0.00	$ 450.00	$ -450.00
2021-05-08	INV	Deco Addict		INV/2021/0003	[FURN_8220] Four Person Desk Four person modern office workstation	$ 0.00	$ 75.00	$ -75.00

Рисунок 5.9 - Щоденний звіт компанії

Також є інструменти для роботи з постачальниками бізнесу, на рисунках 5.10 – 5.13 зображені сторінки для роботи з постачальниками

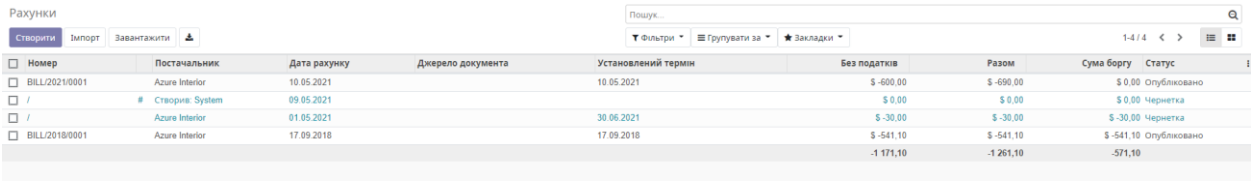


Рисунок 5.10 - Рахунки постачальників

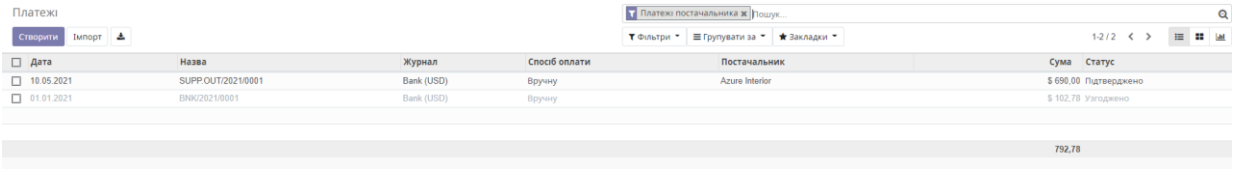


Рисунок 5.11 - Платежі постачальників

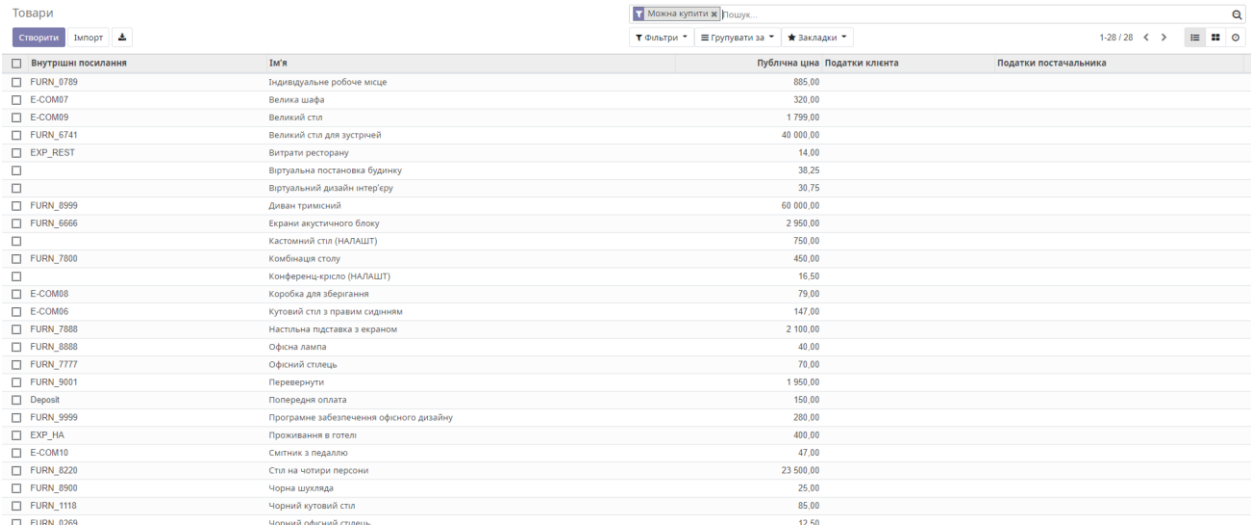


Рисунок 5.12 - Перелік товарів постачальників

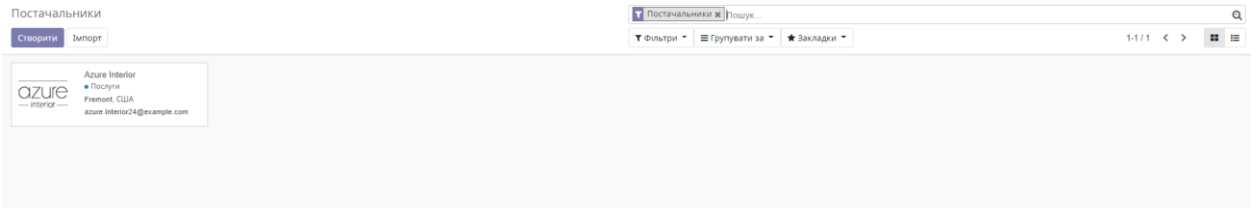


Рисунок 5.13 - Перелік постачальників

Для роботи з клієнтами використовуються аналогічні інтерфейси, різниця з точки зору системи полягає тільки в таблицях, з яких береться необхідна інформація. Така структура дозволяє добре масштабуватись та використовувати менше пам’яті

сервера. На скріншотах наведених вище зображені таблиці, в яких перелічені всі необхідні поля для виведення інформації.

Для клієнтів було додано розширення Follow-up Reports – це спеціальні нагадування клієнту, що він має сплатити рахунки не пізніше ніж зазначене число, якщо він цього ще не виконав

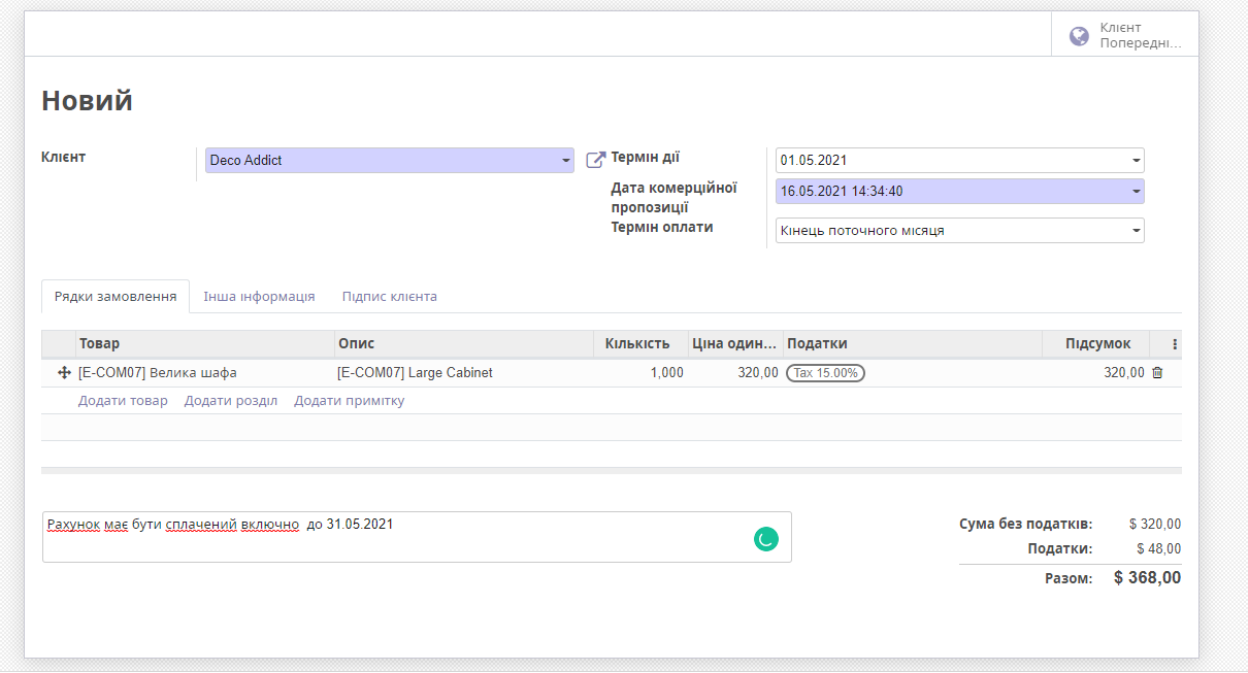


Рисунок 5.14 - Оформлення нового замовлення

На рисунку 5.14 зображено скріншот з процесу створення нового замовлення. У формі представлені такі поля: вибір клієнта, термін дії замовлення, максимальний термін сплати рахунків. Також є можливість додати товар (ціну вказувати не треба, вона буде отримана з бази даних), податки будуть автоматично додані до вартості та занесені у журнал. Знизу сторінки можна побачити суму без податків та загальну суму до сплати. Після відправки нового замовлення клієнт отримує замовлення, зображене на скріншоті 5.15. Клієнт отримує суму замовлення, інформацію, детально вказану ціну з податками, коментар від продавця, термін сплати. Клієнт має можливість в один клік оплатити замовлення, завантажити або роздрукувати замовлення. Як тільки замовлення оплачується клієнтом, то продавець отримує звіт про сплачений рахунок, який зображено на рисунку 5.16.

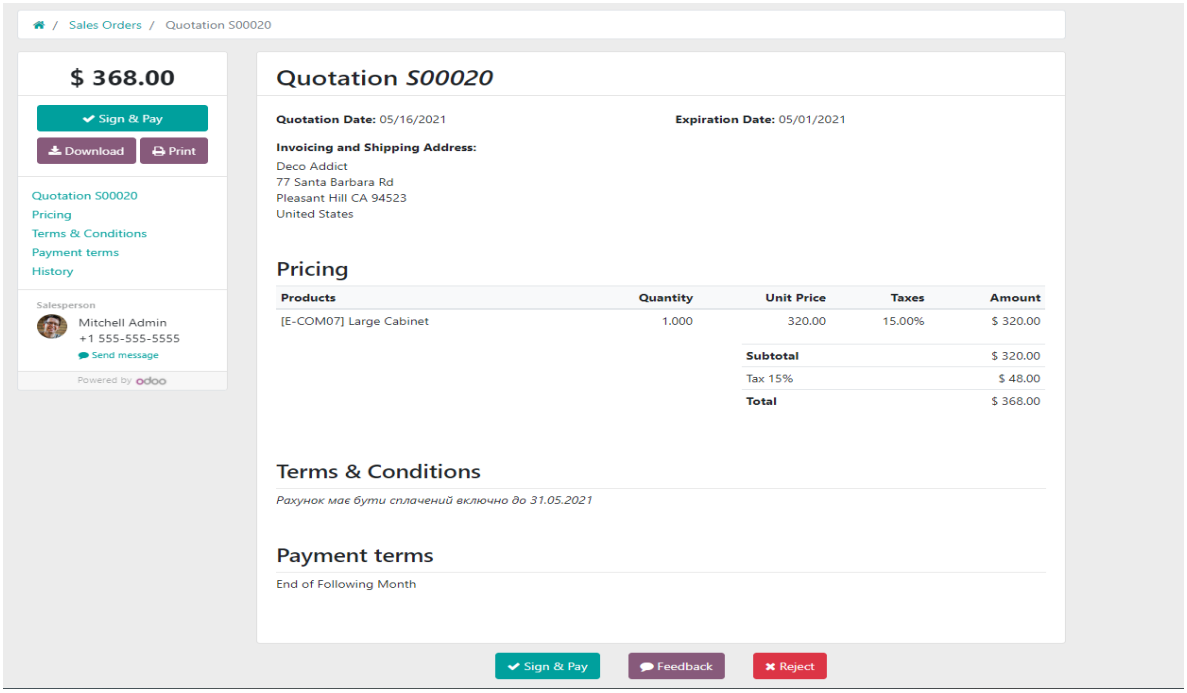


Рисунок 5.15 - Сформоване замовлення

Сплачений рахунок має штамп “Виплачено“, а також заплачену суму, а також суму боргу, якщо така наявна, оскільки клієнт має право сплатити рахунок частинами, або сплатити після зазначеного терміну кінця оплати.

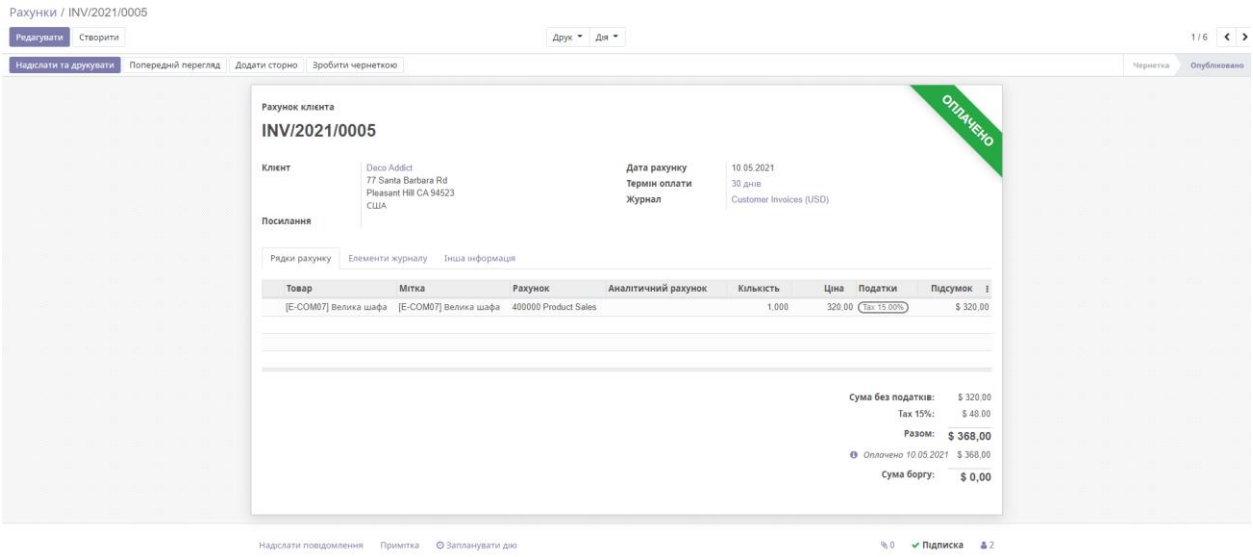


Рисунок 5.16 - Сплачений рахунок

5.2 Висновки до розділу

В останньому розділі було наведено приклади основних функцій, які надаються користувачу згідно технічного завдання. В ході візуальної презентації описані кроки для коректної роботи користувача з системою.

З метою полегшення адаптації користувача до нової системи було надано велику кількість скріншотів із зображенням описаних дій.

Отже, було створено детальну карту користувацьких дій та доступно описано набір функціональних можливостей модуля, створеного на базі системи Odoo 13 Community Edition, завдяки цьому покроковому керівництву користувач зможе без зайвих зусиль використовувати створений модуль.

Висновки

Під час проведення цієї роботи було проаналізовано задачу створення модулю автоматизації бухгалтерії, проаналізовано проблему галузі, проведено вивчення конкурентів для пошуку оптимального рішення

Проведено аналіз інструментів та засобів створення програм, обрано найкращий варіант для даної специфічної розробки, а саме: Python 3.6, JavaScript, HTML, CSS. Для роботи з базою даних застосовувались такі засоби, як мова структурованих запитів SQL, СУБД PostgreSQL, pgAdmin. Розробка велась у спеціалізованій IDE PyCharm.

В ході розробки програмного продукту були редаговані та деталізовані існуючі таблиці бази даних, створено засоби взаємодії між користувачем і сервером у межах модуля, розширено стандартні засоби виводу інформації та додано новий функціонал до системи Odoo у межах створеного модуля бухгалтерського обліку.

Отже, завдяки створенню модулю бухгалтерського обліку на базі ERP Odoo 13 Community Edition ми виконали технічне завдання науково-дослідницької роботи, а також вирішили необхідні питання, а саме:

- Створення звітів доходів, коштів, балансів, рахунків;
- Формування бухгалтерського обліку;
- Вирішено проблему частої зміни форми оподаткування;
- Експорт даних з інших подібних систем;
- Створення документів та довідників;

Завдяки чому сприяли створенню аналітики фінансів всередині компанії, зменшили витрати людино-годин на створення звітів, зменшили ризики помилки у складанні звітності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Brunn H. Odoo Development Cookbook / H. Brunn, A. Fayolle, D. Reis. — Birmingham: Packt Publishing Ltd, 2016. — 377 с.
2. Лень В. Бухгалтерський облік в Україні. Основи та практика. Навчальний посібник / В. Лень, В. Гливенко., 2018. – 608 с. – (3).
3. Правила Інкотермс, 2020. – 392 с.
4. B. Doar M. Practical JIRA Administration: Using JIRA Effectively: Beyond the Documentation 1st Edition / Matthew B. Doar., 2011. – 144 с. – (1).
5. Радченко М. 1С:Программирование для начинающих Детям и родителям, менеджерам и руководителям Разработка в системе "1С:Предприятие 8.3" / Максим Радченко. – Москва: ООО "1С-Паблишинг", 2016. – 780 с.
6. Carvalho A. Discover SAP: An Introduction to SAP, Beginner's Guide / A. Carvalho, V. Krishnamoorthy., 2014. – 536 с. – (3). – (SAP).
7. Перевод и локализация: опыт разработки профессионального стандарта / Убоженко, Александрова, Берендяев та ін.]. – Красноярск: Сибирский федеральный университет, 2019.
8. Лутц, М. Программирование на Python. Т. 1 / М. Лутц. - М.: Символ, 2016. - 992 с.
9. Резиг Джон , Бибо Беэр Секреты JavaScript ниндзя; Вильямс - М., 2015. - 416 с.
10. Роббинс Д. HTML 5. Карманный справочник / Дженнифер Роббинс., 2020. – 192 с. – (5).
11. Макфарланд Д. БОЛЬШАЯ КНИГА CSS3 / Дэвид Макфарланд., 2016. – 608 с. – (Бестселлеры O'Reilly).
12. Beaulieu A. Learning SQL: Master SQL Fundamentals / Alan Beaulieu., 2009. – 338 с. – (2nd edition).
13. O. Obe R. PostgreSQL: Up and Running: A Practical Guide to the Advanced Open Source Database / Regina O. Obe., 2017. – 314 с.

14. JetBrains [Електронний ресурс] - – Режим доступу до ресурсу: https://www.jetbrains.com/
15. Moss G. Working with Odoo / Greg Moss. – Birmingham: Packt Publishing Limited, 2015. – 432 с.
16. Beazley D. Python Cookbook / D. Beazley, B. K. Jones., 2013. – 706 с.
17. Bauer C. Hibernate in Action / C. Bauer, G. King., 2004. – 408 с. – (In Action).
18. Moss G. Learn Odoo : A beginner's guide to designing, configuring, and customizing business applications with Odoo / Greg Moss. – Birmingham: Packt Publishing Limited, 2019. – 504 с.
19. Stuttard D. The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws / D. Stuttard, M. Pinto., 2011. – 912 с. – (2nd edition).
20. Odoo docs [Електронний ресурс]. – Режим доступу до ресурсу: https://www.odoo.com/documentation/14.0/
21. Odoo фінансовий аналіз [Електронний ресурс] – Режим доступу до ресурсу: https://doc.odoo.com/5.0/ru/book/3/3_8/

ДОДАТОК А

Створення компонент бухгалтерського обліку на базі ERP системи Odoo 13 Community Edition

Специфікація

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ71174_21Б

Аркушів 2

Київ — 2021

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС _ ТМ71174_20Б	Записка.docx	Текстова частина дипломної роботи
Компоненти		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС _ ТМ71174_20Б 12-1	account_account.py account_asset.py accounf_followup.py account_journal.py account_payment.py	Основні компоненти модуля

ДОДАТОК Б

Створення компонент бухгалтерського обліку на базі ERP системи Odoo 13 Community Edition

Текст програмного модулю

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ71174_21Б

Аркушів 11

Київ-2021

```
# -*- coding: utf-8 -*-

import calendar
import datetime
from datetime import datetime

from dateutil.relativedelta import relativedelta

from odoo import models, api
from odoo.http import request


class DashBoard(models.Model):
    _inherit = 'account.move'

    # function to getting expenses

    # function to getting income of this year

    @api.model
    def get_income_this_year(self, *post):

        company_id = self.get_current_company_value()

        month_list = []
        for i in range(11, -1, -1):
            l_month = datetime.now() - relativedelta(months=i)
            text = format(l_month, '%B')
            month_list.append(text)

        states_arg = ""
        if post != ('posted',):
            states_arg = """ parent_state in ('posted', 'draft')"""
        else:
            states_arg = """ parent_state = 'posted'"""

        self._cr.execute(('''select sum(debit)-sum(credit) as income
,to_char(account_move_line.date, 'Month')  as month ,
                             internal_group from account_move_line ,account_account
where
                             account_move_line.account_id=account_account.id AND
internal_group = 'income'
                             AND to_char(DATE(NOW()), 'YY') =
to_char(account_move_line.date, 'YY')
                             AND account_move_line.company_id in ''' +
str(tuple(company_id)) + '''
                             AND %s
                             group by internal_group,month
                         ''') % (states_arg))
        record = self. cr.dictfetchall()

        self._cr.execute(('''select sum(debit)-sum(credit) as expense
,to_char(account_move_line.date, 'Month')  as month ,
                            internal_group from account_move_line ,account_account
where
                            account_move_line.account_id=account_account.id AND
internal_group = 'expense'
                            AND to_char(DATE(NOW()), 'YY') =
to_char(account_move_line.date, 'YY')
                            AND account_move_line.company_id in ''' +
str(tuple(company_id)) + '''
                            AND %s
                            group by internal_group,month
```

```
                        ''') % (states_arg))

        result = self._cr.dictfetchall()
        records = []
        for month in month_list:
            last_month_inc = list(filter(lambda m: m['month'].strip() == month,
record))
            last_month_exp = list(filter(lambda m: m['month'].strip() == month,
result))
            if not last_month_inc and not last_month_exp:
                records.append({
                    'month': month,
                    'income': 0.0,
                    'expense': 0.0,
                    'profit': 0.0,
                })
            elif (not last_month_inc) and last_month_exp:
                last_month_exp[0].update({
                    'income': 0.0,
                    'expense': -1 * last_month_exp[0]['expense'] if
last_month_exp[0]['expense'] < 1 else
                    last_month_exp[0]['expense']
                })
                last_month_exp[0].update({
                    'profit': last_month_exp[0]['income'] -
last_month_exp[0]['expense']
                })
                records.append(last_month_exp[0])
            elif (not last_month_exp) and last_month_inc:
                last_month_inc[0].update({
                    'expense': 0.0,
                    'income': -1 * last_month_inc[0]['income'] if
last_month_inc[0]['income'] < 1 else
                    last_month_inc[0]['income']
                })
                last_month_inc[0].update({
                    'profit': last_month_inc[0]['income'] -
last_month_inc[0]['expense']
                })
                records.append(last_month_inc[0])
            else:
                last_month_inc[0].update({
                    'income': -1 * last_month_inc[0]['income'] if
last_month_inc[0]['income'] < 1 else
                    last_month_inc[0]['income'],
                    'expense': -1 * last_month_exp[0]['expense'] if
last_month_exp[0]['expense'] < 1 else
                    last_month_exp[0]['expense']
                })
                last month inc[0].update({
                    'profit': last_month_inc[0]['income'] -
last_month_inc[0]['expense']
                })
                records.append(last_month_inc[0])
        income = []
        expense = []
        month = []
        profit = []
        for rec in records:
            income.append(rec['income'])
            expense.append(rec['expense'])
            month.append(rec['month'])
            profit.append(rec['profit'])
```

```
        return {
            'income': income,
            'expense': expense,
            'month': month,
            'profit': profit,
        }

    # function to getting income of last year

    @api.model
    def get_income_last_year(self, *post):

        company_id = self.get_current_company_value()

        month_list = []
        for i in range(11, -1, -1):
            l_month = datetime.now() - relativedelta(months=i)
            text = format(l_month, '%B')
            month_list.append(text)

        states_arg = ""
        if post != ('posted',):
            states_arg = """ parent_state in ('posted', 'draft')"""
        else:
            states_arg = """ parent_state = 'posted'"""

        self._cr.execute(('''select sum(debit)-sum(credit) as income
,to_char(account_move_line.date, 'Month')  as month ,
                            internal_group from account_move_line ,account_account
                            where account_move_line.account_id=account_account.id AND
internal_group = 'income'
                            AND Extract(year FROM account_move_line.date) =
Extract(year FROM DATE(NOW())) -1
                            AND account_move_line.company_id in ''' +
str(tuple(company_id)) + '''
                            AND %s
                            group by internal_group,month
                  ''') % (states_arg))
        record = self._cr.dictfetchall()

        self._cr.execute(('''select sum(debit)-sum(credit) as expense
,to_char(account_move_line.date, 'Month')  as month ,
                            internal_group from account_move_line , account_account
where
                            account_move_line.account_id=account_account.id AND
internal_group = 'expense'
                            AND Extract(year FROM account_move_line.date) =
Extract(year FROM DATE(NOW())) -1
                            AND account_move_line.company_id in ''' +
str(tuple(company id)) + '''
                            AND %s
                            group by internal_group,month
                        ''') % (states_arg))

        result = self._cr.dictfetchall()
        records = []
        for month in month_list:
            last_month_inc = list(filter(lambda m: m['month'].strip() == month,
record))
            last_month_exp = list(filter(lambda m: m['month'].strip() == month,
result))
            if not last_month_inc and not last_month_exp:
                records.append({
```

```
                    'month': month,
                    'income': 0.0,
                    'expense': 0.0,
                    'profit': 0.0,
                })
            elif (not last_month_inc) and last_month_exp:
                last_month_exp[0].update({
                    'income': 0.0,
                    'expense': -1 * last_month_exp[0]['expense'] if
last_month_exp[0]['expense'] < 1 else
                    last_month_exp[0]['expense']
                })
                last_month_exp[0].update({
                    'profit': last_month_exp[0]['income'] -
last_month_exp[0]['expense']
                })
                records.append(last_month_exp[0])
            elif (not last_month_exp) and last_month_inc:
                last_month_inc[0].update({
                    'expense': 0.0,
                    'income': -1 * last_month_inc[0]['income'] if
last_month_inc[0]['income'] < 1 else
                    last_month_inc[0]['income']
                })
                last_month_inc[0].update({
                    'profit': last_month_inc[0]['income'] -
last_month_inc[0]['expense']
                })
                records.append(last_month_inc[0])
            else:
                last_month_inc[0].update({
                    'income': -1 * last_month_inc[0]['income'] if
last_month_inc[0]['income'] < 1 else
                    last_month_inc[0]['income'],
                    'expense': -1 * last_month_exp[0]['expense'] if
last_month_exp[0]['expense'] < 1 else
                    last_month_exp[0]['expense']
                })
                last_month_inc[0].update({
                    'profit': last_month_inc[0]['income'] -
last_month_inc[0]['expense']
                })
                records.append(last_month_inc[0])
        income = []
        expense = []
        month = []
        profit = []
        for rec in records:
            income.append(rec['income'])
            expense.append(rec['expense'])
            month.append(rec['month'])
            profit.append(rec['profit'])
        return {
            'income': income,
            'expense': expense,
            'month': month,
            'profit': profit,
        }

    # function to getting income of last month

    @api.model
    def get_income_last_month(self, *post):
```

```
        company_id = self.get_current_company_value()
        day_list = []
        now = datetime.now()
        day = \
            calendar.monthrange(now.year - 1 if now.month == 1 else now.year,
                                now.month - 1 if not now.month == 1 else 12)[
                1]

        for x in range(1, day + 1):
            day_list.append(x)

        one_month_ago = (datetime.now() - relativedelta(months=1)).month

        states_arg = ""
        if post != ('posted',):
            states_arg = """ parent_state in ('posted', 'draft')"""
        else:
            states_arg = """ parent_state = 'posted'"""

        self._cr.execute(('''select sum(debit)-sum(credit) as income
,cast(to_char(account_move_line.date, 'DD')as int)
                             as date , internal_group from account_move_line ,
account_account where
                             Extract(month FROM account_move_line.date) in ''' +
str(tuple(company_id)) + '''
                             AND %s
                             AND account_move_line.company_id in ''' +
str(tuple(company_id)) + '''
                             AND account_move_line.account_id=account_account.id AND
internal_group='income'
                             group by internal_group,date
                              ''') % (states_arg))

        record = self._cr.dictfetchall()

        self._cr.execute(('''select sum(debit)-sum(credit) as expense
,cast(to_char(account_move_line.date, 'DD')as int)
                             as date ,internal_group from account_move_line
,account_account where
                             Extract(month FROM account_move_line.date) in ''' +
str(tuple(company_id)) + '''
                             AND %s
                             AND account_move_line.company_id in ''' +
str(tuple(company_id)) + '''
                             AND account_move_line.account_id=account_account.id AND
internal_group='expense'
                             group by internal_group,date
                                  ''') % (states_arg))
        result = self. cr.dictfetchall()
        records = []
        for date in day_list:
            last_month_inc = list(filter(lambda m: m['date'] == date, record))
            last_month_exp = list(filter(lambda m: m['date'] == date, result))
            if not last_month_inc and not last_month_exp:
                records.append({
                    'date': date,
                    'income': 0.0,
                    'expense': 0.0,
                    'profit': 0.0
                })
            elif (not last_month_inc) and last_month_exp:
                last_month_exp[0].update({
```

```
                    'income': 0.0,
                    'expense': -1 * last_month_exp[0]['expense'] if
last_month_exp[0]['expense'] < 1 else
                    last_month_exp[0]['expense']
                })
                last_month_exp[0].update({
                    'profit': last_month_exp[0]['income'] -
last_month_exp[0]['expense']
                })
                records.append(last month exp[0])
            elif (not last_month_exp) and last_month_inc:
                last_month_inc[0].update({
                    'expense': 0.0,
                    'income': -1 * last_month_inc[0]['income'] if
last_month_inc[0]['income'] < 1 else
                    last_month_inc[0]['income']
                })
                last_month_inc[0].update({
                    'profit': last_month_inc[0]['income'] -
last_month_inc[0]['expense']
                })
                records.append(last_month_inc[0])
            else:
                last_month_inc[0].update({
                    'income': -1 * last_month_inc[0]['income'] if
last_month_inc[0]['income'] < 1 else
                    last_month_inc[0]['income'],
                    'expense': -1 * last_month_exp[0]['expense'] if
last_month_exp[0]['expense'] < 1 else
                    last_month_exp[0]['expense']
                })
                last_month_inc[0].update({
                    'profit': last_month_inc[0]['income'] -
last_month_inc[0]['expense']
                })
                records.append(last_month_inc[0])
        income = []
        expense = []
        date = []
        profit = []
        for rec in records:
            income.append(rec['income'])
            expense.append(rec['expense'])
            date.append(rec['date'])
            profit.append(rec['profit'])
        return {
            'income': income,
            'expense': expense,
            'date': date,
            'profit': profit

        }

    # function to getting income of this month

    @api.model
    def get_income_this_month(self, *post):

        company_id = self.get_current_company_value()

        states_arg = ""
        if post != ('posted',):
            states_arg = """ parent_state in ('posted', 'draft')"""
```

```
        else:
            states_arg = """ parent_state = 'posted'"""

        day_list = []
        now = datetime.now()
        day = calendar.monthrange(now.year, now.month)[1]
        for x in range(1, day + 1):
            day_list.append(x)

        self. cr.execute(('''select sum(debit)-sum(credit) as income
,cast(to_char(account_move_line.date, 'DD')as int)
                            as date , internal_group from account_move_line ,
account_account
                            where   Extract(month FROM account_move_line.date) =
Extract(month FROM DATE(NOW()))
                            AND Extract(YEAR FROM account_move_line.date) =
Extract(YEAR FROM DATE(NOW()))
                            AND %s
                            AND account_move_line.company_id in ''' +
str(tuple(company_id)) + '''
                            AND account_move_line.account_id=account_account.id AND
internal_group='income'
                            group by internal_group,date
                        ''') % (states_arg))

        record = self._cr.dictfetchall()

        self._cr.execute(('''select sum(debit)-sum(credit) as expense
,cast(to_char(account_move_line.date, 'DD')as int)
                            as date , internal_group from account_move_line ,
account_account where
                            Extract(month FROM account_move_line.date) =
Extract(month FROM DATE(NOW()))
                            AND Extract(YEAR FROM account_move_line.date) =
Extract(YEAR FROM DATE(NOW()))
                            AND %s
                            AND account_move_line.company_id in ''' +
str(tuple(company_id)) + '''
                            AND account_move_line.account_id=account_account.id AND
internal_group='expense'
                            group by internal_group,date
                         ''') % (states_arg))
        result = self._cr.dictfetchall()
        records = []
        for date in day_list:
            last_month_inc = list(filter(lambda m: m['date'] == date, record))
            last_month_exp = list(filter(lambda m: m['date'] == date, result))
            if not last_month_inc and not last_month_exp:
                records.append({
                    'date': date,
                    'income': 0.0,
                    'expense': 0.0,
                    'profit': 0.0
                })
            elif (not last_month_inc) and last_month_exp:
                last_month_exp[0].update({
                    'income': 0.0,
                    'expense': -1 * last_month_exp[0]['expense'] if
last_month_exp[0]['expense'] < 1 else
                    last_month_exp[0]['expense']
                })
                last_month_exp[0].update({
                    'profit': last_month_exp[0]['income'] -
```

```
last_month_exp[0]['expense']
                })
                records.append(last_month_exp[0])
            elif (not last_month_exp) and last_month_inc:
                last_month_inc[0].update({
                    'expense': 0.0,
                    'income': -1 * last_month_inc[0]['income'] if
last_month_inc[0]['income'] < 1 else
                    last_month_inc[0]['income']
                })
                last_month_inc[0].update({
                    'profit': last_month_inc[0]['income'] -
last_month_inc[0]['expense']
                })
                records.append(last_month_inc[0])
            else:
                last_month_inc[0].update({
                    'income': -1 * last_month_inc[0]['income'] if
last_month_inc[0]['income'] < 1 else
                    last_month_inc[0]['income'],
                    'expense': -1 * last_month_exp[0]['expense'] if
last_month_exp[0]['expense'] < 1 else
                    last_month_exp[0]['expense']
                })
                last_month_inc[0].update({
                    'profit': last_month_inc[0]['income'] -
last_month_inc[0]['expense']
                })
                records.append(last_month_inc[0])
        income = []
        expense = []
        date = []
        profit = []
        for rec in records:
            income.append(rec['income'])
            expense.append(rec['expense'])
            date.append(rec['date'])
            profit.append(rec['profit'])
        return {
            'income': income,
            'expense': expense,
            'date': date,
            'profit': profit

        }

    # function to getting late bills

    @api.model
    def get_latebills(self, *post):

        company_id = self.get_current_company_value()

        states_arg = ""
        if post != ('posted',):
            states_arg = """ state in ('posted', 'draft')"""
        else:
            states_arg = """ state = 'posted'"""

        self._cr.execute(('''  select res_partner.name as partner,
res_partner.commercial_partner_id as res  ,
                              account_move.commercial_partner_id as parent,
sum(account_move.amount_total) as amount
```

```
                            from account_move,res_partner where
                            account_move.partner_id=res_partner.id AND
account_move.type = 'in_invoice' AND
                            invoice_payment_state = 'not_paid' AND
                              account_move.company_id in ''' + str(tuple(company_id))
+ ''' AND
                            %s
                            AND
account_move.commercial_partner_id=res_partner.commercial_partner_id
                            group by parent,partner,res
                            order by amount desc ''') % (states_arg))

        record = self._cr.dictfetchall()

        bill_partner = [item['partner'] for item in record]

        bill_amount = [item['amount'] for item in record]

        amounts = sum(bill_amount[9:])
        name = bill_partner[9:]
        results = []
        pre_partner = []

        bill_amount = bill_amount[:9]
        bill_amount.append(amounts)
        bill_partner = bill_partner[:9]
        bill_partner.append("Others")
        records = {
            'bill_partner': bill_partner,
            'bill_amount': bill_amount,
            'result': results,

        }
        return records

        # return record

    # function to getting over dues

    @api.model
    def get_overdues(self, *post):

        company_id = self.get_current_company_value()

        states_arg = ""
        if post != ('posted',):
            states_arg = """ state in ('posted', 'draft')"""
        else:
            states_arg = """ state = 'posted'"""

        self._cr.execute(('''' select res_partner.name as partner,
res_partner.commercial_partner_id as res  ,
                             account_move.commercial_partner_id as parent,
sum(account_move.amount_total) as amount
                            from account_move, account_move_line ,res_partner where
                            account_move.partner_id=res_partner.id AND
account_move.type = 'out_invoice' AND
                            invoice_payment_state = 'not_paid' AND
                            %s
                            AND   account_move.company_id in ''' +
str(tuple(company_id)) + ''' AND
                           account_move_line.account_internal_type = 'payable' AND
```

```
account_move.commercial_partner_id=res_partner.commercial_partner_id
                            group by parent,partner,res
                            order by amount desc
                            ''') % (states_arg))

        record = self._cr.dictfetchall()
        due_partner = [item['partner'] for item in record]
        due_amount = [item['amount'] for item in record]

        amounts = sum(due amount[9:])
        name = due_partner[9:]
        result = []
        pre_partner = []

        due_amount = due_amount[:9]
        due_amount.append(amounts)
        due_partner = due_partner[:9]
        due_partner.append("Others")
        records = {
            'due_partner': due_partner,
            'due_amount': due_amount,
            'result': result,

        }
        return records
```

ДОДАТОК В

Створення компонент бухгалтерського обліку на базі ERP системи Odoo 13 Community Edition

Опис програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ71174_21Б

Аркушів 10

Київ-2021

АНОТАЦІЯ

Додаток містить опис модуля бухгалтерського обліку створеного на базі ERP системи Odoo 13 Community Edition. Модуль виконує всі описані в першому розділі дипломної записки вимоги (технічні, користувацькі, конфігураційні), а саме:

- Адаптація під українську систему обліку
- Шаблони записів
- Журнал продажів та покупок
- Записи рахунків і касова книга
- Активи компанії
- Звіти по всім фінансовим операціям

Модуль розроблено мовами програмування Python, JavaScript, мовою розмітки HTML, каскадної таблиці стилів CSS в IDE PyCharm.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

У додатку міститься опис основних компонентів бухгалтерського обліку, створеного на базі ERP системи Odoo 13 Community Edition, що використовуються підприємствами для автоматизації роботи з бухгалтерським обліком, створення звітів, ведення архіву та створення статистики. Додаток Б містить програмний код цих компонентів.

Отже, завдяки створенню модулю бухгалтерського обліку на базі ERP Odoo 13 Community Edition ми виконали технічне завдання науково-дослідницької роботи, а також вирішили необхідні описані питання з першого розділу дипломної роботи.

ФУНКЦІОНАЛЬНІ ПРИЗНАЧЕННЯ

Отже, завдяки створенню модулю бухгалтерського обліку на базі ERP Odoo 13 Community Edition ми виконали технічне завдання науково-дослідницької роботи, а також вирішили необхідні питання, а саме:

- Створення звітів доходів, коштів, балансів, рахунків;
- Формування бухгалтерського обліку;
- Вирішено проблему частої зміни форми оподаткування;
- Експорт даних з інших подібних систем;
- Створення документів та довідників;

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Папка models є необхідної і містить в собі файли з усім функціоналом, що необхідний для роботи модуля, а саме: створення таблиць в базі даних при ініціалізації модуля, оброблення інформації, що надсилається користувачем, описані функції для виконання бухгалтерських розрахунків, а також здійснюється читання та запис у базу даних.

Папка data є опціональною і містить файли формату .xml в яких перераховані дані для демонстрації роботи модуля, тобто це ті дані які відображаються при першому встановлені модуля.

i18n – папка з перекладами слів з англійської мови. Оскільки в системі Odoo не передбачено автоматичного перекладу, ця задача лягає на розробника модуля. Для кожної мови ставиться у відповідність файл з відповідною назвою, наприклад для української мови файл перекладів матиме назву uk_UA і міститиме переклади з англійської на українську у форматі ключ-значення.

Папка report була створена для зручності, вона відсутня у специфікації розробки Odoo. В цій папці зберігаються всі необхідні моделі для роботи зі звітами, які мають формуватись у відповідь на запит клієнта.

Security папка містить файли з даними, що визначають списки контролю доступів для різних користувачів у форматі .xml

Папка static призначення для зберігання всіх файлів, які використовуються для інтерфейсу користувача: зображення, відео, html, css, JavaScript файли та всі інші необхідні ресурси, які мають виконуватись або відобразитись у браузері.

У Views зберігаються файли у форматі .xml які визначають інтерфейс користувача, пов’язують інтерфейс з діями користувача у браузері: кліки, скроли, переходи по сторінкам.

СИСТЕМНІ ВИМОГИ

Для роботи з бухгалтерським модулем на базі ERP Odoo 13 Community Edition необхідно мати будь-який пристрій, на який можливо встановити браузер. Також обов’язково потрібно мати доступ до мережі інтернет, оскільки система Odoo є веб-додатком та має обмінюватись даними з сервером.

Найкращий варіант використовувати пристрій з великою кількістю оперативної пам’яті через те, що в додатку виконуються JavaScript скрипти.

ВСТАНОВЛЕННЯ

Для початку роботи з бухгалтерським модулем на базі ERP Odoo 13 Community Edition необхідно завантажити та встановити на свій пристрій Odoo 13 Community Edition, перейти до пункту меню “Додатки”, знайти в списку додатків base_accounting_kit та натиснути кнопку “Інсталювати”.

Після першого встановлення завантажаться всі необхідні для роботи модуля залежності, до бази даних будуть додані необхідні таблиці. Відкриється сторінка з коротким описом модуля та презентацією основних функцій.

Посилання на головну сторінку модуля з’явиться на панелі навігації і користувач може почати роботу з модулем.

ВХІДНІ ДАНІ

Перед початком розробки було розгорнуто DEVOPS-система на засобах Jira, Git, Kubernets. Завдання на написання коду та контроль виконаних завдань виконується в DEVOPS системі.

Git Flow організовано таким чином, що будь-які дані, які потрапляють до репозиторію будуть запускати скрипти, що в свою чергу запустять команди зборки проекту. Після проведення зборки роботами Kubernets код потрапляє на сервер, який в свою чергу перезапускається та повідомляє розробнику про готовність внесених змін. Також реалізовано доступ до серверу шляхом підключення через VPN до локальної мережі сервера за допомогою SSH. Таким чином дані можуть редагуватись безпосередньо на сервері, але при наступній зборці проекту будуть замінені.

ВИХІДНІ ДАНІ

В ході розробки програмного продукту були редаговані та деталізовані існуючі таблиці бази даних, створено засоби взаємодії між користувачем і сервером у межах модуля, розширено стандартні засоби виводу інформації та додано новий функціонал до системи Odoo у межах створеного модуля бухгалтерського обліку.

Завдяки створенню модулю бухгалтерського обліку на базі ERP Odoo 13 Community Edition ми виконали технічне завдання науково-дослідницької роботи, а також вирішили необхідні питання, а саме:

- Створення звітів доходів, коштів, балансів, рахунків;
- Формування бухгалтерського обліку;
- Вирішено проблему частої зміни форми оподаткування;
- Експорт даних з інших подібних систем;
- Створення документів та довідників;

Завдяки чому сприяли створенню аналітики фінансів всередині компанії, зменшили витрати людино-годин на створення звітів, зменшили ризики помилки у складанні звітності.